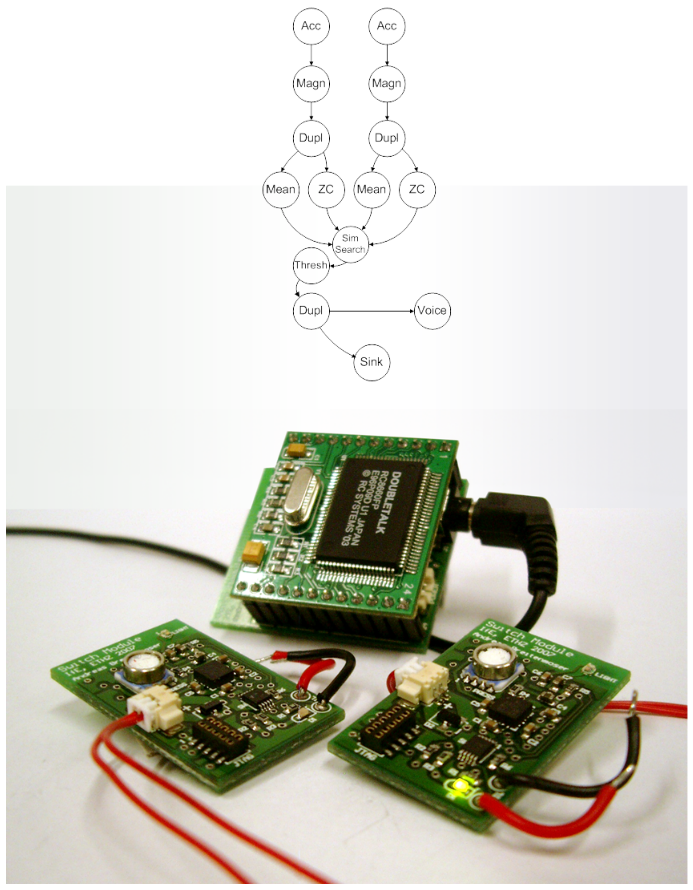
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Electronics Laboratory
Prof. G. Tröster
Autumn Semester 2007

Master Thesis

Titanic Smart Objects



Author:
Andreas Breitenmoser

Supervisors:
Clemens Lombriser
Andreas Bulling

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Electronics Laboratory
Prof. G. Tröster
Autumn Semester 2007

Master Project

for
Andreas Breitenmoser

Titanic Smart Objects

Betreuer:	Clemens Lombriser, ETZ H64
Stellvertreter:	Andreas Bulling, ETZ H64

Ausgabe:	24. September 2007
Abgabe:	21. March 2008

Introduction

The Wearable Computing Laboratory has focused in the past years on the recognition of people using wearable sensors [1, 2]. However, using sensors only worn on the body only pose restrictions that make it much harder to recognize context information. By using wireless sensors integrated into objects in the environment, so-called *smart objects*, the information available for the inference of the current context of a person is greatly improved. Early examples of smart objects are the MediaCups [3] and Smart-Its [4] projects. Integrating wirelessly connected objects into a distributed recognition system for human activities has recently been investigated in the DART project [5].

This project will focus on the dynamic integration of sensors and the distributed inference of context information. To this end, a number of smart objects with sensors and actuators will be built, which will be integrated into Titan, a distributed processing environment for context recognition algorithms developed here at the IfE [6]. The goal is to demonstrate the functionality of the complete system using a small game which recognizes different actions performed by the user with or without objects and checks whether a number of required actions have been executed. The exact setup and functionality will be specified during the project.

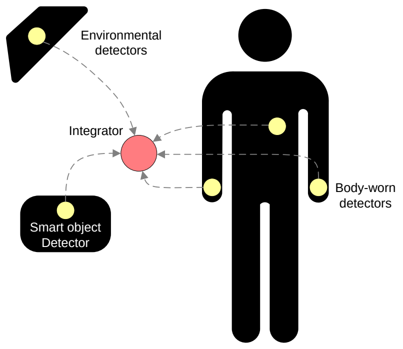


Figure 1: Detectors and integrators.

Implementation

The hardware basis of the smart objects are the Tmote Sky Mini modules[1]. Those modules need to be extended by sensors (accelerometers, proximity sensors, magnetic field detectors, light sensors, microphones, temperature sensors, . . .) and actuators (leds, displays, speakers, motors, electrically active polymers, . . .). The corresponding hardware needs to be designed and built into objects used for the demonstrator. For the on-body sensors, the SensorButton [7] could be used in its wrist-worn casing.

An optional extension would be to use the Wireless Gateway [8] to connect to a mobile phone or PDA, where a graphical user interface might be shown to the user for interaction.

The recognition of the activity will be based on multiple sensing devices as displayed in Figure 1. Every *sensor-detector* node will perform an initial detection of how it is manipulated, and send the result to a *integration* node, which compares the information received from the different participating smart-objects and decides which overall activity is currently performed by the user. Depending on this result, actuators will be activated, or different detectors will be loaded on the smart objects.

How a recognition of activities can be performed in Wireless Sensor Networks has been worked on by Osmani, which defines Context Zones [9], Li, integrating the recognition into its middleware [10], or Predd [11], which discuss how nodes can learn what they should recognize.

The recognition algorithms will be built into the Titan framework [6]. This framework allows to execute applications described as data processing task graphs on dynamic Wireless Sensor Networks in a distributed fashion. As it allows for a quick reconfiguration of the network, it is ideally suited to adapt the currently executed recognition algorithm to the currently needed situation. For this project, it needs to be extended with suitable recognition tasks and an algorithm for the distribution and dynamic integration of sensors. This can be done by saving a number of task graphs into a database and execute those when all necessary tasks

[1]http://www.moteiv.com

and sensors are available in the network.

Example demonstrator

As an example and starting point for investigations, the Tmote Mini sensor nodes should be equipped with an accelerometer and be integrated into a foam dice as can be bought in toy stores. The acceleration data can be used to determine whether a person is carrying the dice [12], who is carrying the dice [13], or just to relate two objects with each other [4]. A first demonstrator can be built using those techniques: A number of foam dice could be provided which can be picked up by a number of people. Each would choose on their own how many to pick up and shake them to gather the dice. After rolling the dice, each person will get their own sum of the dice eyes rolled.

Task Description

1. Make yourself familiar with the state-of-the-art in the recognition of context in the area of pervasive computing. The references in the introduction serve as a starting point.
2. Define a demonstrator setting with multiple smart objects, on-body sensors, the activities that should be recognized, and the feedback that should be generated.
3. Design and develop the hardware needed for the smart objects.
4. The operating system for the smart object nodes will be TinyOS 2.0[2] [15]. Write drivers for the peripherals of the hardware
5. Evaluate recognition algorithms for the smart objects and the integration of the different detectors. A tool that may help is the Weka toolkit[3].
6. Implement the recognition algorithms for Titan.
7. Test and evaluate the performance of the algorithms you have selected.
8. Document your work in a report.

[2]http://www.tinyos.net
[3]http://www.cs.waikato.ac.nz/ ml/weka/

Project Management

General guidelines

- Create a work plan with milestones and monitor your progress. Unforeseen problems may make it necessary to change this plan. Such changes should be documented.
- You will have a work place in ETZ G66 with 1 PC
- Present your project in a short talk (about 5min) at the beginning of the semester on October 1st 2007.
- Present the results and finish your report of your project until March 21st 2008.
- Meet with your supervisor on a regular basis to discuss the work progress.

Delivery

- Hand in two versions of your report to your supervisor or his substitute until March 21st 2008.
- Design a short web page describing what you have done in this project. This page will be added to the Wearable Lab website `http://www.wearable.ethz.ch`.

Zürich, den 24. September 2007

Prof. G. Tröster

Bibliography

[1] C. Lombriser, N.B. Bharatula, D. Roggen, G. Tröster. On-Body Activity Recognition in a Dynamic Sensor Network. 2nd International Conference on Body Area Networks (BodyNets), 2007

[2] T. Stiefmeier, G. Ogris, H. Junker, P. Lukowicz, G. Tröster. Combining Motion Sensors and Ultrasonic Hands Tracking for Continuous Activity Recognition in a Maintenance Scenario. *10th IEEE International Symposium on Wearable Computers (ISWC)*. 2006

[3] M. Beigl, H.W. Gellersen, A. Schmidt. Mediacups: experience with design and use of computer-augmented everyday artefacts. *Computer Networks*, vol. 35, nr. 4, pp 401–409, 2001.

[4] L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, H.W. Gellersen. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. *Third International Conference of Ubiquitous Computing (Ubicomp)*, p. 116. 2001

[5] O. Amft, C. Lombriser, T. Stiefmeier, G. Tröster. Recognition of user activity sequences using distributed event detection. In *European Conference on Smart Sensing and Context (EuroSSC)*, 2007

[6] C. Lombriser, D. Roggen, M. Stäger, G. Tröster. Titan: A Tiny Task Network for Dynamically Reconfigurable Heterogeneous Sensor Networks. *15. Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, pp. 127–138, 2007.

[7] M. Graf. Wearable Sensor Package 3. Masterarbeit, ETH Zürich, 2005

[8] F. Schenkel. Wireless Gateway. Semester Thesis, ETH Zürich, 2007

[9] V. Osmani, S. Balasubramaniam, D. Botvich. Self-Organising Object Networks using Context Zones for Distributed Activity Recognition. *Proceedings of the Second International Conference on Body Area Networks (BodyNets)*. 2007

[10] S. Li, Y. Lin, S.H. Son, J.A. Stankovic, Y. Wei. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. *Telecommunication Systems*, vol. 26, nr. 2, pp 351–368. 2004

[11] J.B. Predd, S.R. Kulkarni, H.V. Poor. Distributed Learning in Wireless Sensor Networks. *IEEE Signal Processing Magazine*, vol. 23, nr. 4, pp. 56–69, 2006.

[12] J. Lester, B. Hannaford, G. Borriello. “Are You with Me?” – using accelerometers to determine if two devices are carried by the same person. In *2nd International Conference on Pervasive Computing (Pervasive)*, pp. 33–50. 2004

[13] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *4th International Conference on Pervasive Computing (Pervasive)*, pp. 144–161, 2007

[14] M. Strohbach and H. Gellersen. Smart clustering – networking smart objects based on their physical relationships. In *5th IEEE International Workshop on Networked Appliances*, pp. 151–155, 2002

[15] D. Gay, P. Levis, D. Culler. Software Design Patterns for TinyOS. *ACM Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, pp. 40–49. 2005

Contents

List of Figures

List of Tables

Abstract

This project presents *smart objects* that run *Titan*, the Tiny Task Network framework [1]. Titan reprograms wireless sensor network nodes dynamically and allows the adaptation of distributed context recognition algorithms at runtime. Recognition algorithms are executed as task graphs on the nodes. Two hardware modules, a Switch Module and a Voice Module, equipped with different sensors and actuators, such as a voice synthesiser, have been developed to build the smart objects. Based on these modules new functionality is added to Titan: the task pool is extended, a finite state machine for controlled reconfiguration is integrated and online activity recognition is performed. The activity recognition is studied with focus on efficient computation and communication to counter the resource constraints on networked embedded devices. Demonstrator scenarios have been evaluated and the functionality of the system is demonstrated by using a small game which recognises different actions performed by the user and checks whether a number of required actions has been executed. The project analyses recognition and discrimination of shaking movements. Recognition rates of 73.5% and 76% could be achieved with a Similarity Search classifier using an optimised set of features minimising the communication overhead for distributed recognition.

Chapter 1

Introduction

Wireless sensor networks (WSN) have already found their way into a wide range of application areas, such as environment, health, home, security and military. According to [2] the largest number of breakthroughs in the WSN field will happen in the next 5 to 10 years and market studies predict the deployment of over 100 million WSN nodes in 2010. In certain application domains WSN might become an integral part of our all lives.

WSN especially include small computing devices cooperating through radio communication. Wirelessly communicating modules with microprocessors, sensors and actuators built into every-day objects lead to cooperating *smart objects*. Such smart objects can either be portable devices, parts of Body Sensor Networks worn on the body, or immobile appliances mounted in the environment.
Smart objects aim to provide added value in terms of usability and functionality. To benefit the user in his every-day life smart objects need to be ubiquitous, distributed and seamlessly integrated in the environment. Such smart and mobile devices provide the users with possibilities to interact implicitly and unobtrusively with the computing system. The data collected by smart objects makes information available that can be used to compute the context of users in the vicinity. This context awareness serves to improve the way the users interact with the objects.
The Wearable Computing Laboratory has concentrated on activity recognition using wearable sensors in the past years [3, 4, 5]. If only body-worn sensors are used the context recognition task will be a complex one. The combination of wearable sensors and sensors integrated into smart objects in the environment reduces the complexity and improves the context information.

Demonstrative examples of WSN nodes and smart objects were presented (cf. section 1.1 further below). But to make WSN establish on a broad market and to launch successful products based on smart objects the underlying technologies must continue to improve. Therefore a couple of challenges have to be addressed: scalability, production costs, hardware constraints and power consumption, sensor network topology, mobility and fault tolerance [6].

This project addresses the distributed inference of activity information. A number of smart objects with sensors and actuators is built and online algorithms for the distributed recognition of activity implemented. The recognition algorithms are integrated into TinyOS [7, 8] and built into the *Titan* (Tiny Task Network) framework [1].
The goal of this work is to move Titan forward. Small networked modules are developed. These hardware platforms comprise components and interfaces with differing characteristics, which allows for an expanded study of Titan.
The project will mainly focus on the efficient computation and dynamic adaptation of algorithms while activity recognition is conducted. Computations should preferably be executed directly on the single nodes in the network, thus reducing the need for radio communication and saving energy. Titan reconfigures individual nodes to update the execution of an algorithm in a changing network. While people are interacting with smart objects the number and type of active objects may vary at a time. Furthermore, as smart objects are moved or as people themselves move around settings are continuously changing. Distributed context algorithms must be constantly adapted to occurring heterogeneity and momentary availability in WSN.
The modules running Titan are applied in a demonstrator. The demonstrator presents the process of dynamic reprogramming of the WSN in a vivid way. A user can perform different actions. Whether an action has been executed and which action it concerns should be recognised and the smart objects are eventually going to react through their actuators.

This project report is organised as follows. Chapter 2 introduces various ideas of smart object scenarios and points out how we arrived at the final decision for the demonstrator. Chapter 3 gives an overview of the system architecture and describes the hardware designed. In chapter 4 the activity recognition algorithms are presented, followed by a detailed evaluation of recorded accelerometer data. Information about the software including the operating system and the drivers is found in chapter 5. Incorporation of the algorithms into Titan is described. Thereon, chapter 6 summarises the main results in a conclusion. Chapter 7 highlights future work in the field and finally concludes this report.

1.1 Related Work

Research into WSN was initially driven by military applications (Smart Dust [9]). More recently, the growth of the WSN market has increased in other civilian areas: various applications in monitoring environmental conditions [10, 11], agriculture [12], and health care [13, 14] emerged. Environmental control in buildings enables the control of air flow and temperature in different parts of rooms and the reduction of energy consumption. In home automation and smart environments, sensors and actuators are integrated into domestic devices and allow the interac-

tion with users in a much more intuitive way.
In the MediaCups project [15] a usual coffee cup is augmented with sensors, additional processing and communication. The SensorButtons [16] resemble ordinary buttons and can be unobtrusively integrated in garments as networked wearable computing devices. [17] contributes to research in ubiquitous computing and activity recognition with a living laboratory. PlaceLab, a smart environment, is created by embedding sensing devices into the furniture and the fabric of the architecture.

Smart-Its [18] presents a visionary new approach of connecting computational entities and provided some inspiration for our project. Context proximity enables implicit and explicitly user-controlled connections among two objects by shaking them together. Our project takes the idea a step further by embedding the approach into a complete system with more than two objects. [19] and [20] also carry on the idea of Smart-Its but they both differ in their focus and make the analysis of context information using a FFT. Calculations in the frequency domain may rapidly exceed the limited computing power on sensor nodes. In addition, [19] focus on walking and does not provide online activity recognition. A method for constructing dynamic groups based on movement information is proposed by [21]. In our project different types of activities are carried out and different types of sensors and actuators get included. Recent research at the Wearable Computing Laboratory (DART project [22]) has investigated a distributed architecture for the online recognition of user activity sequences in a car assembly scenario. This project contributes to DART by realising the distributed online implementation in a WSN.

Chapter 2

Scenario

The final outcome of the project is a demonstrator. Its main part comprises smart objects which together build a wireless sensor network. The demonstrator should include different HW components and clearly demonstrate how the complete system reconfigures dynamically.

We started an idea finding process and held brainstorming sessions at the beginning of the project's work to find a meaningful composition for the demonstrator. First a set of valuable ideas was collected, then the single concepts were evaluated using the following criteria (see also section 2.2):

- A focus of this project is the *dynamic reprogramming* of wireless networks and it is of high priority to make use of it in the show case.
- As we are talking about sensor networks several devices should be involved so that computation tasks can be *distributed* over a whole sensor network.
- To be really smart, objects should apply *context recognition* and even include learning methods.
- For this purpose *sensors* must be introduced. Particularly motion sensors, such as accelerometers or gyroscopes, should be used. Besides, adding *actuators* will even make the smart objects proactive.
- *Wearable systems* would be a favoured extension and should fit in the scenario reasonably well.
- The appeal or *eye-catching* portion of an idea was judged.
- Eventually, the idea had to be *realisable* within the project's lifetime.

2.1 Ideas for a Demonstrator

The following subsections present ideas and reveal more insight to their attractiveness. A possible scenario is described in each case and the technical requirements are identified.

Whenever wearable devices are mentioned, especially the SensorButton [16], developed here at the Wearable Computing Laboratory, is concerned as a valuable candidate. It could be worn in its wristwatch casing for example to capture hand movements.

2.1.1 Control by Movement

Wearable acceleration sensors have been used in many projects (e.g. [4, 23, 24]). When wearing, smart objects can be controlled by own body movement. We especially liked the idea of an unconstrained user interface that makes controlling more intuitive and natural. Such a user interface provides three complete degrees of freedom enabled by free body movement in space and includes the option of extending even to further degrees of freedom, such as voice, which can be captured by microphones. Moving things by gestures only, brings a sort of magic into play, which makes the application more attractive. Earlier projects that made interesting contributions in this vein are [25] and [26]. Those approaches demanded from users to hold a wand as teleguidance. One could think of a demonstrator that allows for the same actions. As the hardware would be integrated into clothing and objects themselves the user would be bearing nothing anymore.
One could think of several demonstrators:

- Sensors are to be attached to the user's extremities and mobile smart objects, e.g. miniature toy helicopters, are controlled by movements of arms or legs.
- Music and light could be controlled by bare body movement. Sound and light in a room might change depending on the activities of a person. So a personal sound surrounding can be generated.

According to the given criteria we have reservations that the focus is more on the human-machine interface than on the network aspect including the distribution over the network. Additionally, this scenario comes rather short of a scientific purpose. Gestures needed for control seem to be relatively simple. Activity recognition does not get more intricate until several objects are controlled simultaneously. Moreover, the challenging bit lies more on the hardware side, trying to hack the hardware of the remote-controlled objects.

2.1.2 Assignment of Gestures

This approach would like to make use of assigning a gesture to a certain smart object, the gesture being fundamentally typical for that object. The underlying idea is originally based on the concepts stated by Holmquist, Mattern et al. [18].

Again a motion sensor is worn on the wrist. The hand-held smart object is also augmented with a motion sensor and further includes additional sensors and actuators. When the sensor on the wrist and the sensor on the object are moved together the object learns the movement and additionally detects information about

Smart objects	Wearable device to detect movement Miniature vehicles (cars, helicopters, robots, ...) which are moved
Number of nodes	1-2 wearable devices 1-3 miniature vehicles
Topology	Star network, 1-hop-network if vehicles also communicate among each other; heterogeneous
Sensors	1 accelerometer and 1 microphone per wearable device Proximity sensor for each vehicle
Actuators	Motors for the vehicles
Algorithms	Gesture recognition, swarm algorithms
Wearable devices	Yes, devices are attached at different parts on the body
Dynamic reconfiguration	Yes, if number of objects to control is changing or if different tasks are executed during control

Table 2.1: Technical requirements for "Control by Movement".

the context. Later the person performs the same movement or gesture a second time but in a distance of several meters away from the object, i.e. without holding the object in the hands. The wrist-worn sensor node detects and forwards the action. Thus the remote object is able to recognise the gesture which was learnt before and to react correspondingly by using the collected context information.
Below two examples for such gesture assignment are given. For demonstrators, both active and passive systems can be considered:

- *Smart cup*, augmented with acceleration and temperature sensors: When the gesture "drinking from a cup" is performed, while lifting the cup, the temperature of its content is measured.

  Passive, i.e. without any actuator: If the gesture is performed by the same person out of eyespot a second time later on and if the drink in the cup is still hot right at that moment, the information could be that the cup is used by another person at present.

  Active, i.e. with an integrated heater as actuator: Let us assume, the content of the cup is not of the same temperature it had initially when drinking, i.e. when the cup learnt the gesture "drinking form a cup". If the same person performs the gesture remotely the second time, the cup receives the signal to heat the drink up to the temperature once sensed while drinking.

- *Smart keys*, augmented with acceleration and light sensors: If the gesture "unlocking" is performed by turning a key in the lock the ambient brightness is measured.

  Passive, i.e. without any actuator: If the light sensor on the key detects that the environment is light right at the moment the same person again executes the gesture without having the key with him this time, it is concluded that the key is not stowed in a pocket but is just used by someone else.

  Active, i.e. with a LED lamp as an actuator: Additionally, a LED lamp is attached at a bunch of keys. If the person performs the learnt gesture again at night, the key will recognise the context and turn on the light at the bunch so that the whole bunch can be found or the right key can be picked or even the keyhole can be located in the dark.

Smart object	Wearable device to detect gesture Every-day objects, such as cups or keys, which are controlled
Number of nodes	1 wearable device 3-5 every-day objects
Topology	Star network; heterogeneous
Sensors	1 accelerometer for the wearable device For each object: 1 accelerometer and at least one further sensor (microphone, temperature sensor or light sensor) for context recognition
Actuators	One actuator (speaker, heater or LED) per object as sensor's matching part
Algorithms	Gesture recognition, context recognition
Wearable devices	Yes, device is worn on the wrist
Dynamic reconfiguration	Yes, but only if number of objects to control is changing

Table 2.2: Technical requirements for "Assignment of Gestures".

The lack or at least the limitation of dynamic reprogramming is seen as one major downside of this idea. A solution could be provided by a simple configuration protocol. This does not include interaction between multiple devices, so there is no need for distribution over a network.

2.1.3 Sensitive Plush Toy

The subject for this demonstrator are smart toys, e.g. plush toys, which set up according to their vis-a-vis, react and adapt. [27] shows what smart systems can

be created based on simple plush toys and serves as source of inspiration to that effect.
The functionality of a possible demonstrator can be described as follows:

- Fondling, hugging, shaking and striking, approaching or speaking – all these actions could be recognised and learnt by such smart plush toys. Therefore diverse sensors must be used including accelerometers, pressure, touch or proximity sensors and microphones.

- As described in the previous subsection 2.1.2, the same actions can be repeated in a certain distance away from the toy not holding the toy in the hand anymore. The toy shows reactions according to the gestures performed. Again this remote interaction relies on body-worn sensors.

- Environmental conditions will also contribute to the manner the plush toys behave. If the toy is left back in the dark or if it feels cold (using light or temperature sensors for example) the toy stores these contexts and can remember them. This way completely individual toys are created.

- Further, aspects of group dynamics could be introduced. In a group of plush toys we could imagine toys become "jealous" and forget the learnt gestures all the more whenever a single toy is "preferred", i.e. used more often than all the others.

Smart object	Wearable device to detect gesture Plush toys which are influenced
Number of nodes	1 wearable device 3-4 plush toys
Topology	1-hop-network; heterogeneous
Sensors	1 accelerometer for the wearable device For each toy: 1 accelerometer and at least one further sensor (microphone, touch sensor, temperature sensor or light sensor) for context recognition
Actuators	At least one actuator (voice/speaker, vibration motor, LED) per object
Algorithms	Machine learning (supervised/unsupervised), gesture and context recognition
Wearable devices	Yes, device is worn on the wrist
Dynamic reconfiguration	Yes, if number of objects to control is changing or if different tasks are executed depending on how the toys are treated

Table 2.3: Technical requirements for "Sensitive Plush Toy".

Here we see the difficulties in terms of the system's complexity. Especially the software side poses a problem. To make the toys learn continuously, online learning is needed. Online learning on sensor network nodes is a research topic that is still open. The project time does not permit the development of the hardware and the additional investigations needed for online learning. This could be a future separate project.

2.1.4 Picking Up Lorry

Picking up lorry means a robot or a pick-up lorry with gripper arm and loading platform.
[21] describes an application of WSN in the field of transport and logistics. Sensor nodes with accelerometers are tagged to shipped products. Whenever a mote on a product shows a motion profile different from the profiles of the remaining products in the group an alarm is set off to indicate that the product is missing in its group. A demonstrator that utilise those principles could work this way:

- The lorry moves forth as long as all the motes or smart objects, respectively, are loaded. When an object gets lost, the lorry is dynamically reprogrammed and changes its behaviour, i.e. it turns back to collect the lost object. The lorry may receive the position of the lost object from the object itself. After the object is found and loaded up again the lorry continues on its path.
- The lorry can also pick up new objects which it encounters on its way.
- Depending on the number and type of loaded or lost objects the behaviour of the lorry can change. Hence the shipped objects determine the transport to some degree.
- A possible application could be in the field of environmental monitoring. Sensor nodes are distributed in the environment, then measurement data is recorded for a certain period and finally the sensor nodes get recollected autonomously.

Indeed, one of the caveats of this demonstrator is the absence of a wearable device. Even if only parts of the lorry and its gripper were to be developed by ourselves time would hardly suffice. A further difficulty is the localisation of the sensor nodes. Localisation by radio using the signal strength is unprecise and not practicable indoors. Solutions with ultrasound must be considered. Although the system would be interesting for sure, the focus is too much on robotics and thus out of our scope.

2.1.5 Vita Parcours

Vita parcours is the Swiss term for a fitness trail which is usually layed out in recreation areas and consists of several posts each offering a specific exercise.

Smart object	Lorry or robot with a gripper arm Boxes which are collected and picked up
Number of nodes	1 node on the lorry 5-10 boxes
Topology	Star network, 1-hop-network if boxes also communicate among each other; homogeneous
Sensors	1 accelerometer (and an image sensor) on the lorry For each box: 1 accelerometer and 1 microphone
Actuators	Motors in the lorry, 1 speaker LEDs for the boxes to indicate that they got lost
Algorithms	Context recognition
Wearable devices	No
Dynamic reconfiguration	Yes, if number of boxes is changing or when lorry changes its behaviour due to a loss of boxes

Table 2.4: Technical requirements for “Picking up Lorry”.

Such fitness trails could be augmented by adding sensor nodes to the posts as well as the sports equipment. In addition, the sportsman should wear devices on the body that allow for health monitoring on the trail and always arrange for optimal combinations of exercises:

- The exercises depend on present strength and dexterity of the users and adapt to personal characteristics. Wearable sensors including electrocardiogram (ECG) devices could be used for monitoring the runner while he is on the way between the posts as well as while he is conducting the exercises. Training is optimised.

- Not only the state of health is observed but also the actions performed at the posts are analysed. Sensor nodes are embedded in sports equipment to record motion sequences. Smart sports equipment helps to prevent from overstresses and warns against injurious misuse of the equipment.

- Instead of having the posts communicating with each other the posts would only communicate with the wearable devices. The user really transports the information from one post to the other by carrying the wearable devices.

To appear authentic the complete set-up must be installed outdoors. In building this demonstrator indoors as it would be the case, the installation loses much of its impression. Of course, data collection can also be carried out in a simulated environment and the demonstrator could be downscaled somewhat. The “Vita Parcours” scenario might be less suitable for short demonstrations, too. A further challenge is to arrive at a satisfying reliability in given time.

Smart object	Wearable devices to detect movements Posts which provide different exercises Pieces of sports equipment which are used for the exercises
Number of nodes	1-4 wearable devices 2-3 posts 1-2 pieces of sports equipment per post
Topology	1-hop-network; heterogeneous
Sensors	1 accelerometer and 1 microphone per wearable device, optionally: ECG electrodes 1 temperature sensor, 1 light sensor and 1 humidity sensor per post For each piece of sports equipment: 1 accelerometer and 1 microphone, 1 pressure sensor depending on the sports equipment
Actuators	Voice/speakers at the posts, LEDs on the post and on the sports equipment to give feedback concerning the exercises
Algorithms	Recognition of motion and context
Wearable devices	Yes, devices are worn on the extremities
Dynamic reconfiguration	Yes, each time a new post is reached

Table 2.5: Technical requirements for “Vita Parcours”.

2.1.6 Interactive Museum

The visitors can interact with smart objects in museums and influence the exhibition by their actions. So each person receives an individual impression on his visit. The exhibits and the rooms in the museum react on visitors’ activities and provide interactive learning. Each visitor wears a wristwatch-like device on the body to detect his movements, noises and voices of people are recorded by microphones and cameras may provide localisation and catch further context. One can think of completely new possibilities in connection with a visit to a museum:

- The visitor is guided through the exhibition depending on his own behaviour. The interests of the visitors are recognised and the available interactions are adjusted accordingly.

- The cumulative behaviour of all the visitors or the number of visitors may influence the exhibition, too.

- Every time a person visits the museum it is stored what the person has seen so far and what not yet. This information can be used for visits to the museum in the future.

Smart object	Wearable device to detect visitors' activities Exhibition areas which inform about different topics Various exhibits, such as weapons, ancient or technical instruments which visitors can try out
Number of nodes	1 wearable device 2-3 exhibition areas 2-3 exhibits per exhibition area
Topology	1-hop-network; heterogeneous
Sensors	1 accelerometer and 1 microphone per wearable device 1 microphone, optionally 1 camera per exhibition area For each exhibit: 1 accelerometer and 1 microphone, optionally 1 air pressure sensor
Actuators	Voice/speakers and displays at the exhibition areas
Algorithms	Recognition of motion and context
Wearable devices	Yes, device is worn on the wrist
Dynamic reconfiguration	Yes, each time a new exhibition area is reached and each time the museum is visited again

Table 2.6: Technical requirements for "Interactive Museum".

The downsides are quite similar to those of the "Vita Parcours". Even an installation for emulating an authentic exhibition area must be of a certain size and building such a demonstrator is highly time-consuming.

2.1.7 Foam Dice

According to the description in the mission statement smart objects can be created from foam dice. Foam dice, available from toy stores, are extended by 3D-accelerometers. Accelerometer data allows to infer whether dice are moved, if they are picked up and shaken by the same person [20] or if they are thrown.

As dice are basic elements of many games several applications could be thought of:

- A number of players picks up a number of foam dice each player choosing on his own how many to take. After everyone has shaken the dice to gather them, the dice are rolled. In the end all the players will get their own sum of numbers on the dice. Such a set-up could be used in common dice games, such as Yatzi, in calculating and displaying the results immediately round by round.

- Smart dice could also be part of an overall system. Playing the dice might serve in a preliminary step as a selection mechanism to decide what actions, e.g. in a game, should follow.
- A cube, e.g. fabricated as foam dice, can also be used as an interface. By turning, shaking or squeezing the dice functionalities of a system are selected. Applications in the field of building automation could be targeted for example.

Smart object	Foam dice which detect movements and orientations
Number of nodes	1-5 foam dice
Topology	1-hop-network homogeneous
Sensors	For each foam dice: 1 3D-accelerometer, optionally further sensors (microphone, pressure or light sensor)
Actuators	No
Algorithms	Recognition of motion and context
Wearable devices	No (when carried, foam dice behaves much like a wearable device worn at the extremities)
Dynamic reconfiguration	Yes, each time a new dice is added to a group of dice and at state transitions, e.g. transition from pairing of dice to playing the dice.

Table 2.7: Technical requirements for “Foam Dice”.

The foam dice represents an example of a simple object which is augmented by additional computing and communication. It is a good indicator for the main difficulties that occur in relation with smart objects. Aspects of dynamic reconfiguration, HW constraints and context recognition can be investigated while ease of integration is ensured.

Unfavourable is the lack of actuators. The range of sensors is limited as well. Most likely, only one sensor, an acceleration sensor, is used. Besides, no wearable system is involved in this scenario.

2.1.8 Game in Interactive Room

The players must solve a riddle or execute tasks to complete the game successfully. The game takes place in a room and the players are supposed to interact with the equipment of the room on their mission. Smart objects are created by embedding sensors and actuators into furniture and appliances in the room.

A scenario could be devised with the goal to escape from the room. The player has to use some tricks, i.e. he will interagate with several smart objects, to find the key and succeed in opening the door. The necessary interactions will include:

- Objects must be placed at certain locations or several objects must be moved together.
- In approaching an object an event is activated, e.g. light or noise is generated.
- Wearing a device on the wrist one has to control objects which are placed at different positions in the room simultaneously. This will only be achieved if all the objects react on a single gesture.

Such a demonstrator also offers the possibility to involve a computer. So communication could be expanded in connecting other rooms or further devices. In addition, the computer's display can be used for explaining the next task to the player.

Eventually, this demonstrator could take advantage of the "Foam Dice" scenario, which was mentioned in the previous subsection. As the foam dice already represent a game themselves the entire demonstrator could evolve from initial playing the dice.

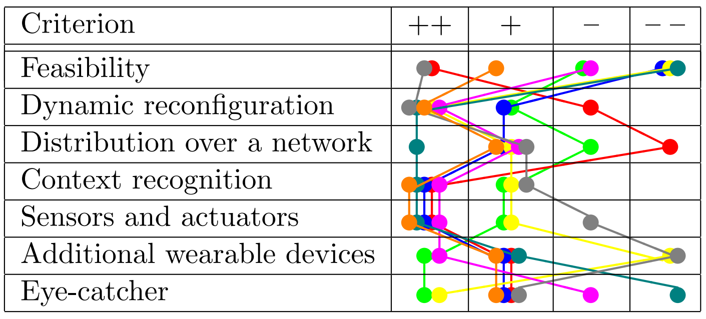
Smart object	Wearable devices to detect players' activities Every-day objects and toys which are used to cope with the requests
Number of nodes	1-2 wearable devices 5-10 every-day objects
Topology	1-hop-network; heterogeneous
Sensors	1 accelerometer and 1 microphone per wearable device For each object: 1 accelerometer, 1 light sensor, optionally 1 microphone or 1 air pressure sensor
Actuators	One actuator (voice/speaker, motor or LED) per object 1 PC to display the requests
Algorithms	Gesture recognition, context recognition
Wearable devices	Yes, devices are worn on the wrist
Dynamic reconfiguration	Yes, each time a new task is set

Table 2.8: Technical requirements for "Game in Interactive Room".

This scenario is quite balanced. Attention might be payed to conception. The construction of such a game asks for a good concept, i.e. the set of tasks should be meaningful and unified to a certain extent.

2.2 Choice of Demonstrator

Table 2.9 lists all the ideas considered and rates the concepts in a decision matrix. The evaluation directly results from considerations made in the previous subsections using the criteria mentioned at the beginning of section 2.



🟢	Control by Movement	Score: 5
🔴	Assignment of Gestures	Score: 6
🔵	Sensitive Plush Toy	Score: 5
🟡	Picking up Lorry	Score: 3
🟣	Vita Parcours	Score: 8
●	Interactive Museum	Score: 5
⚪	Foam Dice	Score: 8
🟠	Game in Interactive Room	Score: 13

Table 2.9: Decision matrix: The ideas were rated according to the listed criteria. The points are added up to the score whereas “feasibility” and “dynamic reconfiguration” are weighted twice.

For evaluation the ideas were scored as shown in Table 2.9. The score is received by giving the two most important criteria, “dynamic reconfiguration” and “feasibility”, double weight while counting the others only once.

If one looks at the results some points stand out. Table 2.9 shows a dependency between the criteria “dynamic reconfiguration” and “distribution over a network”, i.e. a scenario rated low for the one criterion also achieves a low score for the other. An increase in distributed computation typically comes along with an increased limitation of nodes’ hardware resources. Hence parts of tasks can be executed on the nodes only, raising the need for dynamic reconfiguring. Considering criterion “eye-catcher” some scenarios appear to be much less attractive. These scenarios are rather costly for a demonstrator implementation and must be downsized a lot, which is mainly the reason for the lower score. Criterion “feasibility” shows the largest variation. Unfortunately, quite a number of ideas appeared to be too time-consuming and rather complex.

"Foam Dice" and "Vita Parcours" are among the top ranked scenarios. "Game in Interactive Room" seems to be the overall favourite. Flexibility, vividness and simplicity make the difference. A broad range of sensors and actuators can be factored, tasks can be distributed over several devices placed in the room and various activities can be the target of recognition. As it is about a game, spectators can be additionally motivated by just inviting them to participate in the game. Simple objects are often used in games, which will help saving effort concerning demonstrator design, assembly and tests and give enough time to concentrate on the algorithmic side later in the project.

We use the dice from scenario "Foam Dice" for the introduction of the game. The foam dice are smart and recognise movements performed with them: gathering the dice, simultaneous shaking the dice in a correlated manner by a single person and rolling the dice, as described in subsection 2.1.7. The game is started when a player shakes the dice together and throws them to randomly select the subsequent course of the game depending on the total sum of the dice' score.

After the dice are rolled the game is designed to continue by a mission. For completion of the missions the foam dice and a set of LED lamps, so-called candle lights, are provided. Similar to [28] we included light control in our demonstrator. With light the effect of the actuator is easily and immediately noticed. Moreover, the lamps provide a basis for further study of node failures and recovery, which could be simulated by just switching lamps off. As a reaction the network is reconfigured and further lights may turn on in exchange.

Following listing suggests possible missions that could occur in the game and provides ideas for further implementations:

- Switching candle lights on by turning the room lighting off (detection of difference in ambient light).
- Switching candle lights on by raising the candles (detection of difference in air pressure).
- Weeping out lights by shaking (detection of acceleration).
- Weeping out lights by rolling the dice nearby (based on noise of the radio signal).

Ideas from other scenarios could also be involved:

- "Vita Parcours" (cf. section 2.1.5) brings up the concept of different posts which set up a new configuration each time a person wearing a portable device is approaching. The context is successively transported from one location to another through the wearable device. The same effect can be achieved by playing the dice: Replaying the dice is similar to a change of posts and the context stored up to that point could impact the action coming next, i.e. the course of missions might not be absolutely random but influenced by previous achievements in the game.

- “Assignment of Gestures” (cf. section 2.1.2) might simply be one of the missions that must be fulfilled during the game: Weeping out lights remotely by gestures.

[29] explored speech- and gesture-based interfaces in the context of intelligent environments. Brumitt and Cadiz addressed the issue how users might control lights in a home of the future. A preference for voice interfaces is reported. We introduced a voice synthesiser module in the demonstrator to provide immediate feedback for the player. The number on the dice is called out by the module for example. Voice is a natural interaction form. A voice synthesiser is an output device that could also be of interest for broader applications in smart environments and especially in wearable systems.

The entire set-up of the game is shown in Figure 2.1. Since the missions are chosen by chance dynamic reconfiguration of the foam dice and the candle lights is necessary. Dynamic reprogramming of a wireless network can be clearly demonstrated.

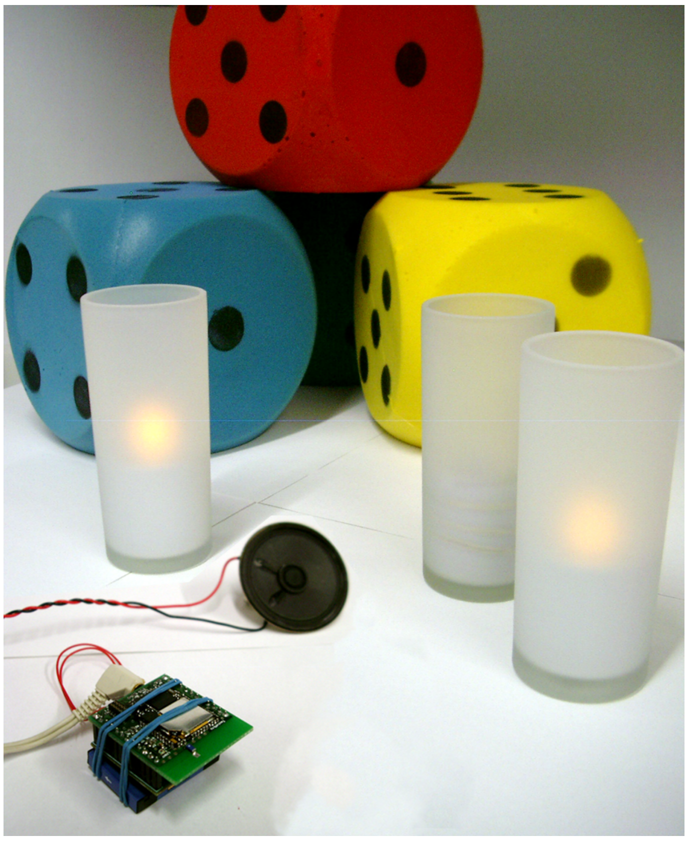


Figure 2.1: The demonstrator consists of three parts: foam dice, candle lights and a voice synthesiser module. The foam dice are used by the players to select a task randomly at the beginning of the game or at the start of each new round. The candle lights are induction lamps augmented with sensors which allow the players to take further actions. Both, the dice and the lamps are based on a modular concept, i.e. additional dice and lamps can be added to the system at any time. The voice synthesiser module offers text-to-speech conversion and helps as an assistant in the game, providing feedback to the user.

Chapter 3

Hardware Design

This chapter describes the hardware components and highlights considerations of the hardware design. First an overview over the whole system is given in section 3.1. The system is based on two modules, the Switch Module and the Voice Module, which both have been built in the course of this project. A specific description of the basic system components follows in section 3.2. In section 3.3 we go more into the peculiarities of the two modules. Finally the chapter concludes with characteristic data of the system gained from power consumption measurements (section 3.4).

3.1 System Overview

The design of the hardware was driven by the choice of the demonstrator. The chosen scenario has been presented in section 2.2. Figure 2.1 has shown the single parts of the system: foam dice, candle lights and the voice synthesiser. The division into these parts given, we decided for an approach based on two different modules:

- The *Switch Module* includes a microcontroller and a radio transceiver, three sensors (accelerometer, pressure and light sensor) and an analog switch. It can be integrated into the foam dice and the candle lights. The architecture of the Switch Module is depicted in Figure 3.1.

- The *Voice Module* is equipped with a microcontroller and a radio transceiver, three sensors (accelerometer, pressure and light sensor), a phone jack and a voice synthesiser. A low dropout (LDO) regulator and a boost converter form the module's power supply. The module provides the circuitry to control a voice synthesiser through wireless access. Either music can be recorded and played, or ASCII strings sent which will be synthesised. Figure 3.2 shows the architecture of the Voice Module.

Although the two modules are developed having the demonstrator in mind, they are useful beyond this specific application and can be applied in general wherever a networked embedded device is required.

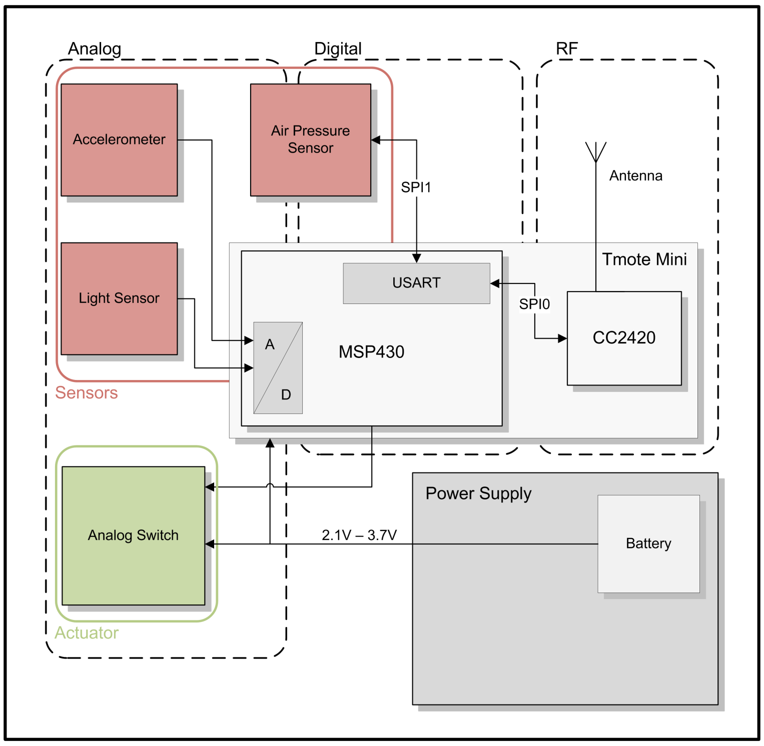


Figure 3.1: System overview of the Switch Module.

3.2 System Components

3.2.1 Tmote Mini Module

The Tmote Mini from Sentilla[1] is the follow-up model of the Tmote Sky motes and with a size of only 2.54cm × 2cm (miniSDIO^{TM} format) the last generation of WSN hardware. Tmote Mini has been chosen for its small form factor and its support of TinyOS[2] (see section 5.1). It is compatible to the standard WSN platforms TelosB[3] and Tmote Sky, thus reducing the overhead of software platform adaptation.

Tmote Mini's most important components are the Texas Instruments MSP430F1611 microcontroller and the Chipcon CC2420 low-power radio:

- *MSP430 Microcontroller* MSP430 is one of the most commonly used microcontrollers in WSN research (used on different platforms, such as Tmote Sky or Tmote Mini, TelosB and eyesIFX[4], and by different groups [16, 21, 30]).

[1]Moteiv is now Sentilla. Tmote Sky and Tmote Mini WSN hardware was developed and sold by Moteiv (http://www.moteiv.com). Sentilla (http://www.sentilla.com) has taken over Moteiv. Since the beginning of 2008 Sentilla has been launching new WSN hardware products which run their own application framework based on Java.

[2]http://www.tinyos.net

[3]http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

[4]http://www.ebv.com/media.php/EBV/Products/Brochures/infineon_eyesifx_flyer_20_02_06.pdf?dl=1

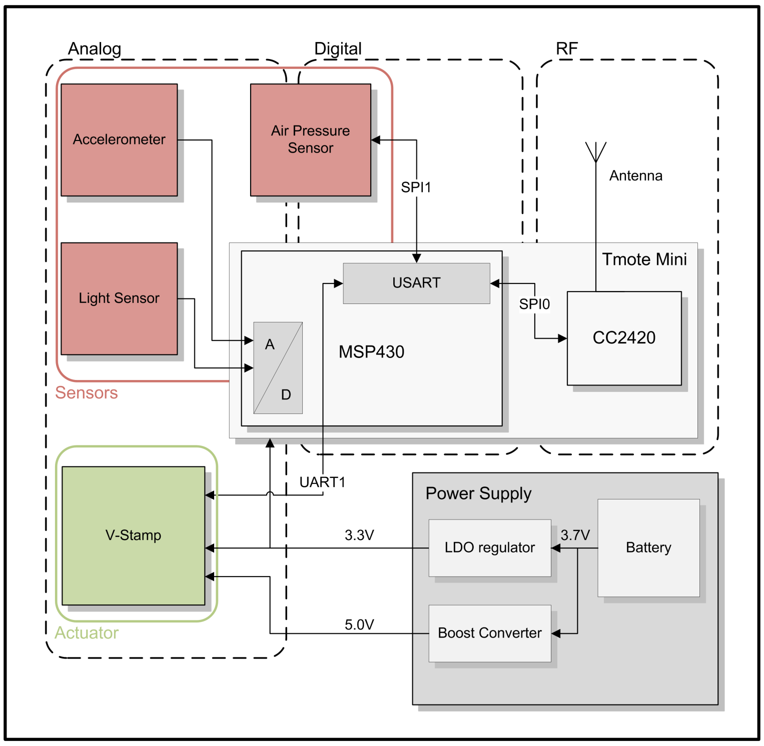


Figure 3.2: System overview of the Voice Module.

TinyOS supports the Tmote Sky and Tmote Mini platforms, i.e. MSP430 runs TinyOS. The Switch Module as well as the Voice Module are operated by the MSP430F1611[5].
MSP430F1611 is a ultralow-power microcontroller featuring a 16-bit RISC CPU, operating at a maximum of 8MHz and providing memory of 10kB RAM and 48kB Flash. It offers two built-in 16-bit timers, a fast 12-bit A/D converter, a 12-bit D/A converter, DMA, and 48 I/O pins. Multiple interfaces are included: 2 USART modules, where USART0 can be set to UART-, SPI- or I2C-mode and USART1 to UART- or SPI-mode, additionally 8 ADC signal input and 2 DAC output channels.

- *CC2420 Radio* The wireless transceiver CC2420[6] implements the ZigBee MAC layer, i.e. the PHY and MAC layers (IEEE 802.15.4 protocol[7]). CC2420 is a low-power solution for robust wireless communication in the 2.4GHz unlicensed ISM band offering an effective data rate of 250kbps. According to data sheets, CC2420 allows for data exchange over short distances, i.e. up to 30m indoors and more than 100m outdoors. First tests with the fully equipped

[5]http://focus.ti.com/docs/prod/folders/print/msp430f1611.html
[6]http://focus.ti.com/docs/prod/folders/print/cc2420.html
[7]http://grouper.ieee.org/groups/802/15/pub/TG4.html

Switch Module yielded maximum distances between 25m and 60m indoors and confirmed the specifications given in the data sheets.

The antenna design is very important in order to achieve a useful communication range. Two options are possible: a PCB or a chip antenna. We decided for a 2.4GHz chip antenna due to its much smaller size. Moreover, an antenna connection with implemented balun network is already provided by the Tmote Mini and a chip antennna was also used successfully in a previous project [31].
A proper layout is crucial to ensure optimal operation of the antenna. The area next to the antenna has to be free of components or traces and no ground plane or traces must lie on the opposite side of the PCB. The feed trace from the RF stage to the antenna should be kept as short as possible. In addition, we provided RF GND planes on the mid and bottom layer of the PCB for counterpoise (cf. Appendix A.3).

Tmote Mini connects CC2420 to the MSP430 through the SPI-interface of USART0 (see Table 3.1), i.e. USART0 of the Tmote Mini module is shared with the radio transceiver.

The Switch Module and the Voice Module, both use the USART interface for the pressure sensor. The voice synthesiser on the Voice Module also needs the USART for communication. Having not more than two USART modules at hand, we decided for a solution in which the actuator and the sensor share USART1 whereas CC2420 radio owns USART0 exclusively. Hence the most important and most used component is not restricted. Actuators and sensors need arbitration but this restriction is relatively small because the voice synthesiser is used only sporadically.

CC2420	Tmote Mini	MSP430	Dir	Description
SI	P3.1/SIMO0	29	O	USART0 SPI SIMO
SO	P3.2/SOMI0	30	I	USART0 SPI SOMI
SCLK	P3.3/UCLK0	31	O	USART0 SPI Clock

Table 3.1: Pin assignment and connections to CC2420 radio transceiver. Here only the pins concerning the SPI-interface between MSP430 and CC2420 are listed. “Dir” indicates the direction in relation to Tmote Mini or MSP430, respectively.

Tmote Mini contains an EPSON quartz which provides MSP430 with an external clock signal of 32.768kHz. This clock signal can be output from Tmote Mini by passing it to the general purpose digital input/output pin (GPIO) P2.0 and selecting the pin’s module function. This way, system components which require an external clock generation can be timed properly without the use of an additional oscillator.

3.2.2 Sensors

Both modules are equipped with three different sensors: an accelerometer, a pressure sensor and a light sensor.

- The *accelerometer* enables the detection of movement and orientation (in terms of where earth gravity pulls towards).
- The *pressure sensor* allows reasoning about the rough altitude. Measuring the barometric pressure the altitude is detected and the object can sense a change in height.
- The *light sensor* collects information about the environmental state. As one actuator of the demonstrator was given to be a lamp, its counterpart, an ambient light sensor is chosen.

Different sensor solutions have been evaluated for each sensor and the chosen sensors are presented briefly in the following subsections. During evaluation focus was put on the power consumption of the devices. Supply currents should be low to guarantee maximum runtime and supply voltage was considered to be around 3V as the sensors are powered directly by the microcontroller (see also section 3.2.5). Furthermore, devices were chosen which require only few external components to save PCB space. Besides power and size, further criteria have been the sensors' range and sensitivity and - with respect to soldering - footprint and package types.

3.2.2.1 ADXL330 Acceleration Sensor

The ADXL330 [8] is a small 3-axis MEMS accelerometer from Analog Devices. Each of the three outputs of the ADXL330 is sampled by an analog connection to one of the ADC inputs of the Tmote Mini.

Implementation of the accelerometer driver in TinyOS followed the guidelines provided by TEP 109[9]. The driver is structured into a wrapper component and a hardware interface layer beneath. The three axes are sampled sequentially. Each time the value for the last axis has been read an event indicating the completion of the sampling process is signaled.

Acceleration sensors with sampling frequencies of 50Hz up to 500Hz were successfully applied for activity recognition [32, 24, 33, 34]. ADXL330 has a typical bandwidth of more than 500Hz. The filter of the accelerometer is optimised by adjusting the capacitances C_X, C_Y, C_Z (see Figure 3.3). The capacitors implement low-pass filtering, i.e. there is a trade-off between required minimum bandwidth, resolution and noise reduction. As no additional filtering of accelerometer raw data is applied for noise reduction, we optimised the accelerometer circuit for a maximum of 50Hz sampling frequency.

[8] http://www.analog.com/UploadedFiles/Data_Sheets/ADXL330.pdf

[9] see TEP 109 on http://www.tinyos.net

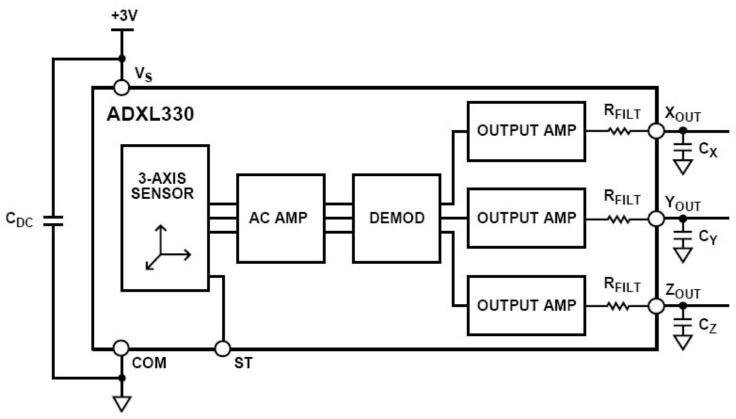


Figure 3.3: Functional block diagram of the ADXL330. The accelerometer is decoupled from noise on the power supply by capacitor C_{DC}. Capacitors C_X, C_Y, C_Z provide for filtering the output pins X_{OUT}, Y_{OUT}, Z_{OUT}.

One restriction of the ADXL330 is the measurement range of about $\pm 3.6g$. If objects are shaken, the acceleration exceeds the accelerometer's maximum rating and can become as high as 12g by human movement only [24]. As a consequence, the accelerometer saturates. As long as no precise acceleration values are required or only a binary decision is to be made about the occurrence of any movement, that is not of major importance. But as soon as correct data are needed, e.g. for the calculation of features, such as signal frequency or maximum values, saturation causes distortion and can result in losses of recognition.
A second downside is ADXL330's small chip scale package which is difficult to solder by hand.

3.2.2.2 MS5540B Air Pressure Sensor

The MS5540B [10] is a SMD-hybrid device working as a miniature barometer/altimeter module. It includes a piezoresistive pressure sensor, a temperature detection and a 15-bit ADC. MS5540B is a low power, low voltage device with automatic power down switching, designed for microcontroller applications. To use a pressure sensor as an altimeter the pressure range should go from a few millibar up to about 1bar. There are differential and absolute sensors available. For measuring the altitude an absolute sensor, such as the MS5540B, is the right choice.
Pressure sensors are usually rather large (diameter of 1.5cm or more). Here MS5540B offers a good solution with a dimension of 6.2mm $\times$ 6.4 mm only.

Pressure sensors often have a digital interface. MS5540B comes with a 3-wire serial interface and an additional 32.768kHz system clock line. The system clock

[10] http://www.intersema.com/site/technical/ms5540.php

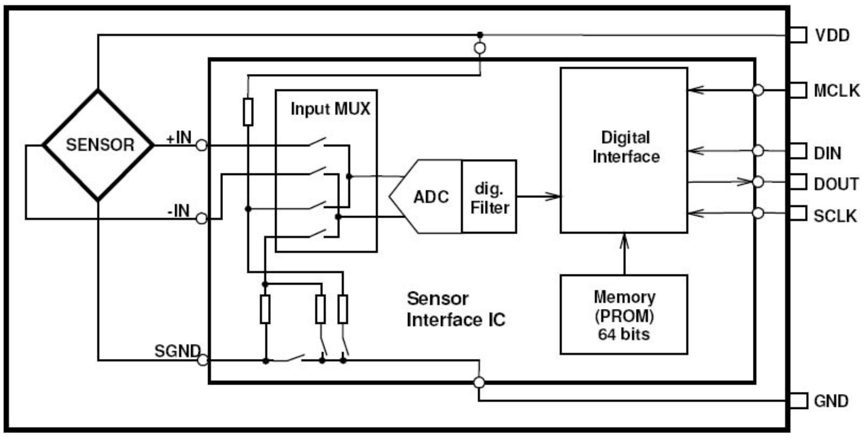


Figure 3.4: Functional block diagram of the MS5540B. Besides the actual senor the pressure sensor consists of an ADC with digital filter, a PROM to store factory calibration data and a unit providing the digital interface.

is provided by reading out the clock signal from the EPSON quartz internal to the Tmote Mini (as described in section 3.2.1 further above). The pressure sensor is connected to the Tmote Mini using the USART1 module in SPI-mode. An overview of pin assignments is given in Table 3.2 and Figure 3.4 shows the overall structure of the MS5540B.

MS5540B	Tmote Mini	MSP430	Dir	Description
DIN	P5.1/SIMO1	45	O	USART1 SPI SIMO
DOUT	P5.2/SOMI1	46	I	USART1 SPI SOMI
SCLK	P5.3/UCLK1	47	O	USART1 SPI Clock
MCL	P2.0/ACLK	20	O	ACLK output (32.768kHz)

Table 3.2: Pin assignment and connections to MS5540B. Here only the pins concerning the SPI-interface between MSP430 and MS5540B are listed. "Dir" indicates the direction in relation to Tmote Mini or MSP430, respectively.

A downside of the MS5540B is the non-standard SPI interface. The manufacturer designed an own communication protocol based on bit sequences of different lengths. Using a SPI configuration with a constant character length (e.g. 8 bit), with a particular clock phase and clock polarity one runs into troubles since the character lengths and bit sequences do not match. Additionally, the clock polarity changes within a communication sequence. A detailed description of the interface can be found in the device's data sheet.

MS5540B can be adapted well to particular applications because the sensor is compensated in software. In our present version of the pressure sensor driver the procedure for sensor calibration and proper scaling is implemented but not fully tested. For the use in the scenario calibration and temperature compensation is not absolutely necessary. When moving an object not primarily the exact pressure value or altitude is of interest but more the *change* in height, i.e. the immediate pressure difference. Figure 3.5 shows sample data recorded by the MS5540B. Although the signal drifts over time and is rather noisy changes of about 0.8m to 1m in height can be detected. The change in height was recognised especially well when the module with the pressure sensor was raised rapidly.

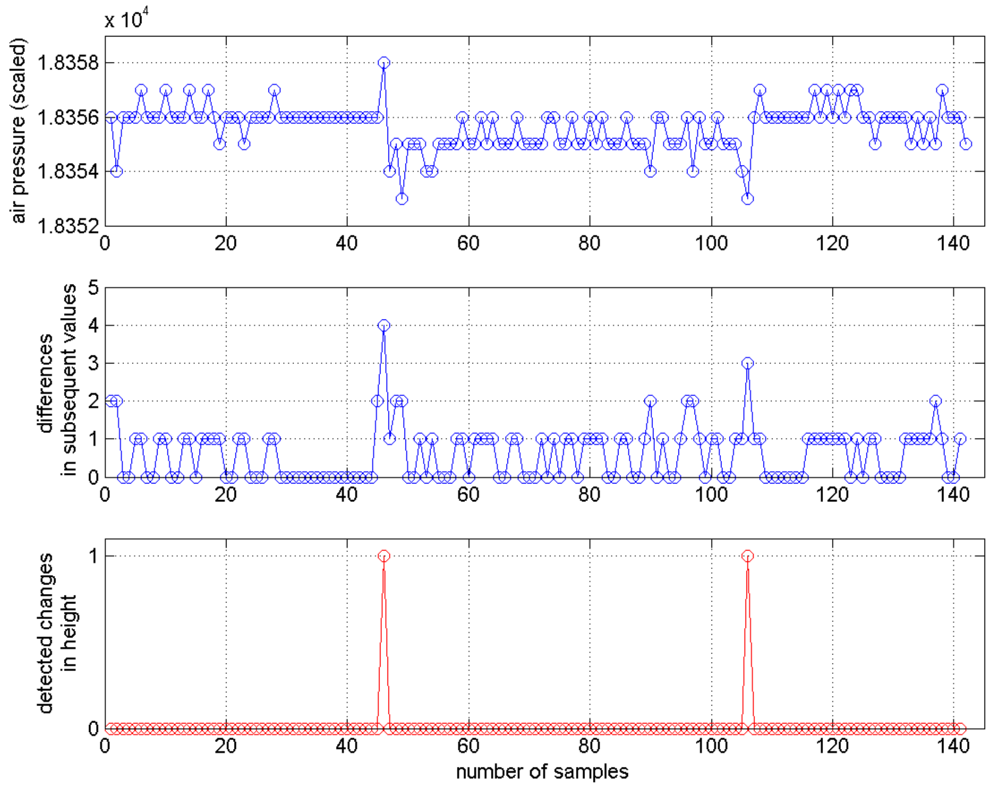


Figure 3.5: Air pressure sensor readings (sensor sampled at 10Hz). The differences in the input signal are calculated and the changes in height estimated by thresholding appropriately. The two detected changes correspond to a positive and a negative change in altitude of about 0.8m.

3.2.2.3 APDS-9003 Light Sensor

The APDS-9003[11] is a low-cost ambient light sensor which provides a spectral response characteristic close to those of human eyes.

We initially selected another light sensor. SFH 5711[12] is a high accuracy ambient light sensor from OSRAM which consists of a photo diode and an IC. It would have offered additional amplification and a logarithmic conversion of the photo diode output signal such that resolution at low brightness levels is improved. But we had to resort to the APDS-9003 because SFH 5711 was not available due to logistic reasons.

There are different types of ambient light sensors which vary in their complexity and quality. The simplest are photo resistors, next step in complexity are photo diodes offering high performance at a rather large size or photo transistors including amplification at a small size, followed by entire modules containing several components.

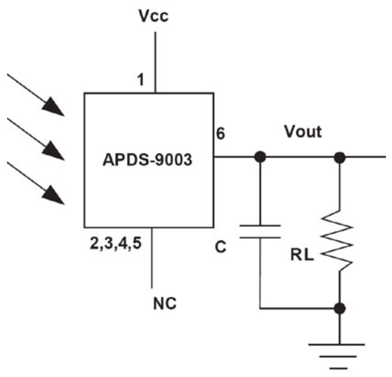


Figure 3.6: Functional block diagram of the APDS-9003. The ambient light sensor basically works as a photo diode. Voltage drop at resistor R_L is input into the microcontroller, capacitor C acts as a low-pass filter.

The APDS-9003 sensor module is sketched in Figure 3.6. It mainly comprises a photo diode. APDS-9003 yields an analog output current which is transformed into voltage V_{out} by the load resistor R_L. V_{out} is used as input signal into one of the Tmote Mini ADC pins. An additional capacitor C adjusts the reaction time of the sensor and reduces noise of incoming light.

An important parameter is the absorbed radiant power per area, so called illuminance. Illuminance is a measure of the intensity of the incident light. Its unit is Lux ($1lx = 1lm/m^2$). The range of illuminance that can be detected increases with complexity of the light sensor. APDS-9003 is designed for a maximum of about

[11]http://www.avagotech.com/products/ir_sensors/ambient_light_photo_sensors/apds-9003
[12]http://catalog.osram-os.com/catalogue/catalogue.do?favOid=0000000300037a3100ad0023&act=showBookmark

1000lx, i.e. bright indoor light can be detected without reaching saturation. By testing the APDS-9003 indoors, several brightness levels could be distinguished. The relation between illuminance and output current (or output voltage) is logarithmic. As APDS-9003 does not provide logarithmic conversion the ADC input was divided into exponential bins to reach optimal resolution. For low illuminance small differences in the input values already indicate a significant change in brightness while these differences must be considerably higher at high brightness levels.

3.2.3 Actuators

3.2.3.1 DG2016 Dual SPDT Analog Switch

The DG2016[13] is a CMOS dual single-pole/double-throw (SPDT) analog switch. It can turn on and off a connection and with this control an actuator. The SPDT is bidirectional, i.e. it conducts in both directions equally well and supports a continuous current of about $\pm 50mA$ and a peak current of up to $\pm 200mA$. When turned off it can block voltages up to the power supply level. It operates at a supply voltage in the range of 1.8V to 5.5V and has reduced power consumption with a supply current of typically 0.01μA.
Figure 3.7 shows the functional block diagram of the DG2016. It includes two analog switches which can be toggled by applying a high/low voltage level to the input pins. The switch provides fast switching speeds with turn-on times of about 30ns.

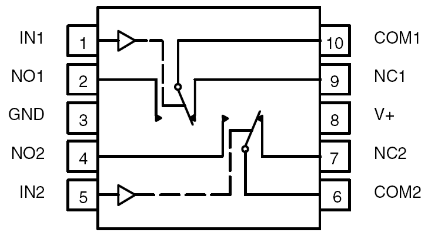


Figure 3.7: Functional block diagram of the dual SPDT analog switch. At logic input 0 COM1 and COM2 are connected to NC1 and NC2. At logic level 1 the switch toggles, COM1 and COM2 are connected to NO1 and NO2.

3.2.3.2 V-Stamp Voice Synthesiser Module

V-Stamp voice synthesiser module[14] from RC Systems features an integrated text-to-speech processor, a tone generator and the recording, downloading and playing of sound files.

[13]http://www.vishay.com/analog-switches/list/product-72030
[14]http://www.rcsys.com/Downloads/v-stamp.pdf

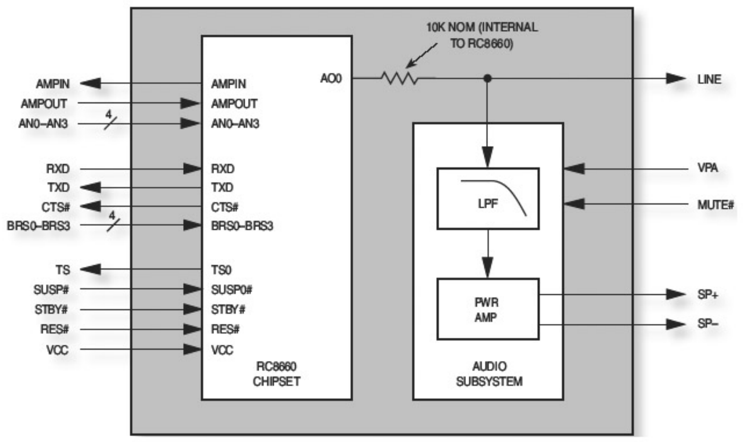


Figure 3.8: Functional block diagram of the V-Stamp. The text-to-speech synthesiser is divided into a digital and an audio subsystem. The audio subsystem adds a low-pass filter and an audio power amplifier and thus makes the system completely functional.

Figure 3.8 presents a functional description of the system: it is composed of an audio subsystem and a digital subsystem. The digital subsystem is based on RC's text-to-speech processor chipset[15]:

- V-Stamp converts ASCII text into speech automatically and is mainly designed for English text. It also operates in phoneme mode, i.e. it allows for modification of single phonemes within words, which could be useful when dealing with other languages. A lot of commands are provided to control different aspects of speech, such as speed, volume or voice type.

- An onboard nonvolatile memory is included. So up to 33 minutes of recorded messages and sound files can be stored in the V-Stamp.

- V-Stamp offers a musical tone generator that can generate three tones simultaneously over a four-octave range. For signaling applications V-Stamp's sinusoidal tone generator can be used.

V-Stamp is available together with a development kit[16]: a docking board includes a RS-232 serial interface to connect the V-Stamp modules with the PC. Sound files can easily be downloaded to the modules using the application programs RCStudio and RCLink from RC System. The stored sound files get an index or tag which can be used afterwards to access the files by calling the "Play Sound File" command. A more detailed description how to program the V-Stamp modules

[15]http://www.rcsys.com/Downloads/rc8660.pdf
[16]http://www.rcsys.com/Downloads/v-pod.pdf

from the PC is given in Appendix C.

The Voice Module connects to the V-Stamp through a standard asynchronous serial interface (UART). UART interface is provided by Tmote Mini's USART1 module set to UART-mode. Table 3.3 shows the ports used with respect to the UART interface. For interfacing the digital subsystem the baud rate of the serial port must be set. It can be selected either by auto-detection, i.e. baud rate is automatically detected from input data or by setting pins' logic levels directly in the hardware design. In order that the Voice Module accepts both methods a set of 0Ω resistors are arranged in the hardware design. Baud rate of the present system is adjusted to a typical value of 9600bps. The serial interface operates with 8 data bits (LSB first), no-parity and 1 stop bit.
The driver implemented for the V-Stamp module provides an interface to set several parameters of V-Stamp's extensive parameter set, to invoke text-to-speech conversion by passing strings and to play sound files previously loaded into V-Stamp's memory.

V-Stamp	Tmote Mini	MSP430	Dir	Description
RXD	P3.6/UTXD1	34	O	USART1 UART Transmit
TXD	P3.7/URXD1	35	I	USART1 UART Receive
TS	P2.7	27	I	GPIO[1]
SUSP♯	P2.1	21	O	GPIO
CTS♯	P2.6	26	I	GPIO[1]
STBY♯	P2.3	23	O	GPIO
RES♯	P1.2	14	O	GPIO

[1] Port 2 offers interrupt capability

Table 3.3: Pin assignment and connections to V-Stamp. Here only the pins concerning the interface between MSP430 and V-Stamp's digital subsystem are listed. "Dir" indicates the direction in relation to Tmote Mini or MSP430, respectively. General purpose digital input/output pins (GPIO) of port 2 (and port 1) accept interrupts. Especially for the input pins Talk Status (TS) and Clear To Send (CTS) interrupt capability is a useful feature.

V-Stamp's digital subsystem and audio subsystem work independently of each other. Each system can be powered separately, which is beneficial. The digital part operates at 3.3V. Connecting to a lower voltage (than 5V) helps minimising digital power drain and results in relatively low supply currents. The opposite holds for the audio subsystem: supply voltages of 5V provide for maximum power output. Volume can be at high settings while keeping distortion low at the same time.
The audio subsystem can directly drive an 8 ohm speaker. Power output is increased

as the speaker drives the load differentially.
Audio subsystem operates at supply currents as high as 300mA. To reduce power consumption the audio subsystem can be shut down when the V-Stamp is not in use. Thus supply current is reduced to about 1μA. Shutdown of audio subsystem does not have to be executed externally by a microcontroller but can be assigned to V-Stamp directly by connecting the MUTE$\sharp$ pin to the TS pin. This solution is simple and effective, so it was implemented in the Voice Module.

3.2.4 JTAG Connector

The idea was to be compatible to the SensorButton [35] interface. Both modules are connected to the JTAG programmer using a 10 pin Harwin connector mounted on the PCB and the JTAG adapter designed by Marco Graf for the SensorButton. The JTAG adapter enables to load compiled code onto the MSP430 on the modules (cf. chapter 5 and Appendix C). The JTAG adapter board must be connected to the Switch Module and the Voice Module as shown in Figure 3.9.

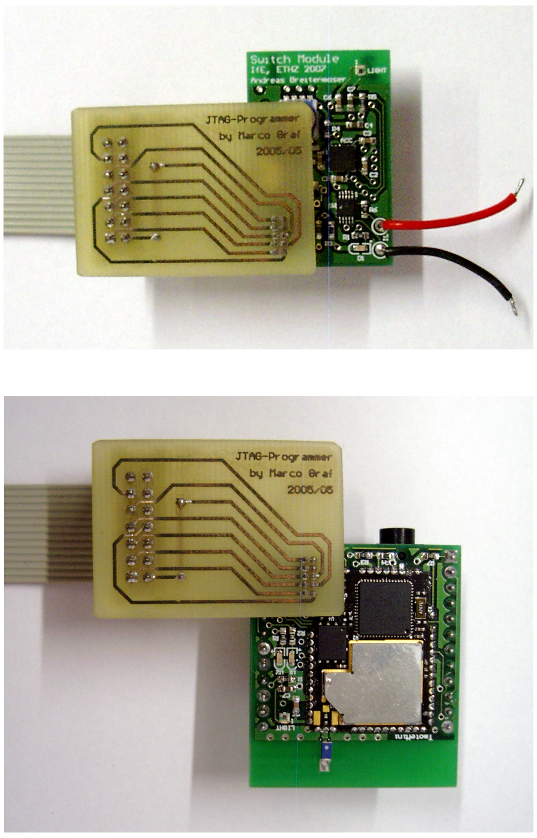


Figure 3.9: Top: Connecting the JTAG adapter to the Switch Module. Bottom: Connecting the JTAG adapter to the Voice Module.

3.2.5 Power Supply

3.2.5.1 Low Power Mode and Power Switch

Tmote Mini operates at voltages from 2.1V to 3.7V and offering a complete WSN solution, it consumes not more than a current of 1μA in sleeping mode, 2mA when active and up to 20mA during USART communication.

The sensors used for the two modules were not directly connected to the power supply but supplied with the operating voltage over the Tmote Mini module. In connecting the power line to the general purpose digital input/output pins (GPIO) of the Tmote Mini each sensor can be individually powered up or shut down. There is no need for a separate sensor power switch on the board.

A voltage divider is connected to one of the ADC-pins of the Tmote Mini allowing for monitoring of the battery voltage. WSN are supposed to work autonomously and are constrained in available power. It is the detection of battery running low that makes an appropriate reaction of the system possible.

3.2.5.2 XC6217 Low Dropout Regulator

The XC6217 [17] is a low dropout (LDO) regulator in a SOT-25 package. LDO's advantages are low cost, low size and low noise. The LDO regulator of XC6217 series represents a relatively simple converter structure that offers quite a low dropout voltage especially for output voltages above 3V, thus reduces power dissipation and increases efficiency.

3.2.5.3 TPS61202 Boost Converter

The boost converter TPS61202 [18] from TI is especially designed for battery powered and portable products offering a large input range (which is beneficial as power of batteries is decreasing). It can provide up to 600mA output current at an output voltage of 5V. TPS61202 has a wide input range from 0.3V to 5.5V, a quiescent current below 55μA and offers an efficiency around 90% when operated at V_{in} = 3.7V, I_{out} = 300mA and V_{out} = 5V, which meets our system specifictions quite well.

3.2.5.4 Rechargeable Li-ion battery

Different rechargeable Li-ion batteries from Varta[19] were evaluated. The criteria were size and weight, capacity and supply voltage. PLF443441C and LIP103450RC were found to be suitable for devices with either lower or higher power consumption requirements. Both provide 3.7V nominal voltage and measure 3.4cm $\times$ 4.1cm or 4.9cm, respectively. PLF443441C weighs 12.5g and offers 610mAh. LIP103450RC is bigger, has a weight of 40g and a capacity of 1950mAh.

[17]http://www.torex-europe.com/products/ranges/233
[18]http://focus.ti.com/docs/prod/folders/print/tps61202.html
[19]http://www.varta-microbattery.com/en/oempages/product_data/rechargeable.php

3.3 Peculiarities of the Modules

Although two different modules were chosen to realise the full functionality required by the scenario, similarities were spotted and the hardware is designed such that central parts of the two modules, such as the microprocessor and the radio unit as well as the sensors, are identical. A uniform design facilitates the design process and limits the range of components to order.

During the design process of the modules we kept an eye on the special requirements of WSN hardware. The costs and the size of the single nodes of a network should be kept low. Additionally, available RAM is highly limited. Besides memory, one of the strongest constraints is the low power consumption requirement. These resource constraints call for design trade-offs and limit the complexity of the algorithms executed on the motes. Power management or power saving capabilities which allow for switching to sleeping mode are important.

Since the modules were designed to build smart objects, integration of the hardware into objects and the modules' form factor were major issues. Especially the board size of the Switch Module was constrained by the little space left inside the candle lights (see Figure 3.10).

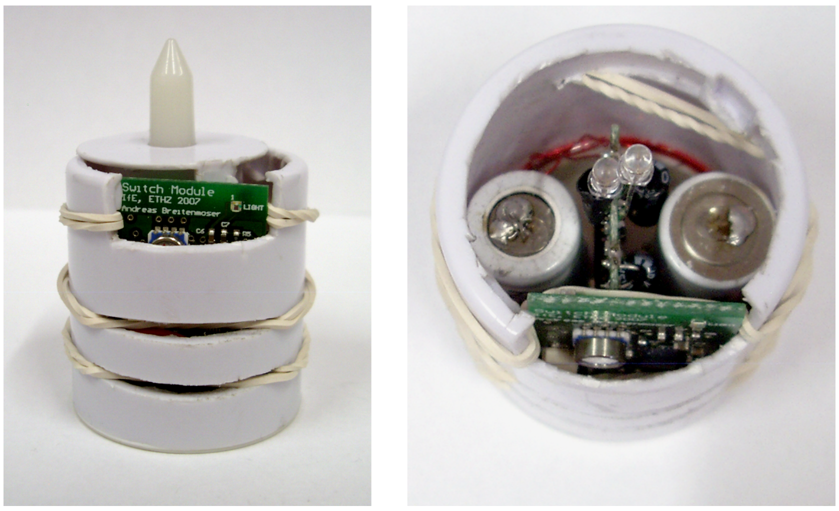


Figure 3.10: Left: Side view of the candle light with the Switch Module integrated (the glass of the lamp was previously removed). Right: Top view of the candle light. Space inside the candle lights is limited: two rechargeable batteries are connected in series, the circuit for charging also provides the LED lamps, induction wires (red) form a loop at the bottom for energy transfer and the Switch Module senses, communicates and controls the lamps.

The connection between the V-Stamp and the Voice Module requires two header sockets. To get a compact design, audio and power connectors were packed in-between such that spare space underneath the voice synthesiser was used optimally (see Figure 3.13). The PCB of the Switch module was downscaled to 2.3cm × 3.5cm. For the Switch Module minimum size was given by the Tmote Mini (width of 2cm). The Voice Module was implemented on a PCB 3.43cm × 4cm in size. Here minimum size was limited by the V-Stamp (width of 3.43cm). Pictures of the final boards are provided by Figures 3.11 and 3.12 below.

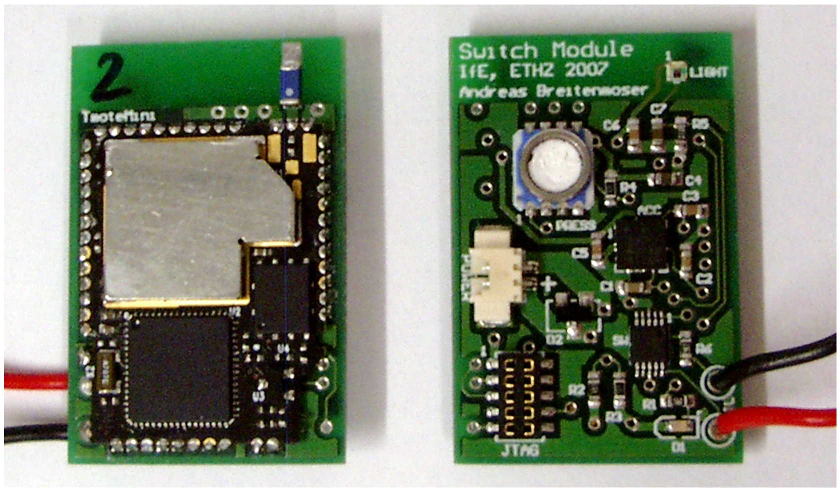


Figure 3.11: Final PCB of the Switch Module. Left: Bottom view. Right: Top view.

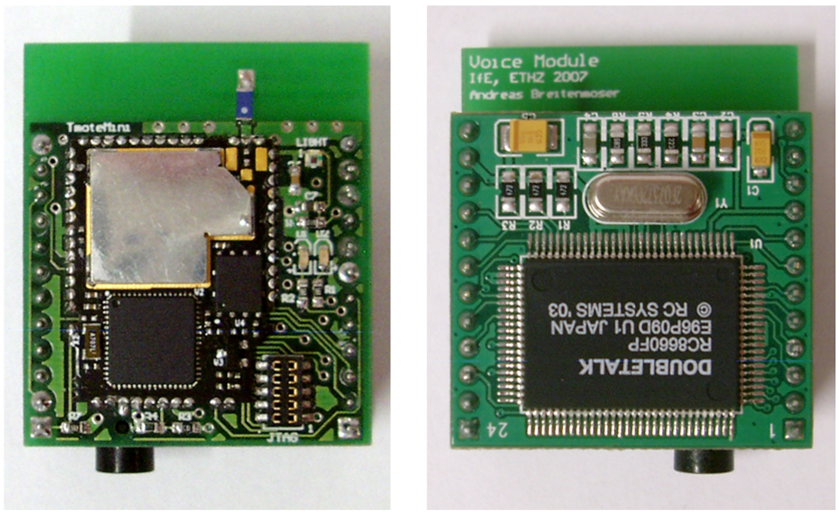


Figure 3.12: Final PCB of the Voice Module. Left: Bottom view. Right: Top view with the V-Stamp voice synthesiser plugged in.

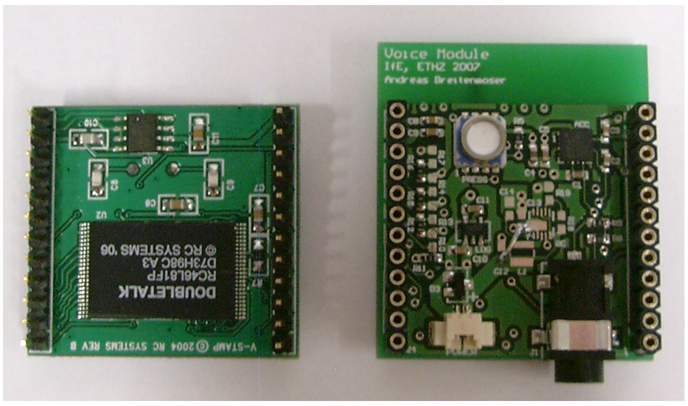


Figure 3.13: Inside of the Voice Module. Left: Bottom view of the V-Stamp voice synthesiser. Right: Top view of the Voice Module without the V-Stamp voice synthesiser plugged in.

The modules are integrated into every-day objects which do not necessarily contain any power supply. Independence of external supplies is reached by powering the modules from Li-ion batteries. Besides batteries' lifetime, again, size was an important factor. Batteries were chosen, of which sizes get as close as possible to the board sizes mentioned above.

For both modules, the functionality is tested by running specific test programs on each of the equipped motes. The test protocols for the Switch and Voice Module can be found in Appendix A.

3.3.1 Switch Module

The Switch Module meets two purposes within the demonstrator. If the Switch Module is fully equipped it can be built in the candle lights (Figure 3.10) where it serves as an actuator and provides full functionality of its sensors. The analog switch is driven by the microcontroller and turns the LED-lamps of the candle on and off. Pressure and light sensors provide further sensing.
To recognise actions performed with the foam dice, basically it is sufficient to have the accelerometer, the Tmote Mini, power and JTAG connectors on the PCB. This can be provided in equipping the Switch Module only partly. The microcontroller samples the accelerometer and computes the acceleration magnitudes which are broadcasted by the CC2420 wireless transceiver. Thus shaking can be detected and grouping of the dice can be carried out.

Imageo Candle Light from Philips is used as the light source in our scenario (see Figure 2.1). These lamps can be turned on and off without an external on/off switch by simply rotating them. As there is no schematic of the candle lights available the system and its operation must have been analysed. A look into the candles showed that they consist of two LEDs which are controlled by a ball switch and powered by two rechargeable batteries. The batteries are connected in series and provide a total voltage of 2.7V at an electric charge of 600mAh, which allows for an operating time of approximately 20 hours. The candle lights come with a induction base which makes wireless recharging possible. The socket of the lamp encloses a loop of wire and energy is transferred by electromagnetic fields inducing a current.

The Switch Module includes a CMOS dual single-pole/double-throw (SPDT) analog switch. The switch is bidirectional and is intended to emulate the ball switch functionality.
The applicability of this replacement was investigated by a test set-up. By turning the ball switch on and off again the voltage along the switch changes from -0.6V to +0.9V. Voltage returns to -0.6V with a further switching on/off sequence. During two succeeding on/off sequences current through the switch changes direction. Therefore an analog switch that operates in both directions is needed. The lamp reacts each time current flows, i.e. on each rising edge of analog switch's logic input only. If the ball switch is closed and opened again the lamp switches on or off, i.e. the frequency of switching for the ball switch or analog switch, respectively, is twice the switching frequency of the lamp. Further it seems as if a capacity is charged while the voltage along the switch remains at one of the both levels. The lamp does not turn on or off if the switch is clocked too fast, which results in a limitation of the lamp's switching frequency at a maximum of about 0.25Hz.

As the candle lights already include two batteries, the Switch Module is supplied from these batteries, too. Instead of connecting a converter in between we decided on powering the Switch Module directly from the batteries. This way, space and components could be saved. On the other hand, voltage and current is decreasing over operation time. Therefore components with relatively wide input ranges are used. The lowest battery voltage at which the candle lights are still on was measured to be about 2.3V. To detect low voltages supply voltage can be monitored by Tmote Mini over the ADC (cf. section 3.2.5).

When the Switch Module is built in the foam dice it can be connected to a battery providing 3.7V directly because input voltages of Tmote Mini and ADXL330 of about 3.7V are maximum allowable. The wide voltage range of all components offers to use the Switch Module without voltage regulator.

3.3.2 Voice Module

A voice synthesiser module allowing text-to-speech conversion was selected as a second actuator. The idea behind the Voice Module is to make the whole system more interactive. Messages with control parameters are received by the CC2420 transceiver and passed on to the V-Stamp voice synthesiser module. Music can be played as soon as an event is set off or comments on the user's present movement can be given. The voice module is also equipped with the three sensors such that it provides for similar actions as the Switch Mode.

Rechargeable batteries with sizes in the range of the Voice Module typically provide voltages around 3.7V. The 3.3V supply voltage of the digital subsystem is set by a LDO regulator.

The audio subsystem is envisioned to be supplied with 5V using a low input voltage step-up converter. The audio subsystem requires a rather high supply current (about 300mA) when active. This imposes a strong restriction. The boost converter TPS61202 from TI meets the requirements by providing output currents up to 600mA.

Layout design for converters operating at high switching frequencies is an especially delicate step. While implementing the converter layout we sticked to the layout specifications at the best:

- The main current path and the power ground tracks use wide and short traces.
- The input and output capacitors as well as the inductor are placed as close as possible to the converter.
- The ground mid layer contains a power ground (PGND) “island” which connects to signal ground (GND) at only one point. This way, GND and PGND are separated to minimise ground noise.

In the implementation, the LDO regulator operates correctly but the boost converter does not. Despite intensive debugging it could not be made working properly. Following steps were taken one after the other:

1. First no voltage was seen from the output. But no shorts could be found. Variations of the load did not yield any change either. After slightly increasing the input voltage the converter started to operate yielding a fluctuating output signal. The inductor was charging/discharging and the IC was controlling, hence the converter was not damaged during soldering.
2. The voltage divider at the input port of the converter (UVLO pin) was designed to shut down the main output voltage if the supply voltage falls below 2.75V. The voltage at the UVLO pin must be greater than 250mV to maintain operation. This voltage was hardly reached so the voltage divider was omitted. Output was now present at 3.7V input voltage but it fluctuated between 1.6V and 5.6V showing a characteristic of a charching/discharging-curve.
3. Subsequently the capacitance at VAUX pin, the input and output capacitances were enlarged. The input and output capacitances were doubled by connecting a similar capacitor in parallel. As expected, this resulted in a reduction of output signal’s frequency by half and additional smoothing of the input and output voltage ripples.

4. As a last step, the layout was inspected. The noise generated by the IC indicated a problem associated with the connection between signal and power ground. Through-connections were added to shorten the connections. Noise could be reduced but no change in the basic characteristic of the converter was achieved.

Since the malfunction of the boost converter could not be solved an alternative solution had to be found. Although V-Stamp's audio subsystem is optimised for 5V-operation it can also be powered by voltages lower than 5V. Therefore the boost converter is bridged and the input of the audio subsystem is directly connected to the battery (used batteries provided 3.7V, i.e. 3.7V instead of 5V were applied to the audio subsystem). The disadvantage is a loss in volume and quality of sound.

3.4 Power Consumption Measurements

As power is one of the most limited resources in WSN devices the analysis of the power consumption is important. For the power consumption measurements a test set-up was used as shown in Figure 3.14. A resistor R is connected to the negative supply input of the Switch Module and the Voice Module. At the input of the whole set-up a voltage U_1 is applied such that the input voltages U_2 of the modules are set to the required values. The voltages at the input of the set-up U_1, at the resistor U_R and at the input of the modules U_2 are measured. The current through the modules can then be calculated from the potential difference $U_R = U_1 - U_2$ by using Ohm's Law $I = \frac{U_R}{R}$ and the power consumption is obtained by $P = U_2 \cdot I$, where the mean values measured by an oscilloscope are used for calculation.

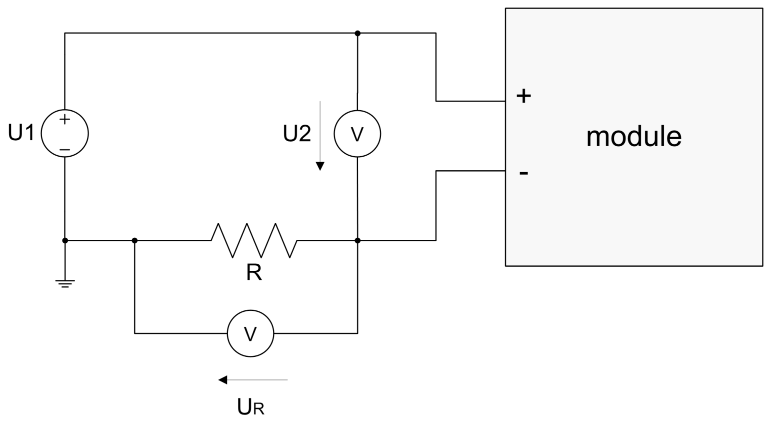


Figure 3.14: Set-up for the power consumption measurements.

The resistance R is maximised to achieve optimal resolution of the measured voltage. Since the input currents for the two modules can vary greatly due to the high supply current of V-Stamp's audio subsystem the test set-up included several resistors with different values which were changed if required. The resistors are connected on the ground (GND) side to reduce noise in the measurement.

3.4.1 Power Consumption of the Switch Module

Switch Module				
Configuration	Measured Values		Referenced Values	
	I [mA]	P [mW]	I_{Ref} [mA]	U_{Ref} [V]
Tmote Mini[1]	1.23	3.32	2	-
LED[2]	8.35	22.55	-	-
CC2420 Sending[3]	18.85	50.90	19 (TX: 17.4)	-
CC2420 Receiving[4]	18.75	50.63	19 (RX: 18.8)	-
Acceleration[5]	1.45	3.92	0.32	3.0
Light[6]	1.58	4.27	2.5	3.0 (at 1kLux)
Pressure[7]	1.68	4.54	0.004 - 1	3.0
all[8]	20.60	55.62	-	-

[1] Only Tmote Mini is active: MSP430 permanently computes squared magnitudes (i.e. multiplications are executed), but CC2420 radio transceiver is deactivated.

[2] Similar to [1] above, additionally the green LED is permanently turned on.

[3] Only Tmote Mini is active: CC2420 radio transceiver is activated, messages are sent at a frequency of 10Hz.

[4] Only Tmote Mini is active: CC2420 radio transceiver is activated, messages are received at a frequency of 10Hz.

[5] Tmote Mini reads acceleration sensor ADXL330 at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[6] Tmote Mini reads light sensor APDS-9003 at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[7] Tmote Mini reads pressure sensor MS5540B at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[8] Everything (except the LED) is turned on: Tmote Mini with CC2420 radio transceiver (sending and receiving at 10Hz), acceleration, light and pressure sensors (all sampling with 10Hz) as well as the analog switch (switching frequency of 0.5Hz) are activated.

Table 3.4: Voltage drop U_R over the resistor R is measured at input voltage $U_2 = 2.7V$. Input currents I and power consumptions P are calculated. Ratings referenced in the data sheets are listed. To compare measured and referenced values the appropriate differences of measured values must be calculated.

The Switch Module operates at a typical input voltage of 2.7V. U_2 was kept at 2.7V and R was set to 1kΩ or 100Ω. The sampling rates of the sensors and the rates of the sending/receiving transceivers were set to 10Hz for the measurements. Table 3.4 lists the results.

The supply current found by the power consumption measurements allows an estimation of the lifetime when the modules are powered from batteries. The Switch Module is mentioned as an example. If the Switch Module is built in the candle light it is powered by a rechargeable battery that provides 600mAh at 2.7V. In the case of full action the Switch Module consumes 20.6mA supply current (cf. Table 3.4). The circuitry of the candle light itself requires about 30mA. The lifetime can be calculated by $T_{lifetime} = Q/I$, which results in an operating time of up to 12 hours. The lifetime can be further increased by duty-cycling.

3.4.2 Power Consumption of the Voice Module

The Voice Module is designed for an input voltage of 3.7V. U_2 was kept at a constant 3.7V. R was set to 1kΩ, 100Ω or 39Ω depending on the expected supply current. The sampling rates of the sensors and the rates of the sending/receiving transceivers were again set to 10Hz for the measurements. The measurement results are presented in Table 3.5.

The audio subsystem of the V-Stamp voice synthesiser was shut down (pin $MUTE\# = 0$) during all measurements except for the power consumption measurements of the V-Stamp itself and of the entire system in action. For the measurement of V-Stamp's power consumption a sine wave of 440Hz was loaded and continuously played. This way fluctuations in the input current (which are especially distinct when playing sound files) could be minimised.

3.5 Conclusions from HW Design

HW design results in two working prototypes. Both, the Switch Module and the Voice Module provide functionalities that are generally useful. They include following main components:

- Tmote Mini module
- Three different sensors: a 3-axis accelerometer, an air pressure sensor and an ambient light sensor
- One actuator on each module: an analog switch (for the Switch Module) and a voice synthesiser (for the Voice Module)

The sensors and actuators can be used for a wide application range. The modules present wireless sensor nodes that allow for further studies in the field of WSN in general and of ambient intelligence in particular. Concrete problems associated with a framework like Titan can be investigated.

An important aspect in WSN is power consumption since systems are supplied from batteries. The modules were specially designed for low power consumption. The V-Stamp's audio subsystem is the component that consumes by far the most power. The measured power consumption is over 650mW. The power needed by the V-Stamp can be reduced by turning it off for most of the time. A minimum of about 5mW is used to drive the Tmote Mini (transceiver deactivated) and the low power sensors. Power consumption measurements proved that the components in the equipped system behave as specified in the data sheets. Our design meets the low power requirements.

Voice Module				
Configuration	Measured Values		Referenced Values	
	I [mA]	P [mW]	I_{Ref} [mA]	U_{Ref} [V]
Tmote Mini[1]	2.10	7.77	2	-
Tmote Mini[2]	3.20	11.84	2 + 0.7 = 2.7	-
LED[3]	17.50	64.75	-	-
CC2420 Sending[4]	20.35	75.30	19 (TX: 17.4)	-
CC2420 Receiving[5]	20.15	74.56	19 (RX: 18.8)	-
Acceleration[6]	1.45	5.37	0.32	3.0
Light[7]	1.13	4.18	2.5	3.0 (at 1kLux)
Pressure[8]	1.28	4.74	0.004 - 1	3.0
V-Stamp[9]	176.92	654.62	240	3.3
all[10]	225.64	834.87	-	-

[1] Only Tmote Mini is active: MSP430 permanently computes squared magnitudes (i.e. multiplications are executed), but CC2420 radio transceiver is deactivated. In addition, the V-Stamp voice synthesiser module is removed.

[2] Similar to [1] above. Now, the V-Stamp voice synthesiser module is plugged back into the two header sockets (for all the following measurements the V-Stamp voice synthesiser module is included on the PCB).

[3] Similar to [2] above, additionally the green LED is permanently turned on.

[4] Only Tmote Mini is active: CC2420 radio transceiver is activated, messages are sent at a frequency of 10Hz.

[5] Only Tmote Mini is active: CC2420 radio transceiver is activated, messages are received at a frequency of 10Hz.

[6] Tmote Mini reads acceleration sensor ADXL330 at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[7] Tmote Mini reads light sensor APDS-9003 at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[8] Tmote Mini reads pressure sensor MS5540B at a sampling frequency of 10Hz. CC2420 radio transceiver and other sensors are deactivated.

[9] Tmote Mini controls the V-Stamp voice synthesiser module. The audio subsystem is activated and a sine wave of 440Hz is played. CC2420 radio transceiver and the sensors are deactivated.

[10] Everything (except the LEDs) is turned on: Tmote Mini with CC2420 radio transceiver (sending and receiving at 10Hz), acceleration and light sensors (both sampling with 10Hz), pressure sensor (just powered) as well as the V-Stamp voice synthesiser module (sine wave tone of 440Hz) are activated.

Table 3.5: Voltage drop U_R over the resistor R is measured at input voltage $U_2 = 3.7V$. Input currents I and power consumptions P are calculated. Ratings referenced in the data sheets are listed. To compare measured and referenced values the appropriate differences of measured values must be calculated.

Chapter 4

Activity recognition

Human interaction with smart objects requires recognition of user activities. A central problem in our application is the robust recognition of movements carried out with simple objects, such as foam dice and candle lights. We focused on recognition of simultaneous shaking of two dice, on the distinction of correlated and uncorrelated shaking more precisely. This can be useful in a broader sense for authentication, initialising and grouping two objects [20, 18, 36]. In WSN, sensor nodes that experience the same movement patterns can be grouped together. Such grouping information indicates which nodes belong together in the near future and allows programs to run separate task networks with their own Network Manager on the nodes of each group.

This chapter describes the algorithms which were analysed and/or applied in the course of the project. Herein, the description follows the perceptual pipeline common in the field of pattern recognition (see Figure 4.1). For analysis of algorithms and selection of promising features measurement data were recorded by experiments. Based on raw data from the 3D-accelerometers, parameters and feature sets were found that perform optimal in recognition of correlated shaking the dice. Results are evaluated in consideration of an embedded implementation, i.e. low computational complexity is targeted: we try to group objects using simple features for classification, meanwhile reducing the need for wireless communication between the objects.

4.1 Algorithms

In the following sections different algorithms are presented. If the algorithms operate on raw data it always concerns data measured by a 3D-accelerometer (either MMA7260Q from Freescale Semiconductor[1] or ADXL330 from Analog Devices).

[1]http://www.freescale.com/files/abstract/event/MMA7260QPK.html?tid=tslp

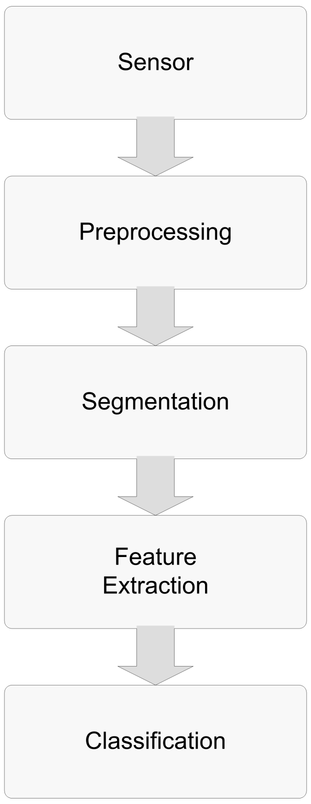


Figure 4.1: Perceptual pipeline: Sensing, preprocessing and segmentation, feature extraction and classification.

4.1.1 Preprocessing and Segmentation

In *preprocessing* steps the raw data is simplified and its quality gets improved. As the accelerometers already provide certain filtering, quality of accelerometer raw data did not need further improvement. Acceleration data coming from different sensor nodes needs preprocessing for alignment of data in time, for normalisation of data detected by different sensors and for handling the different orientations of the sensor nodes:

- *Synchronisation* Time synchronisation is an issue whenever data from different sensor nodes must be compared. Although the same sampling frequency is used on the nodes, delays in processing and communication jitter can occur and data streams may not be synchronised. Whereas delays are largely reduced for serial communication over USB, radio communication can result in significant packet losses or predominant senders. The synchronisation problem can be tackled by synchronisation relying on simultaneous events in the network [37] or by time synchronisation where data gets timestamped.
  To handle unequal sample numbers in acceleration data, the input data was synchronised by linear interpolation. When data was stored on the PC a

timestamp was added to the samples. The data series with more sample points got interpolated, i.e. for the data series that contained fewer sample points new data values were calculated for each time stamp. This way, a single set of time stamps resulted for both data series without introducing artefacts into the raw data.

- *Scaling* Accelerometer data is normalised on 1g gravity: $a_x = (x - x_{offset})/f_{scale}$. Normalisation is needed because sensor devices vary in sensitivity and offset. x_{offset} and f_{scale} were determined by aligning each axis in parallel to earth gravity and measuring the acceleration twice, once for positive and negative orientation of the axis.

- *Magnitude* Measured data should be independent of orientation and alignment of the dice. The X-, Y- and Z-coordinates of the 3D-accelerometer were reduced to one dimension by calculating the magnitude: $\|a\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$. Since the absolute magnitude value is not used only the squared magnitude $\|a\|^2$ was computed saving computational costs of square root calculation.

In a second step the raw data was *segmented* to find shaking patterns. Sliding windows with window sizes over the range of 0.25s, 0.5s, 1s, 2s, 4s and 8s were used. The window segments were not overlapped for the evaluation of the algorithms. Especially when larger windows are applied and edges must be detected the segments can be overlapped to increase the activity detection rate (a common value for overlap is 50%, [24]).
In a next step features were computed for each window based on the signal captured in the window.

4.1.2 Feature Extraction

First, two promising approaches in the detection of joint and separate movement of objects are presented. While one calculates the correlation between the accelerations of two sensor nodes in the time domain, the other is based on the coherence in the frequency domain. Both approaches use constant information from every sensor node for detection. These two approaches were shown to work successful and are used in this project as benchmark on what performances could maximally be reached. Next, our approach is introduced, which attempts to retrieve the same information out of simple features computed locally on each single node reducing the need for communication among the motes.

4.1.2.1 Correlation and Coherence

R. and M. Marin-Perianu et al. describe in [21] a fast incremental correlation algorithm for resource constrained devices. Sensor nodes with accelerometers were attached to transported objects as well as on bicycles (experience higher frequencies) and it is shown that objects moved together can basically be distinguished from objects moved separately by calculating the correlation coefficient of two nodes **x** and **y**

$$\rho_i(X,Y) = \frac{cov_i(X,Y)}{\sqrt{var_i(X) \cdot var_i(Y)}} \tag{4.1}$$

The algorithm is an incremental approach which periodically updates the correlation of node **x** after reception of the k latest samples of the second node **y**. The k transmitted samples are stored in a circular buffer on the node and correlation is calculated over $N = n \cdot k$ samples. Parameter n defines the overlap of the sliding window and k the window size. Both have an impact on the time delay of the computation.

Unfortunately, the paper gives no quantitative results about reliability and precision which could serve as reference for our considerations.
An interesting conclusion of the paper is that the main limiting factor for scalability of a WSN in conjunction with the correlation method is communication (rather than other constrained resources as memory, execution time or energy). If each node transmits a data sequence every Δt, the time available for all the nodes to transmit their data is $M \cdot t_{node} < \Delta t$ where t_{node} is the slot time for sending a packet[2], which depends on the radio transceiver. If simple features can be found that need a minimum of communication, i.e. Δt becomes larger, the number of nodes M can be increased.

A similar idea is pursued by J. Lester et al. [19]: during walking accelerations of carried objects are compared by the coherence function $\gamma_{xy}(f)$, a measure of linear correlation in the frequency domain. In the following considerations coherence is approximated by C_{xy}, the real-valued magnitude squared coherence (MSC). A general framework for coherence estimation is proposed in [38].

$$C_{xy}(f) = |\gamma_{xy}(f)|^2 = \frac{|S_{xy}(f)|^2}{S_{xx}(f) \cdot S_{yy}(f)} \tag{4.2}$$

where $0 \leq C_{xy}(f) \leq 1$, S_{xx} and S_{yy} the power spectra and S_{xy} the cross power spectrum.

Identical fully correlated signals result in $C_{xy} = 1$ and completely uncorrelated signals in $C_{xy} = 0$ at all frequencies. The FFT coefficients $x_k(f) = \mathcal{F}(a_k(t) \cdot h(t))$ and $y_k(f) = \mathcal{F}(b_k(t) \cdot h(t))$ are used to calculate the power spectra

$$S_{xy}(f) = \frac{1}{n} \sum_{k=0}^{n-1} x_k(f) \cdot \overline{y_k}(f) \tag{4.3}$$

Each of the signals $a(t)$ and $b(t)$ are split into n segments where the segments can overlap. The segments are multiplied by a time-weighting function $h(t)$. Settings

[2]Here, TDMA channel access method is assumed. For CSMA t_{node} would be the average time.

similar to those applied in [19] were used for our evaluations: smooth weighting with a Hanning window $h(t) = \frac{1-cos(\frac{2\pi t}{\omega})}{2}$, three windowed segments ($n = 3$), segments with 50% overlapping and a FFT size equal to the size of the segments. A similarity measure was calculated from the coherence by integrating the coherence function over a certain frequency range. The range was set to frequencies from 0Hz to $f_{max} =$ 10Hz (see also section 4.3 further below).

$$P_{xy} = \frac{1}{f_{max}} \int_0^{f_{max}} C_{xy}(f)df \tag{4.4}$$

Experiments from the paper [19] show following results: If the coherence is computed on data recorded from accelerometers worn by the same person, for sliding windows of 8s, mean coherences of over 95% resulted. Coherence based on 8s segments for data sensed by accelerometers on different people only shows an average of about 52%. This means that coherence is a discriminant feature for walking data. Even if people attempted to trick by walking in step coherence did not climb over 56%, which is a sign of coherence's high robustness.

The downside of the approach lies in the FFT's high computational costs and in the fact that it was not tested on a real-time embedded system. Window sizes of 8s seem rather unfeasible in real applications due to memory and computation effort needed. However, for smaller windows coherences decrease but for a window of 2s still a mean coherence value of 75% is reported by Lester et al. They found that the communication delay between two devices should not exceed 25ms at window sizes of 2s. For window sizes of 8s, recognition rates are not considerably degraded up to time delays of 500ms.

As each sample of node **x** has to be sent to node **y** for both approaches, large amounts of data need to be exchanged. This can result in an increased packet loss due to high traffic and is inefficient in terms of limited energy inherent in embedded devices.
Although good results in distinction of correlated from uncorrelated movement are expected, the remaining question is how correlation and coherence will perform on shaking data.

4.1.2.2 Local and Simple Features

The idea behind our approach is as follows: Are there features easy to compute without the need to exchange much data over wireless link such that they can be combined in a way to distinguish between correlated and uncorrelated shaking?
Objects are designated to be *shaken correlated* if they are shaken together and if they do touch each other during the whole shaking action. If objects do not contact one another and are shaken independently, shaking of the objects is defined as *uncorrelated shaking*.

The evaluation of potential features proceeds in two steps. A first set of features only comprises simple features in the time domain that do not require data from

	Feature	Description	Simple	FFT	Ref
1	Mean	Mean of windowed data	x	x	
2	Maximum	Maximum value inside a window	x	x	
3	Minimum	Minimum value inside a window	x	x	
4	Zero Crossings	Number of crossings of the average value inside a window	x	x	
5	Total change in signal	Sum of differences between each pair of consecutive samples inside a window	x	x	
6	Number of descending signal changes	Number of times a sample is smaller than its predecessor inside a window	x	x	
7	Number of ascending signal changes	Number of times a sample is larger than its predecessor inside a window	x	x	
8	Variance	Variance of windowed data	x	x	
9	FFT band 1	Number of FFT coefficients that fall into quantisation band 1		x	
10	FFT band 2	Number of FFT coefficients that fall into quantisation band 2		x	
11	FFT band 3	Number of FFT coefficients that fall into quantisation band 3		x	
12	FFT band 4	Number of FFT coefficients that fall into quantisation band 4		x	
13	FFT band 5	Number of FFT coefficients that fall into quantisation band 5		x	
14	FFT band 6	Number of FFT coefficients that fall into quantisation band 6		x	
15	FFT pair 1	Sum of FFT coefficients at 1Hz and at 2Hz		x	
16	FFT pair 2	Sum of FFT coefficients at 2Hz and at 3Hz		x	
17	FFT pair 3	Sum of FFT coefficients at 3Hz and at 4Hz		x	
18	FFT pair 4	Sum of FFT coefficients at 4Hz and at 5Hz		x	
19	FFT pair 5	Sum of FFT coefficients at 5Hz and at 6Hz		x	
20	FFT pair 6	Sum of FFT coefficients at 6Hz and at 7Hz		x	
21	FFT pair 7	Sum of FFT coefficients at 7Hz and at 8Hz		x	
22	FFT pair 8	Sum of FFT coefficients at 8Hz and at 9Hz		x	
23	FFT pair 9	Sum of FFT coefficients at 9Hz and at 10Hz		x	
47	Correlation	Correlation coefficient of windowed data from two nodes			x
48	Coherence	Coherence function of windowed data from two nodes			x

Table 4.1: Features for feature selection. Features are divided into simple features, FFT features and the features correlation and coherence which are used as benchmark. Simple features are features with low computational costs. FFT features rely on the frequency domain and the FFT transformation. In section 4.3.2.1 only features marked as “simple” are used. The features marked as “FFT” are applied in computations of section 4.3.2.2.

other nodes. A second set contains those simple features and additionally features in the frequency domain that can also be calculated individually but include beforehand the computation of an FFT. Table 4.1 lists the features of the two sets. Each of the features must be calculated twice, once on each sensor node. Instead of having one feature like correlation that takes two separate inputs, one from node **x** and one from node **y**, two identical features (a feature pair) get calculated on node

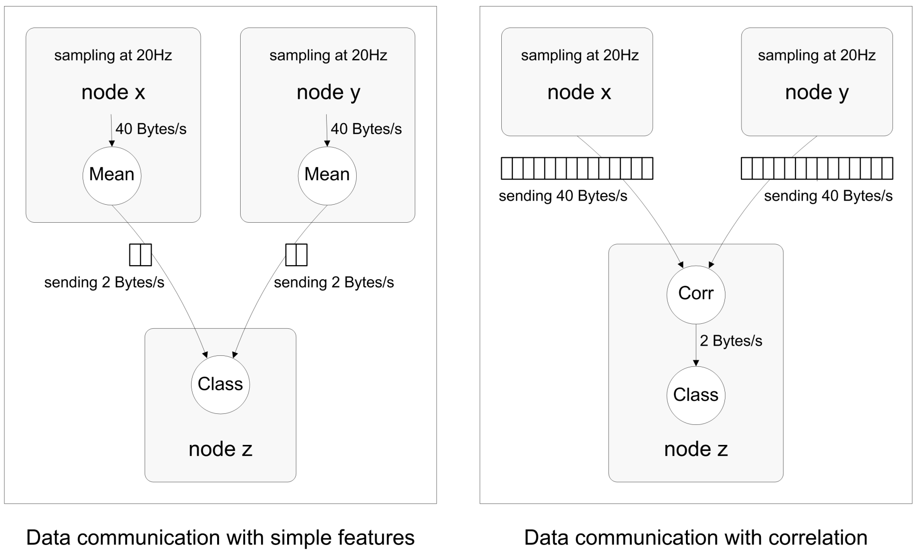


Figure 4.2: Left: First the simple features (e.g. mean) are calculated locally, then the features are sent by radio communication. Right: The data is transmitted, then features like correlation, which use two or more inputs from different nodes, are calculated.

x and **y** separately and are both used later in the classifier. When the features are computed individually only single features and no raw data must be communicated. Figure 4.2 shows this difference in the case of mean and correlation, where both are calculated on a separate node. Assuming best case in terms of communication, acceleration magnitude is calculated on each node before data exchange (data gets reduced to a third beforehand) and feature calculation and classification is done on one of the nodes which already records raw data (for Figure 4.2 feature computation (node z) and sampling (node x) would be done on the same node (node x)). This node then can make a decision or redirect the labels further. However, the node will require sufficient resources.

Wireless data communication varies depending on window size, sampling frequency, data size, messages' packet size, number of nodes and number of features calculated. The following considerations are based on our application: a window size of 1s (without overlaps), a sampling frequency of 20Hz, acceleration magnitudes of 16 bit, a packet size of 24 bytes and a network with 3 nodes **x**, **y**, **z** (as in Figure 4.2) are assumed. If one magnitude is sent per packet, 40 packets are to be exchanged to receive the decision for one single window segment using correlation. This number can be minimised to 4 when messages are used to full capacity. A total of 80 Bytes must be transmitted from node **x** and **y** to node **z**. Calculating simple features locally only needs 2 messages or 4 Bytes, respectively, independent of window size, sampling frequency and data format. As a consequence, the number of exchanged messages gets reduced by a factor in the range from 2 to 20. Even if more than one simple feature must be computed on each node this factor is not much influenced as features to be sent can be packed into the same message.

As applications get more complex additional data, such as a timestamp per magnitude, might be included into a packet, longer window segments or higher sampling frequencies might be needed, which all together rapidly increase the data volume (even if messages are packed optimally). Our approach of using local simple features is much less influenced compared to methods based on features using multiple input data, such as correlation.

The idea behind the simple feature was to "reproduce" the correlation coefficient that is composed of variance and covariance (cf. Equation 4.1). Simple features with information about the variance of data are: mean, maximum, minimum, total change in signal and the variance itself. The variance was left in the set but whenever possible replaced by one of the previous features. Variance uses multiplication, which the other features do not. Number of ascending or descending changes can be seen as a kind of correlation feature, i.e. a measure for linear dependence of data. Ascending and descending changes are rather the same for shaking movements and expected to perform equally well over large data set but both are included in the feature set because the combination of both could add discrimination to the overall system.

Further features for acceleration based activity recognition could be: root mean square (RMS) or cumulative sums. Mean and cumulative sum are the same. Therefore the cumulative sum is omitted keeping in mind that instead of the mean also the feature cumulative sum could be applied. The RMS describes the average power of a signal (Parseval's theorem) $\frac{1}{N}\sum_{i=1}^{N} x_i^2$ which is covered by the mean and the variance $\frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{X})^2$.

The feature zero crossings is the only time domain feature which provides frequency information. Frequency information is assumed to contain additional information, such as periodicity of a signal. Deviations in certain frequency bands aid to keep similar but uncorrelated movements apart. According to [34], using discrete FFT coefficients results in high recognition rates. For the features "FFT band", the magnitude of the FFT coefficients is quantised into 6 exponential bands. Then the number of magnitudes in each band is counted. The features "FFT pair" in contrast sum up two consecutive FFT coefficients up to frequencies of 10Hz.

4.1.2.3 Feature Selection

Not all of the features listed above perform necessarily well for our purposes. Some combinations will outperform others. Feature selection aims to reduce the number of features and to enhance performance. Selection of the best feature mix is addressed from two sides: first a complete search is run on the feature sets, second an approach from information theory using Mutual Information Feature Selection (MIFS) [39] is applied.

- *Complete Search* Complete search tests out all possible combinations of feature numbers and compositions. The feature mixes found are ranked using Pareto optimisation: a 3-dimensional Pareto front is built based on the performance metrics precision, recall and specificity. The highest ranked feature mixes get stored.
  A brute force exhaustive search guarantees to find the best results independent of any assumptions but encounters the problem of exploding computing time. Figure 4.3 shows the rise in computing time with increasing number of feature combinations. Complete search is feasible for feature sets with a limited number of features but already for a set of $n = 23$ features, such as the set with the FFT features ($2 \cdot 23$ features), there exist a maximum of $\binom{n}{k} = 1'352'078$ possible feature combinations at $k = 12$ features. Here only feature mixes that contain just a few features or nearly every feature can be achieved in practicable time.

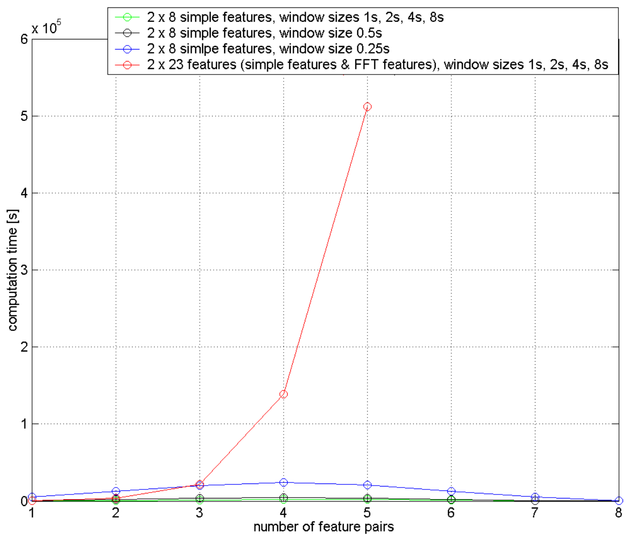


Figure 4.3: Time curves for complete search. Computation time increases exponentially.

- *MIFS* MIFS is a feature selection method based on mutual information that offers an alternative to the complete search. First the mutual information $I(f, C)$ between each single feature f and the classes C is calculated. The feature that maximises the mutual information is selected. In each following step one of the remaining features is added to the set S of previously selected features. The newly selected feature f_i must maximise $I(f_i, C)$ and minimise $I(f_i, S)$, i.e. it maximises $max_i\ (I(f_i, C) - \beta \cdot I(f_i, S))$. The higher the parameter β, the more redundancy among the features is included in the feature selection.

Feature selection with MIFS was carried out by using the *Feature Selection Toolbox* which was implemented in a semester thesis [40]. The thesis evaluated filter-based feature selection methods based on empirical, information theoretical and statistical approaches with the goal to find the best subset of sensors in a multi-sensor system. It found Analysis of Variance (ANOVA) [41] in combination with the Pearson correlation to perform best. An accuracy of around 90% was already achieved with a set of three features using a Nearest Class Classifier (NCC). The Kruskal-Wallis test [42] combined with the Spearman correlation offers an alternative that is independent of any distribution. But [40] shows that Kruskal-Wallis gets a lower recognition rate. MIFS achieves nearly similar performance as ANOVA and runs faster than selection methods based on Random Forests.

As selecting the best feature mix poses an analog problem to the selection of the best subset of sensors, feature selection can be solved by means of the same methods. ANOVA cannot be applied to our data because shaking does not produce normal distributed data due to saturation and sections with no movement. MIFS seems to be a valuable alternative. Since features always come in pairs MIFS was modified in such a way as to calculate mutual information for both features of the pair and to add them up subsequently.

4.1.3 Classification

Activity recognition in the context of the chosen demonstrator comprises the recognition of correlated and uncorrelated shaking, the detection of rolling the dice and the provision of the score (dice side recognition). We especially focus on the most challenging: discrimination between correlated and uncorrelated shaking of objects. Correlated shaking is well specified by features that reach similar values. However, uncorrelated shaking can be anything. Both objects can remain unmoved, only one object can be moved or both objects can be shaken disparately in numerous modalities and combinations. The data sets to deal with consist of only two classes, one of them being the NULL class. There the difficulty is that the NULL class can not be modelled.
The considered classifiers are based on supervised learning, i.e. they must first be trained by a training set before they are ready to use. Classifiers' performance is then evaluated by a test set. Due to limited computational power rather simple classifiers are used which can be trained offline and work rather efficiently online.

4.1.3.1 NCC

The Nearest Class Classifier (NCC) is trained to learn the centroids of the sample clouds for each class. New data gets classified by calculating the distance from each data point in the data set to all the centroids. The centroid that is located nearest to the data point defines the point's class label. The NCC is a very robust classifier [43]. It has a relatively high error on the training data and on the test data, too. But the error on the training data is a good prediction of the error on the test data.

NCC needs at least two classes (aside from the NULL class) to make a relative decision. Therefore it is basically not applicable for our application. Features as correlation and coherence are an exception: as their values reach from 0 to 1, with values close to 1 being one class and values near 0 indicating the NULL class, the NULL class can be seen as a second class and NCC might be applied. But for the simple features and the FFT features, both low and high values belong to the NULL class, which disables relative decision making.

4.1.3.2 Similarity Search

NCC is replaced by a related classification method: the Similarity Search classifier. Without a NULL class model a sort of a similarity measure is needed that defines when data begins to be part of a certain class. [22] applied a feature similarity search to spot relevant activity events and stated the feasibility for online implementation.

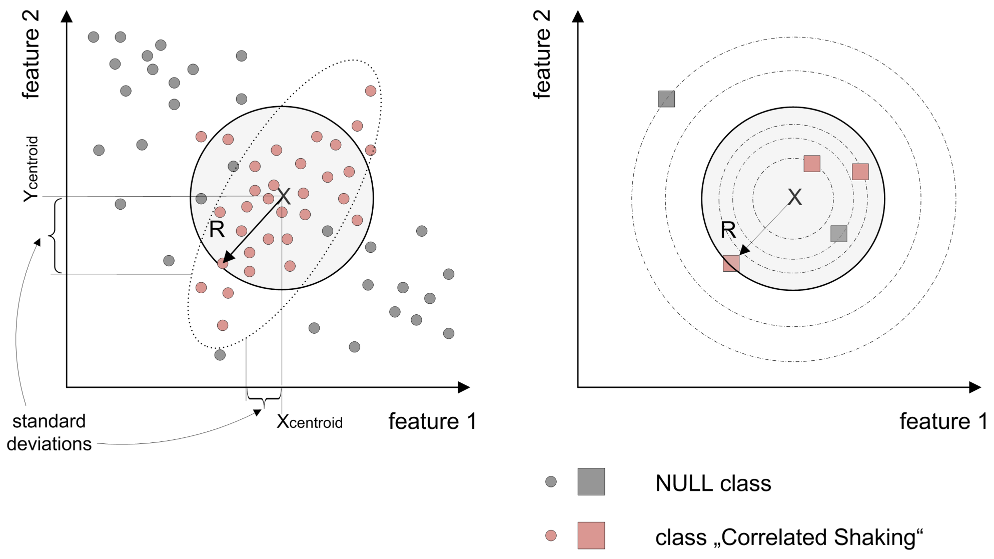


Figure 4.4: Principle of Similarity Search classification. The example shows two classes, the NULL class and one positive class "Correlated Shaking" and describes the principle of Similarity Search for the 2-dimensional case with two features, one assigned to each axis. Left: The centroid's coordinates and the standard deviations are determined from the training samples that belong to the positive class. For training and testing all samples get normalised by the standard deviation. Thus the samples can be classified using a single radius. Right: The optimum radius is found by continuously incrementing its value. Each time a new sample is included into class "Correlated Shaking".

Each feature input to the classifier is attached to one dimension of feature space. During training of the classifier, for every single class (for one class in our case) the coordinates of the centroid and the standard deviations into all dimensions are calculated. Calculation only uses those sample points of the training data labeled with the current class. The training data is normalised by dividing by those standard deviations. For each class the distance from the centroid, i.e. the radius, that best discriminates between the class labels and the different labeled sample points serves as the similarity measure. The maximum allowed distance for accepting a sample around each centroid is continuously blown up and the product of recall and precision is computed (see Figure 4.4). The larger the radius the more samples are contained within the circle of maximum allowed distance. Each time a sample is included precision and recall get updated. This way an optimal radius can be found. As radius, Euclidean and Manhattan distance have been used.
At the output stage of the classifier a smoothing filter is added. The filter behaves similar to a median filter and is designed to reduce insertion errors. As long as positive recognition of correlated shaking is obtained less than n times in series recognition is rejected and samples are classified into the NULL class. Parameter n is set to 4 for window size 1s and to 2 for window size 4s.

4.2 Experiments

4.2.1 Experiment Description

An experiment is conducted to record acceleration data for training and testing the classifier. The experiment enables the analysis of the influence that different parameters have on the performance of activity recognition for shaking.

Five different test subjects were directed to perform differing shaking actions with two foam dice for six runs, each with a duration of 3.3 minutes. The first three runs the subjects were told to hold the dice in contact shaking correlated such that the orientation, speed (or frequency) and strength (or amplitude) were varied. The forth run the subjects were allowed to shake the dice at will but not correlated, i.e. not contacting, holding one dice in each hand. The last two sets are intended to test robustness of the classifier: first one and then two subjects tried to confuse the results in trying to shake in a correlated way. A detailed description of the experiments can be found in Appendix B.

4.2.2 Sensor Data Acquisition

Two Tmote Sky motes extended with the 3-axis accelerometer MMA7260Q are used in the experiments. Tmote Sky was preferred to the Switch Module with Tmote Mini because Tmote Sky communicates over USB and not wirelessly thus not losing any data. Tests showed, if Tmote Mini sent sampled data at frequencies above 20Hz over radio one sender will be dominant and at 50Hz packets from only one mote were received. Wiring the motes allows for higher data rates and more reliable data recording, which results in higher signal quality and much better data synchronisation. In the experiments the accelerometers were sampled at 20Hz. This frequency is certainly high enough to fulfil the Nyquist criterion for a 10Hz bandwidth and remains in the frequency range that is reachable with the real wireless device.

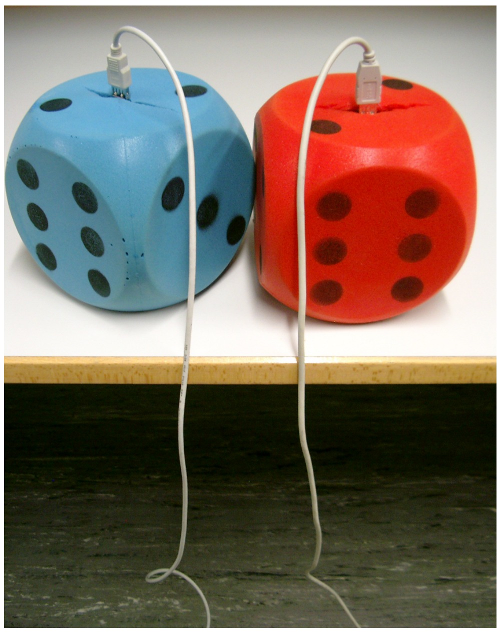


Figure 4.5: Setup for the experiments: two dice with Tmote Sky inserted.

Additional data was recorded at 10Hz and 50Hz. In total, over 170 minutes of accelerometer data could be provided for evaluations. The data sets were labeled later on by visual inspection. For the cross validation (see section 4.3 below) 27 data sets (data sampled at 20 Hz, from 5 different subjects) were selected. These data sets consist of 5312 segments, each segments comprising data of 1s or 20 samples. 2866 segments (54%) belong to data where the subjects moved the dice and for the remaining 2446 segments (46%) the dice were not moved at all. The NULL class contains 3808 segments (72%), i.e. the dice were shaken correlated for 1504 segments (28%). As can be seen from these statistics the NULL class was rather large due to a high portion of unmoved samples. This portion of unmoved samples does not influence the training of the classifier because the centroids and standard deviations of the Similarity Search classifier are evaluated by considering the samples outside the NULL class exclusively. The radius is not influenced either as the samples with no movement are farthest away from the centroids. The only influence when having a rather high section of unmoved data is that the performance measures for accuracy and specificity might get too promising. To take that in consideration recognition rates were recalculated, this time the sections of the NULL class without any movement being removed from the data sets. This correction[3] leads to a total of 2866 segments, a new number of 1362 segments (48%) in the NULL class and 1504 segments (52%) in the class “Correlated Shaking”. The ratio between segments in the NULL class and segments in the class “Correlated Shaking” is reduced from 2.5 down to 0.9, i.e. the two classes get better balanced. This way, more significant performance values are expected, which better allow comparisons to performances achieved by other groups.

Although the motes deviate from the real application in being wired, the difference was kept as small as possible. The motes were inserted into the foam dice and the attached cable were attended to be long enough such that subjects felt no special obtrusiveness by the cables (see Figure 4.5).

4.3 Evaluation of Activity Recognition Algorithms

Evaluation of the algorithms is based on MATLAB. Scripts have been written to perform the feature selection, to train and test the classifiers and to further analyse the performance of the system. The *Feature Selection Toolbox* [40] was integrated into the own MATLAB code and extended by reworked MIFS sort, complete search and Similarity Search classifier.

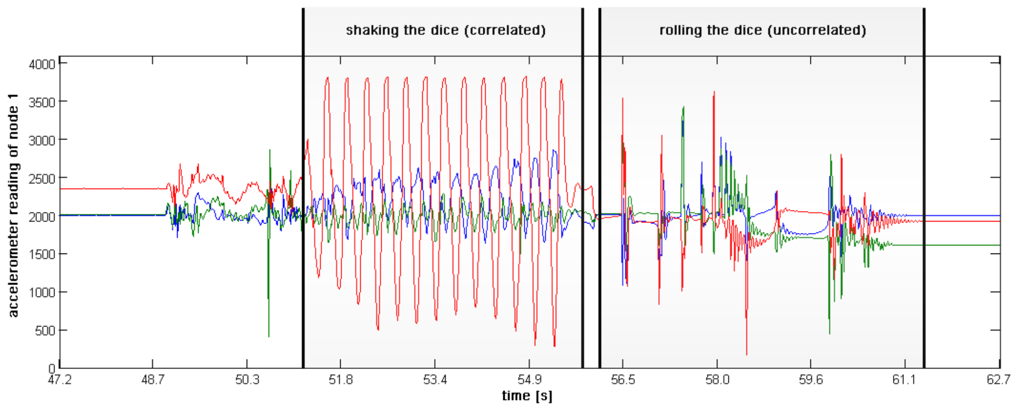


Figure 4.6: Representative example accelerometer data from shaking and rolling the dice. The data shows the course of action: first the dice remain unmoved, then they get shaken and finally rolled. Each step is detected and the score should be returned in the end.

[3]In the further sections of this chapter, performance measures appear in the text and figures. It is distinguished between the *original data set* and the *compensated data set*. For the original set the recognition rates are calculated based on the original raw data, keeping the high portion of the NULL class in the set. Performances for the compensated set are calculated after the correction, i.e. after the parts without movement were removed from the data and the NULL class.

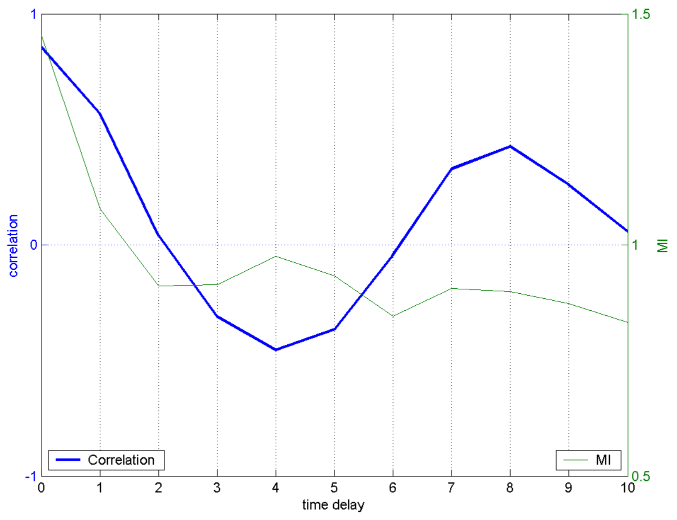


Figure 4.7: Correlation vs. mutual information. Correlation and MI are plotted over time delay. Each increase by 1 in the time delay means that the data streams are misaligned by one more sample or 50ms. The curves show a qualitative comparison, an exact relation between correlation and MI does not exist [44].

27 sets out of the data sets recorded in the experiments were selected for evaluation. Leave-one-out Cross-Validations (LOOCV) were performed on those data sets. Always one set was removed and the classifier was trained on the remaining sets. At the end the mean performance is calculated. The feature mix that gave the best cross-validation score was trained with all data, and that resulted in the predictive model to use. LOOCV is expensive but allows to get the most out of the data, which is an advantage if relative small data sets are available.

Sampling frequency and window size are important parameters in recognition of activities like shaking. [19] states that the frequencies of human motion lie within the range of 1Hz to 10Hz. Our experiments could confirm this range and showed typical shaking frequencies of 2Hz to 5Hz (Figure 4.6). Sampling frequencies must thus take an absolute minimum of 10Hz and to be on the safe side 20Hz or even higher values (e.g. 50Hz) should be chosen.

Window sizes of 0.25s, 0.5s, 1s, 2s, 4s and 8s were selected for evaluation. At a sampling frequency of 20Hz that corresponds to a window length of 5, 10, 20, 40, 80 and 160 samples. As the duration of a shaking action is aimed to be shorter than about 5s a maximum window size of 8s is set. More than 8s is felt as a very long time and thus for a real-time system not practical. From the point of implementation small window sizes are optimal because of reducing the risk of overflowing variables whereas larger window sizes may reduce communication.

4.3.1 Features for Benchmarking

Correlation was chosen as a benchmark for the performance of other features. It seems to be a natural choice to find correlated features containing information of linear dependence. As a preliminary study the correlation is compared with mutual information (MI) [44] for shaking movements. The calculation of a probability distribution function (PDF) for MI computation used $\sqrt{n}$ bins where n is the number of data samples. The behaviour of MI and correlation was analysed for increasing time shifts between acceleration data from a node $\mathbf{x}$ and node $\mathbf{y}$. By increasing the misalignment the required synchronisation accuracy can be evaluated. Figure 4.7 shows representative curves for correlated movement of two objects. MI is more sensitive than correlation, which can be seen from the steeper slope at small lags. Correlation is more robust against slight delays, which normally occur with wireless communication. When delays get too large, correlation increases again due to the underlying periodicity in shaking data whereas MI remains largely unaffected as it also detects non-linear dependencies. Since the devices sampled with equal frequencies in our experimental setup and synchronisation[4] has been applied to the data, only marginal delays are present. Moreover, at similar computational costs, correlation avoids quantisation effects through binning. We conclude that correlation is an adequate measure for the similarity of data.

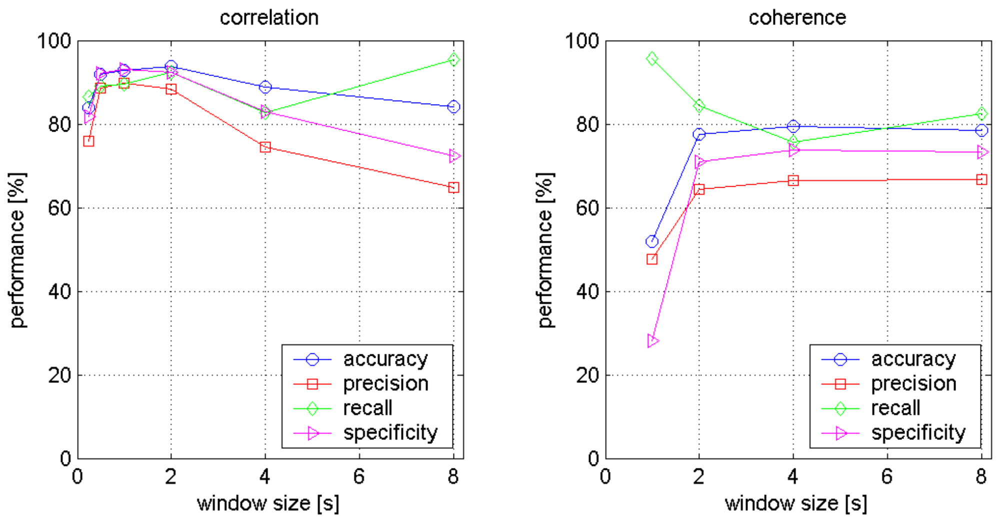


Figure 4.8: Left: Performance metrics of correlation for varying window sizes. Right: Performance metrics of coherence for varying window sizes. The performance is calculated for the original data set.

[4][45] and [46] have implemented timing protocols on TinyOS with accurateness of a few 10μs.

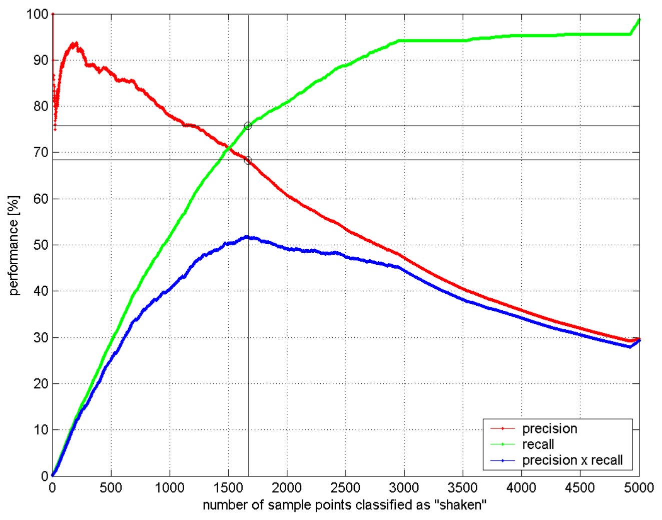


Figure 4.9: Pareto optimisation: weighting function, precision and recall. The performance is calculated for the original data set.

The advantage of the frequency domain is its independence from time delays. The frequency domain is especially suited to detect periodic movements, such as walking or shaking. But as can be seen from Figure 4.8 in the case of coherence, the strength in recognising similar movements can quickly turn into disadvantages: independence from time delays and sensitivity to periodic movements may reduce the discriminant power. Thus, classification results in high recall (FN = 0) but lower precision (FP also high). Possibly, even if a person moves objects uncorrelated, there exists a rather high correlated portion in the data due to the person's inherent correlation of movements.
As consequence it was found that correlation as a whole performs better than coherence in distinguishing between correlated and uncorrelated shaking. Correlation works optimally for shorter window sizes between 0.5s to 2s whereas coherence gives useful results from 2s upwards.

4.3.2 Evaluation of Best Feature Mix

The existence of a NULL class and its large part in the data (originally 72%) complicate the process of finding the right metrics to rate performance achieved with different feature combinations. Accuracy loses much of its expressivity if more activities occur in the data set that belong to the NULL class than to the positive "Correlated Shaking" class. Additionally accuracy does not differentiate between the different error types, such as insertions and deletions. The problem was handled by including different metrics. Precision (indicates insertions), recall (indicates deletions) and specificity can serve as objective functions to reduce the design space to

Pareto optimal feature sets. But in contrast to a global optimum, numerous Pareto points result depending on the window size and the feature number. As shown in Figures 4.9 and 4.13 precision and recall are complementary metrics and a trade-off between precision and recall must be made. To select a design point from all the Pareto optimal points on the Pareto front a weighting function must be used. For feature selection we considered precision rate to be more important than recall, as the number of insertions is reduced in exchange for an increase of deletions. In terms of the demonstrator that means: rather shake twice to be recognised than getting a false alarm each other time. For training of the Similarity Search classifier $f_w = precision \cdot recall$ was chosen as weighting function. We want to arrive at a Pareto point that keeps both rates approximately equally high not to degrade recall any further (see Figure 4.9).

Figure 4.10 can provide an indication of the discriminant power of the features under consideration. The overview confirms that the recognition rates of the single features might change considerably with varying window sizes.

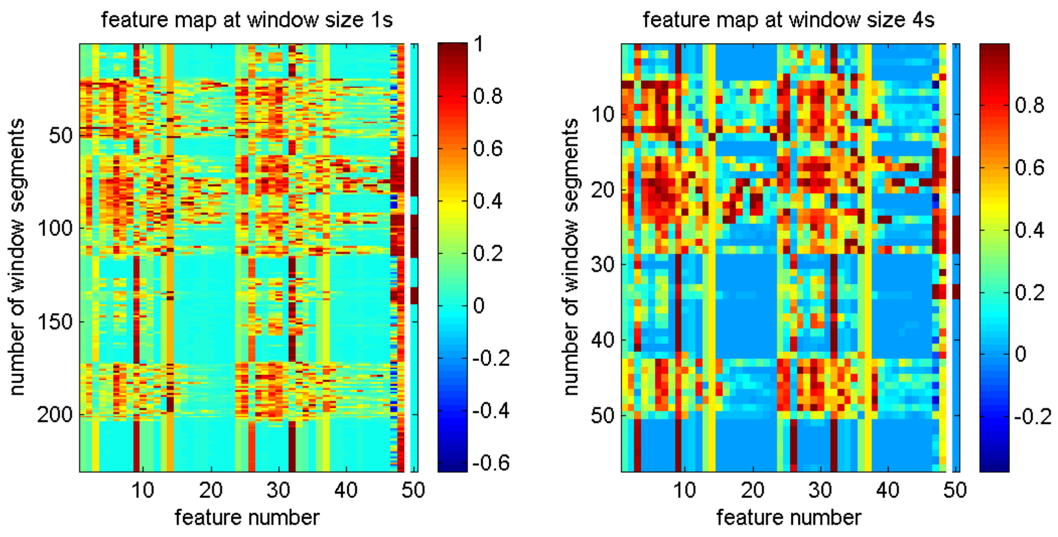


Figure 4.10: The two colour maps graphically represent the values of the different features for a set of 230 windows. The features were calculated for window segments of 1s (left) and 4s (right). The vertical axis contains the sequence of windows in the set. The number of the different features is assigned to the horizontal axis. Number 1 to 23 are the features calculated for the data from node **x** (and correspond to the numbers assigned in Table 4.1). Number 24 to 46 represent the same set of features, calculated for the data from node **y** this time. Number 47 is the correlation and number 48 the coherence. The last bar at 50 shows the label: red means class “Correlated Shaking” and blue stands for the NULL class. It can be seen for example that feature coherence has a relative high insertion rate at window size 1s (left map, e.g. red parts at (48,160) and (48,220)), which is much reduced at window size 4s (right map).

4.3.2.1 Simple Features without FFT

Now we try to find a best feature mix based on simple features in the time domain (cf. section 4.1.2.2). First, results obtained by LOOCV and Complete Search are discussed. The Similarity Search classifier was applied on data types double using the Euclidean distance measure.

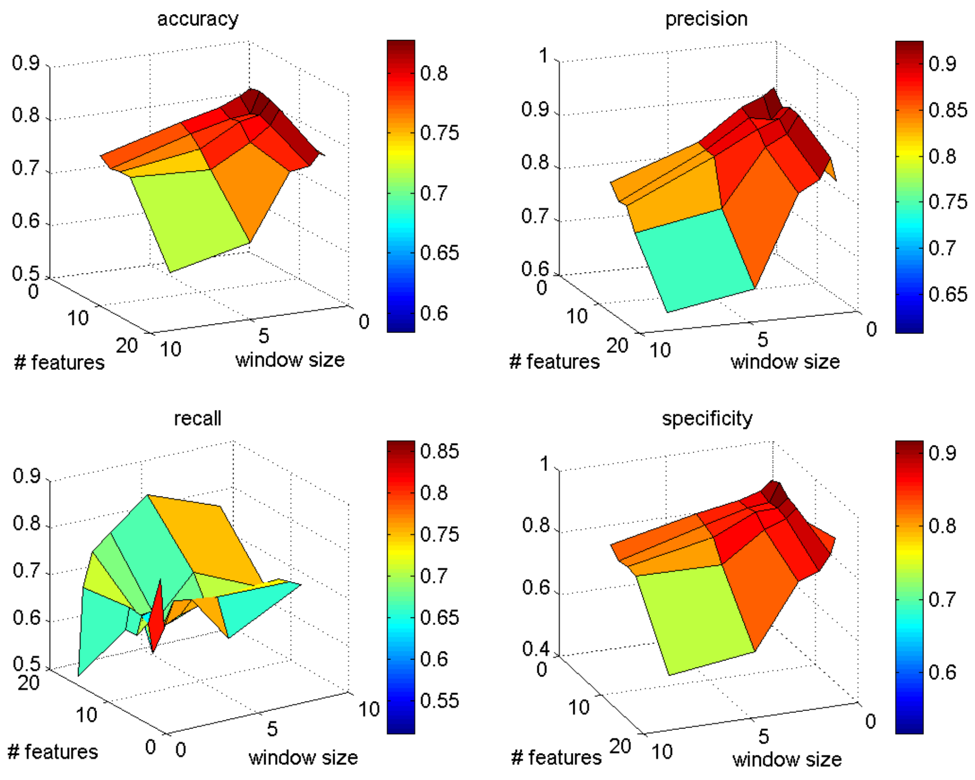


Figure 4.11: Performance metric evaluation.

Figures 4.11 and 4.12[5] present a compilation of the feature sets with the best recognition rates for different window sizes and feature numbers in the set. The optimum window sizes lie between 0.25 and 1s and the optimum number of features in a set is 2 or 3 feature pairs, i.e. 4 or 6 features in total, half from node **x** and half from node **y**. Thorough examination of feature combinations consisting of 2 or 3 feature pairs with window sizes in the aforementioned range reveals the following result: *mean* and *zero crossings* complement each other to a discriminative feature mix. They result in an accuracy of 73.5%, a precision of 78.2% and a recall of 68.6% at a window size of 1s (for the compensated data set). Including further features

[5]In addition to Figure 4.11 features zero crossings, number of descending changes and number of ascending changes have been enhanced by adding a threshold against noise.

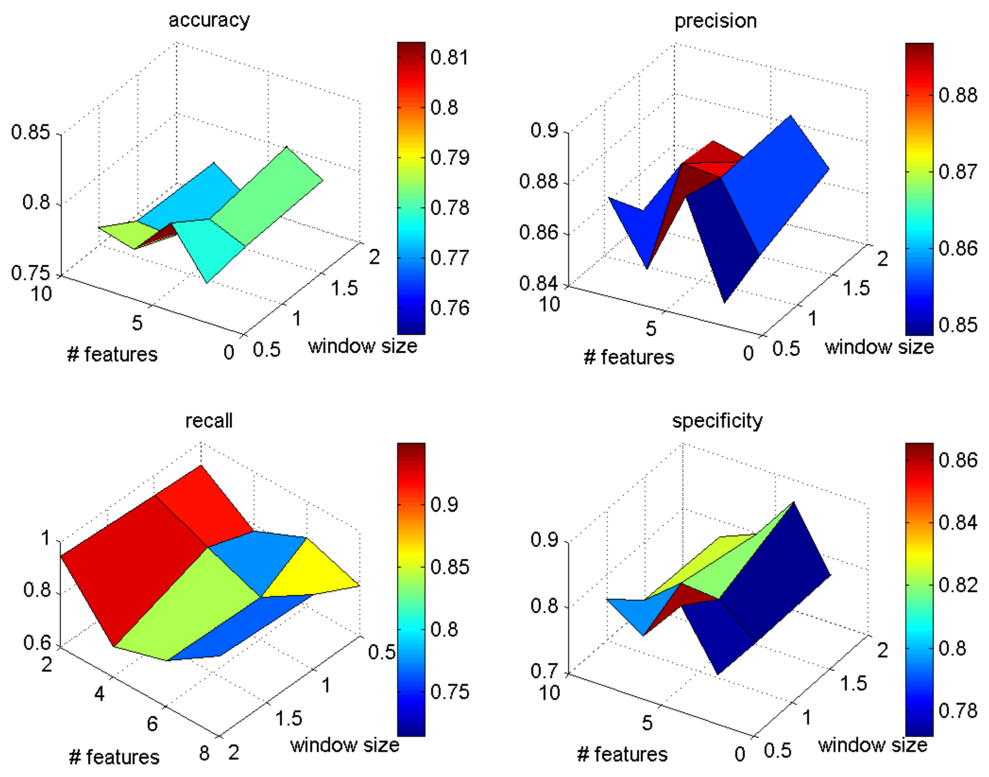


Figure 4.12: Performance metric evaluation. Close-up: window sizes 0.5s, 1s and 2s.

does not add improvement. Recognition rates may remain unaffected or even start degrading again.

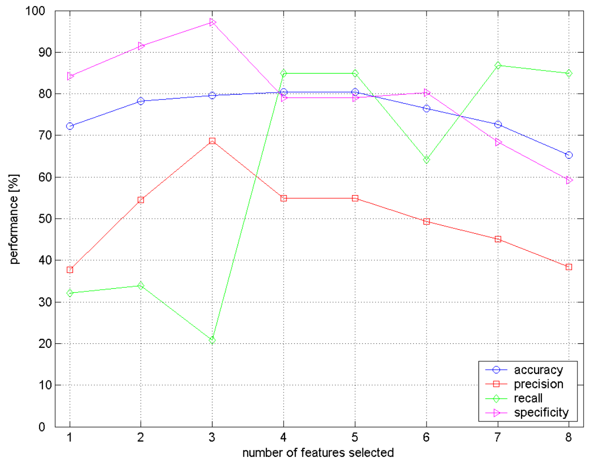
Shaking is a repetitive movement. To better understand why mean and zero crossings are a good choice, we draw a parallel to the model of a sine wave $y(t) = A \cdot sin(\omega \cdot t + \varphi)$. Amplitude A defines the absolute elongation. Features mean, variance, maximum or minimum and total sum of changes are measures of the elongation. ω describes the frequency of the signal and can be measured by the zero crossings (or FFT features in the frequency domain). Eventually, phase φ[6] gives the displacement which is detected by the right choice of the *window size.*

Alternatively, feature selection was addressed using MIFS. Results do not show exactly the same order as Complete Search (see Figure 4.13): maximum is ranked first, mean and zero crossings are at second and third position. Then number of descending and number of ascending changes follow. Mean and zero crossings rank

[6]If the window size is enlarged the influence of the phase decreases.

under the first four features selected by MIFS for small window sizes (0.5s, 1s, 2s) over all the data sets. The parameter β with which the degree of redundancy in MIFS is set was varied in the range of 0.5 up to 0.9 and had no significant influence on the results.

MIFS thus provides the correct tendency which features are more interesting, but cannot determine the concrete set of features for recognition of shaking. A reason could be that the variations in between recall and precision are too high. Maybe the combination of features from node **x** and node **y** to one feature pair (presently a sum) might have to be changed to provide a better balance.



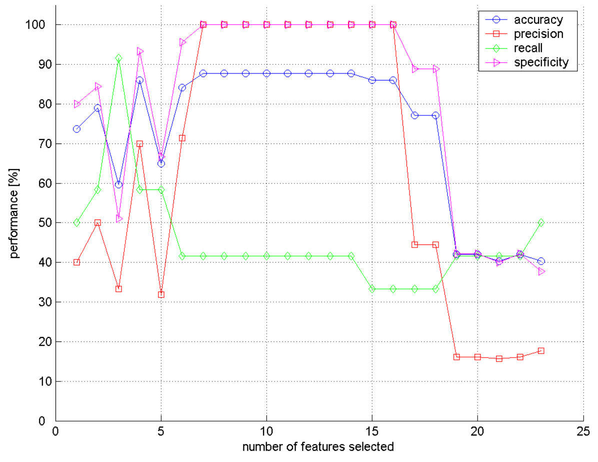


Figure 4.13: Feature selection with MIFS for the simple feature set (top) and for the feature set with additional FFT features included (bottom). Depending on the feature number either precision or recall prevails. MIFS is computed on the original data set.

The features mean and zero crossings are tested on the data sets. Classification results are shown in Figure 4.14.

4.3.2.2 FFT Features

An investigation based on the same procedure as previously described for features in the time domain was conducted. The FFT features (cf. section 4.1.2.2) were added to the set of simple features. As the feature set was now enlarged to $2 \cdot 23$ features, simulations could be run only up to five feature pairs contained in the same set due to computational complexity (see Figure 4.3). Complete Search showed the features *mean*, *FFT band 2* and *FFT band 3* to give especially good results for a window size of 4s. An accuracy of 76%, a precision of 86.0% and a recall of 64.4% were achieved (for the compensated data set). Zero crossings, the frequency feature in the time domain, is replaced by the two FFT features. [34] analysed various features for activity recognition and showed that FFT bands performed well for activities as hopping which is comparable to shaking in its periodicity and occurrence of higher accelerations.

3D-plots of performance metrics evaluation can be found in Appendix B. Figure 4.15 shows the classification result when the feature set is applied to the same data sets as the simple feature set before (see Figure 4.14).

Again MIFS provided the same trend of promising features (see Figure 4.13). The feature set consisting of mean, FFT band 2 and 3 was also included within the higher ranked features.

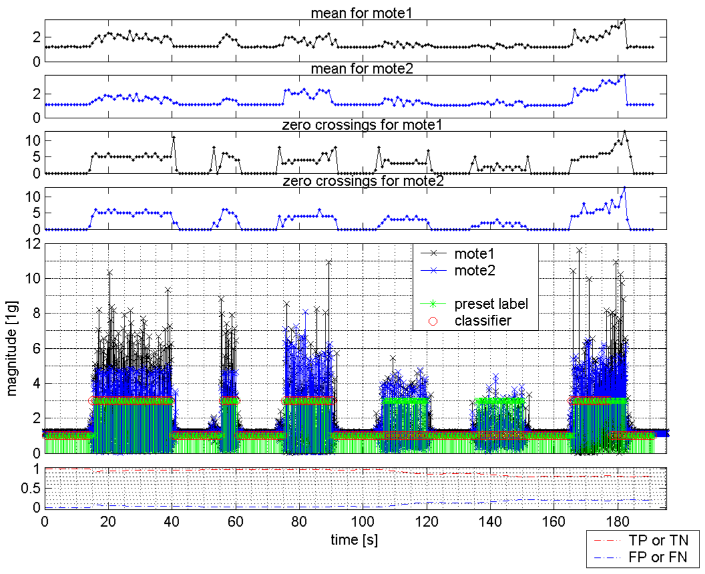
4.3.3 Tests Concerning Robust Recognition

4.3.3.1 Classifier Confusion

In two of the experiments conducted (cf. section 4.2) the subjects were told to trick the system by shaking the dice in the following ways:

- Experiment 5: The more challenging of the two experiments asked a single subject to hold one dice in each hand and to try to shake them as correlated as possible.
- Experiment 6: Two subjects tried to shake such that their shaking movement was as similar as possible. The data might look quite correlated although it is not.

The classifier which was trained on the 27 data sets of correlated and uncorrelated shaking from the first four experiments was applied to the new data sets of experiments 5 and 6. Figure 4.16 presents the results. Confusion by two people was recognised well by the classifier, for the simple feature mix as well as for the feature mix with FFT features. Confusion by a single person was still recognised in the case of using the simple feature mix whereas the classifier often failed for the



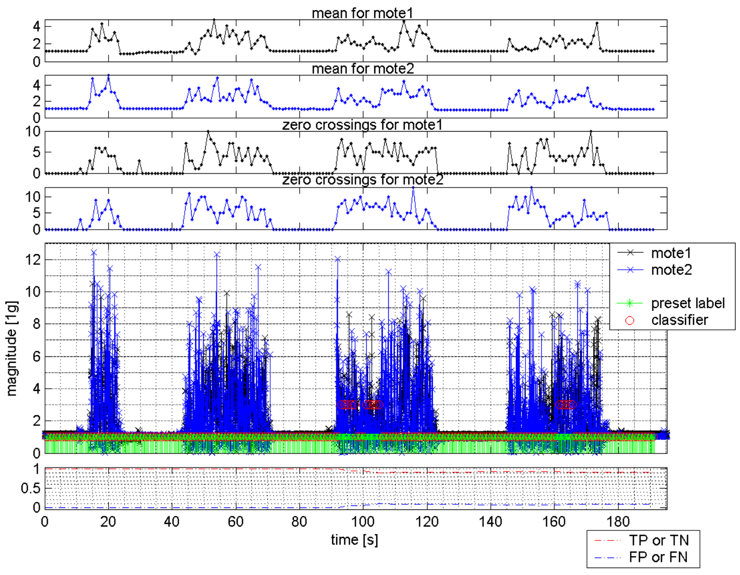
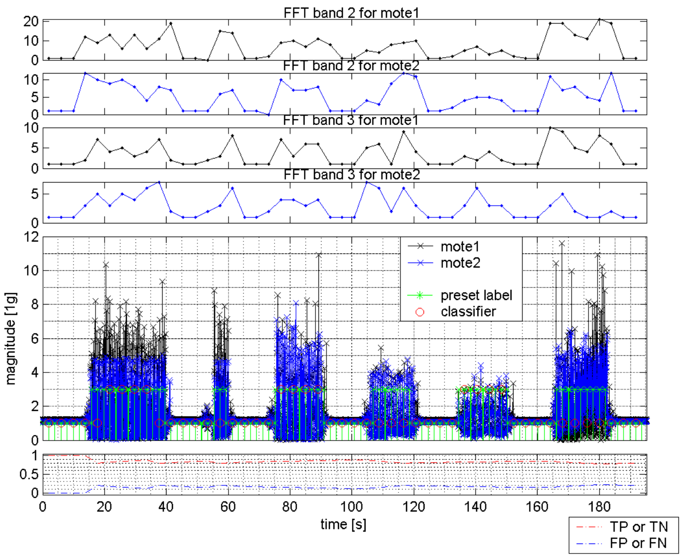


Figure 4.14: Acceleration magnitudes scaled to 1g plotted over time. The simple feature mix with features mean and zero crossings and window size of 1s is tested. The green markers designate the annotated labels while the red markers are the labels assigned by the classifier. In the bottom of each figure the cumulated number of true (TP or TN) and false (FP or FN) classifications is plotted. Top: Correlated shaking. Four of six shaking patterns were recognised as shaken correlated. This corresponds quite well to the found recall of 68.6%. Bottom: Uncorrelated shaking. There is insertion at some parts of the data sequence.



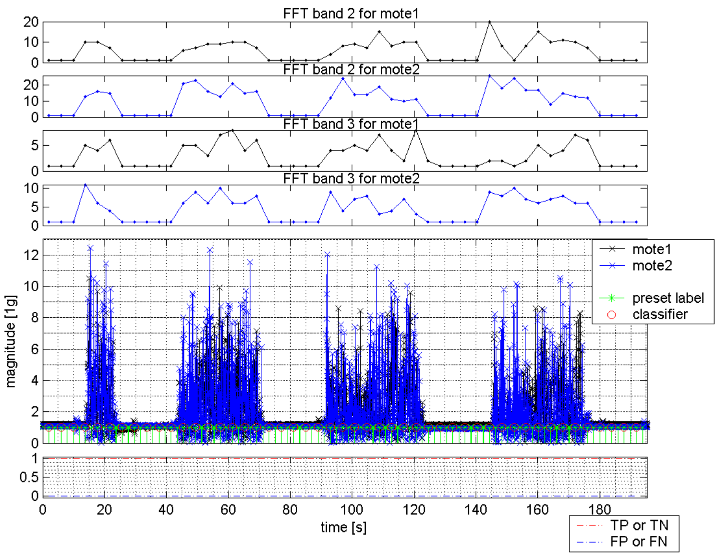


Figure 4.15: Acceleration magnitudes scaled to 1g plotted over time. The feature mix including FFT features with features mean, FFT band 2, FFT band 3 and window size of 4s is tested. The green markers designate the annotated labels while the red markers are the labels assigned by the classifier. In the bottom of each figure the cumulated number of true (TP or TN) and false (FP or FN) classifications is plotted. Top: Correlated shaking. Three of six shaking patterns were recognised as shaken correlated. The short shaking pattern (about 5s) is deleted due to the larger window size. Bottom: Uncorrelated shaking. Recognition was improved: insertions were removed corresponding to the higher precision value of the FFT feature set.

FFT feature mix. These results are consistent with the findings of R. Mayrhofer and H. Gellersen [20]. In the cases of two people and of a single person which shakes two objects, holding both in the same hand or in direct contact two each other, recognition does work well using FFT features. If one object is shaken in each hand FFT features can but do not have to recognise deviations from correlation.

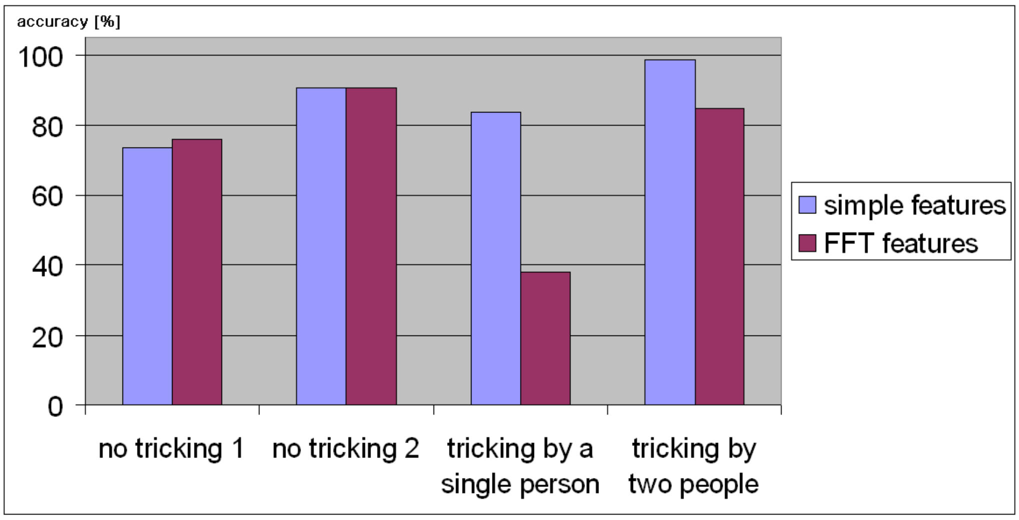


Figure 4.16: Accuracies are listed for recognition based on the simple feature set (mean, zero crossings) and on the feature set additionally including FFT features (mean, FFT band 2, FFT band 3). The performance is computed for the compensated data set. No tricking 1: The classifier was tested on all 27 data sets from the first four experiments. No tricking 2: The classifier was tested on a reduced number of data sets which were similar in size to the sets gained by experiments 5 and 6 and contained mainly uncorrelated data. Tricking by a single person: One single person held one dice in each hand and tried to shake them correlated. Tricking by two people: Two people tried to shake one dice each in a correlated way.

Figure 4.16 indicates that zero crossings with small window sizes capture true correlation better. Zero crossings show the phase between the signals while FFT features do not, i.e. the same frequency in the signal results in the same value and a false positive classification result for the FFT features.

Confusion could only be tested on a subset of 5 sets (5 sets for “tricking by a single person” and 5 sets for “tricking by two people”). The small number of data sets could be another reason for the high deviations in performance of simple features and FFT features. An indication might be the difference between “no tricking 1” (this test result based on 27 sets) and “no tricking 2” (only tested on 5 sets).

4.3.3.2 Person-Independency

Next, deviations in shaking of different people is analysed. The goal is to estimate how classification results are influenced by data from different people in the data sets.

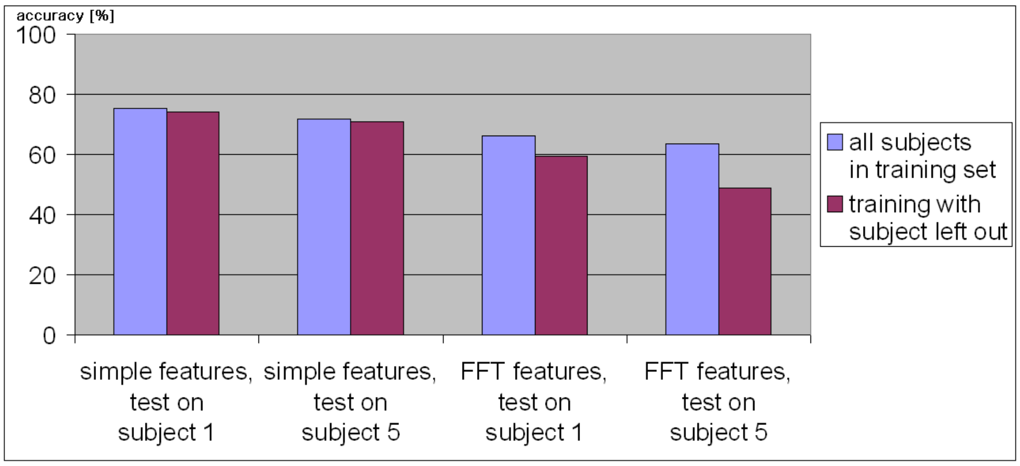


Figure 4.17: Recognition rates if the classifier is only trained on a subset of the subjects. The accuracies are compared to those achieved when the classifier was trained on the data from all the subjects. The accuracy is calculated for the compensated data set.

The classifier was trained on $n - 1$ subjects and then tested on the subject left out. Table 4.17 shows the recognition rates obtained for the subjects 1 and 5. The classifier was once trained on all five subjects (for the found simple feature mix as well as for the FFT feature mix) and then tested on the data sets from subject 1 or subject 5. In a second run, the classifier was trained on all other subjects except subject 1 or 5, respectively.
The recognition rates are lower when one subject was excluded from the training set. Degradation of classification based on the simple feature mix was nearly negligible (lower than 1%). The feature set with FFT features caused a reduction in accuracy of 5%. An explanation might be that shaking by different subjects vary mostly in the average rate of shaking frequency, which would influence the FFT features directly.
All in all, we conclude that shaking movements performed with objects, like foam dice, are similar for different people. For robust classification a certain diversity must be presented to the classifier but classification does not require extra adjustment to a single person.

4.3.4 Implementation Considerations

Computations in MATLAB are not constrained, i.e. with MATLAB operations can be rather complex and calculations can be based on the data type double, which results in high resolutions. In contrast, the implementation on a microcontroller is bound to limitations in operation's complexity and precision.

4.3.4.1 Manhattan vs. Euclidean Distance Measures

Euclidean and Manhattan distances were applied in the Similarity Search classifier as measures for the radius. The following values were computed for the simple feature set (mean and zero crossings at window size 1s) with data type double. Both distance measures roughly resulted in the same recognition rates: an accuracy of 72.7%, a precision of 78.3%, a recall of 66.4% and a specificity of 79.7% for the Manhattan distance and an accuracy of 73.5%, a precision of 78.2%, a recall of 68.6% and a specificity of 78.9% for the Euclidean distance.
Manhattan distance has lower computational costs. Basically Manhattan distance can be used instead of Euclidean distance without losing performance.

In the next section the influence of the data types on the classification results is evaluated.

4.3.4.2 Different Data Types: Double, Int32, Int16, Int8

The features mean and zero crossings were calculated for different data types. For these evaluations a representative subset of the 27 data sets were used. For each computation the data sets were scaled in a way such as the variable range was optimal in terms of variable overflow and resolution.

For the same distance measure (only Euclidean or only Manhattan distance) there were no big deviations found in the metrics precision and recall for the 32-bit, 16-bit and 8-bit integer types (see Figure 4.18). Simulation results showed even slightly higher precision for the 8bit-integers.

When evaluating Manhattan distance for double data types (see subsection 4.3.4.1 above) no significant degradation was found. For integers bigger differences occur when Manhattan distance instead of Euclidean distance is used. A shift in weight can be recognised: precision decreases (-8%) and recall increases (+4%) for the data set used. Still precisions in the range of 80% can be reached for the used subset.

A shift in weight also occurs at the transition from double to integer number formats for the Manhattan distance measure: precision is decreased while recall is increased. Differences between double and integer data types might be caused by quantisation effects.

The results may also strongly depend on the chosen feature set. The feature mean can deviate by 1 at most if integers instead of doubles are used. Zero crossings already return an integer value and do not cause any deviations when using integers for calculations. It is expected that if other features are calculated, the deviations could be larger. Calculation of correlation uses squared sums for example. It thus more strongly limits the maximum values in preventing an overflow and reduces resolution of the results.

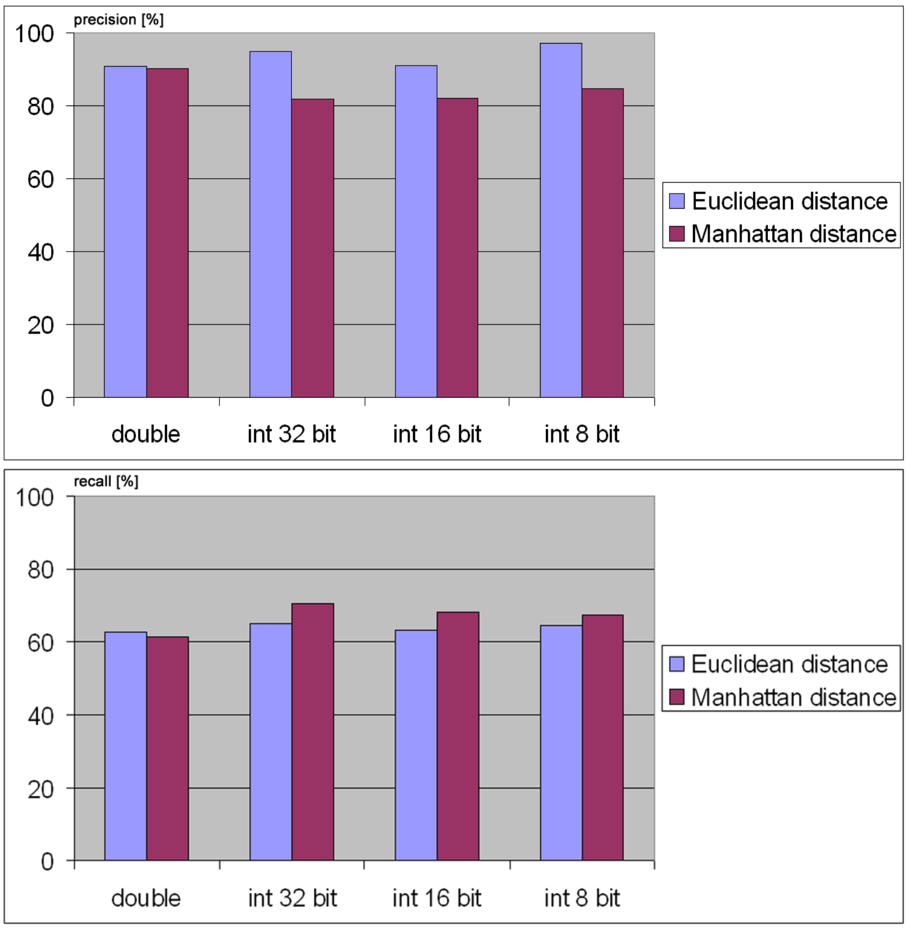


Figure 4.18: Data types: Double, int32, int16, int8. The simple feature set mean and zero crossings is analysed when calculated with different precision. Precision and recall are computed for the compensated data set. Top: The precisions of the different number formats are compared. Bottom: The recall is shown for the different number formats.

4.4 Conclusions from Activity Recognition

Our approach in activity recognition was to replace correlation by using distributed feature calculation and classification. Complex features and features with a rather high need for data exchange were replaced with a set of simple features that can be individually computed on the sensor nodes. This way, reductions in data rates of factor 2 up to 20, depending on parameters as window size, sampling frequency and package size, can be achieved.

Experiments were conducted to record shaking data. The Similarity Search classifier was successfully applied onto the data sets. A feature selection was performed to find feature sets that result in maximum performance. Using simple features only, mean and zero crossings together with a window size of 1s at a sampling frequency of 20Hz formed a optimal feature mix: an accuracy of 73.5%, a precision of 78.2% and a recall of 68.6% were achieved . If the set of simple features is extended by FFT features performance will only be slightly increased. Mean, FFT band 2 and FFT band 3 for a window size of 4s result in 76% accuracy, precision is raised 8% to 86% and recall is 64.4% compared to the simple feature set.

The performance of both feature sets surpasses that obtained for coherence but lies below of that obtained for correlation. Although correlation is more reliable in discriminating between correlated and uncorrelated shaking (accuracy of over 90%) it needs more radio communication, which increases power consumption and causes higher traffic in the network.

The system was tested for the case of confusing the classifier. Here, mean and zero crossings were more robust than the feature set with FFT features and showed better recognition of uncorrelated shaking by a single person that pretended to shake correlated. This difference in recognition could be due to additional phase information included in the zero crossings feature.

Tests with different subjects showed that shaking of the dice is a rather person-independent movement. Thus no extra adjustment to individuals is needed.

In implementation considerations it was found that Euclidean and Manhattan distance measures perform equally well for computations with double precision. Different data types (double, 32-bit integer, 16-bit integer, 8-bit integer) do not influence recognition rates much. The application of integers in combination with the Manhattan distance measure leads to a maximum loss in recognition: precision is lowered about 8%.
Nonetheless, activity recognition using 16-bit integers and Manhattan distance for computation can be realised on a microcontroller in real-time[7].

[7] [1] has presented reconfiguration times of less than 1ms for the Titan framework.

Chapter 5

Software Design

The main challenges of WSN design are the development of small and low-power systems which react to their environment in real-time. Software design for WSN devices must target these requirements. Solutions include:

- *Efficient usage of resources* Processing costs and memory should be saved while supporting low power consumption.
- *Concurrency* The system must react to events from the environment and has to handle many operations, such as data acquisition and processing, system control or radio communication, concurrently.
- *Adaptability* Modular software allows for adaptation to application specific requirements or hardware changes.
- *Simplicity* Eventually, software solutions should assist the programmer and simplify the design process.

This chapter introduces into software designed to operate networked embedded devices. Then the development of applications for *Titan*, the Tiny Task Network, is presented. The process of applying the Titan framework to a real activity recognition application is shown by implementing our demonstrator.

5.1 TinyOS

5.1.1 Operating System for Sensor Networks

Our hardware modules run the operating system TinyOS 2.0[1] [7, 8]. TinyOS is an event-based operating system especially designed for WSN applications. It covers most of the aforementioned requirements.

TinyOS is based on system components which are wired together. Components divide into higher and lower level components and build component layers organised

[1] http://www.tinyos.net

in three different levels of abstraction (Hardware Abstraction Architecture[2]). A component consists of a frame, command and event handlers, and tasks. The size of the frames is defined at compile time, i.e. memory is allocated statically. An advantage of the components is modularity. Just the components get included that are needed by an application, which results in a small memory footprint.

TinyOS provides concurrency via events and tasks. An event is either caused by a hardware interrupt due to reactions to environmental influences (e.g. sensed data, received messages) or by termination of a split-phase operation. Split-phase operations are operations of which start and end are separated. If a command is called, it requests the execution of an operation, then it returns. As soon as the operation is completed an event is signaled to the caller which gets the result. The concept of split-phase operations is intended for operations with longer latencies.
Components can post tasks. Tasks are synchronous and are executed by a FIFO scheduler. Tasks are non-preemptive with respect to other tasks and work in a "run-to-completion" manner. Tasks can post itself and are only preempted by events.
Tasks and events lead to an efficient and reactive system without the need for blocking operations or permanent sensor polling.

TinyOS is programmed in nesC [47], a programming language for networked embedded systems based on C. As nesC was designed to support the programming model of TinyOS, it perfectly addresses the specific design problems of WSN devices. [48] presents several software design patterns that reflect TinyOS's design goals.

nesC code is preprocessed by the nesC compiler. It performs optimisations (e.g. inlining of small functions) and converts the wiring of high level modules into efficient code. nesC output is a single C file which is subsequently compiled and linked by GCC for simulations, or by msp430-gcc for MSP430 microcontrollers respectively.
Compiled code is loaded onto MSP430 using the Flash Emulation Tool (FET) from TI and the JTAG adapter board [35]. Loading the code requires to establish a connection to MSP430 with the GNU debugger's MSP430-version msp430-gdb over gdbproxy to the FET. The necessary commands to load code onto MSP430 (or onto the Switch Module and Voice module respectively) are described in Appendix C.

5.1.2 Drivers and Applications

The Switch Module and the Voice Module represent two new architectures with specific code on the hardware adaptation layer. TinyOS 2.0 was ported to TelosB platform which uses a MSP430 microcontroller and a CC2420 radio. As Tmote Sky and Tmote Mini are basically similar to the Telos motes wide parts of the code, such as the radio stack, can be directly adopted from the TelosB platform. For the Switch Module the platform *tsoswitch* and for the Voice Module the platform *tsovoice* were built which include own drivers for sensors and actuators.

[2]see TEP 2 on http://www.tinyos.net

Drivers for the three sensors of the Switch Module and the Voice Module had to be programmed. The accelerometer and the light sensor are interfaced through the ADC. ADC drivers were programmed straightforward following TEP 109[3] and adapted for the three channels of the accelerometer. The pressure sensor is a digital sensor that provides a SPI interface. Here, the driver was mainly designed according to the sensor's data sheet.
The actuator drivers comprise the drivers for the analog switch and for the V-Stamp voice synthesiser. The driver for the analog switch is similar in parts to the LED driver already provided by the system. The design of the V-Stamp driver on the other hand needed more effort as it offers more functionality and implements a digital interface. The V-Stamp is connected to the microcontroller over an UART interface.
The drivers have all been further described in chapter 3 in the subsections of the respective hardware device.

Apart from the operating system's packages there is a further package in the TinyOS distribution that contains the applications. Code of the application package is based on components of the other packages (core TinyOS components, interfaces, drivers) and describes the actual application that should be executed on the motes. Examples for such applications are the test programs written for the hardware tests (see Appendix A) and for the power consumption measurements of the Switch Module and the Voice Module.

5.2 Titan

5.2.1 Titan "Tiny Task Network" Framework

Titan [1] is a framework designed with the objective of dynamic reprogramming WSNs. Its overall goal in dynamic reconfiguration is to run context recognition algorithms, to maintain the functionality of defective nodes or nodes with broken communication links or to balance variations in the network.
Titan's basic building blocks are tasks. Data gets exchanged through data streams flowing between these tasks. Tasks provide single operations, such as sensing or signal processing. Several tasks are connected to task graphs or to so-called task networks. Individual resources and availability of sensors and actuators on the physical nodes determine the final partition of the task network among the nodes.
Titan is divided into two parts: the code that runs on each of the wireless embedded nodes is based on TinyOS and implemented in nesC, the code to monitor and reconfigure the whole task network currently runs on the PC[4] and is written in Java. Both parts contribute to Titan's architecture and are essential for its full

[3]see TEP 109 on http://www.tinyos.net

[4]Titan's concept does not constrict that to be so necessarily. Software to monitor execution and dynamically reconfigure a task network might also be run directly on the motes (as C code) in the future.

functionality. An overview of Titan's architecture and its main modules is shown in Figure 5.1.

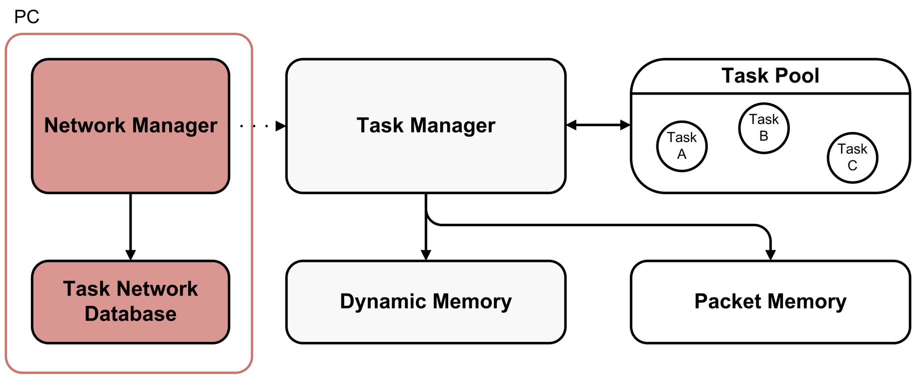


Figure 5.1: Titan architecture [1]. The arrows represent the direction of function calls.

- **Modules for network configuration and control (currently a Java implementation)** The *Network Manager* handles the execution of a task network. It compiles[5] the network using information from the Service Directory for the current network and (re)configures the network to run.
  The *Service Directory* stores information about the nodes available and notifies about changes in the network. It offers a passive or active mode: when set active, it periodically sends a service discovery message asking the network nodes for tasks that are available. In passive mode, the service directory reacts to incoming messages only.

- **Modules for node specific operations (TinyOS implementation)** Each node has a *Task Manager* and a local *Task Pool*. The Task Manager reorganises tasks, which are part of the overall task network, in a task graph (subnetwork) on the local node. During configuration tasks are taken from the Task Pool which contains all the tasks available on the node. The Task Manager handles the packets, the task context queues and the memory.
  Supporting *Dynamic Memory* means that a task allocates RAM memory for state variables and local buffers not until it is configured, i.e. as soon as the task is really needed at runtime. That way, more tasks can be used as scarce RAM (10kB RAM for the MSP430 microcontroller) would permit and a task can be instantiated multiple times.
  Packets sent and received by tasks are filed in shared FIFO queues and stored in *Packet Memory*.

[5]Current applications use a Greedy compiler for compilation. It adds tasks to a node until there is no space left, then it continues to add tasks to the next free node, and so on.

Titan offers several advantages compared to related approaches [4]:

- *Flexibility* Dynamic memory management reduces requirement for static memory. Dynamic reconfiguration allows to react on and adapt to diverse changes in a network.

- *Speed* Titan offers a trade-off between computational speed and dynamic reconfiguration time. When considering dynamic reconfiguration, a system's runtime and configuration time are important characteristics. Titan was analysed under these aspects in [1].

- *Portability and ease of use* As Titan builds on TinyOS it runs on different platforms and in heterogeneous networks. Applications to run on such networks can be set up intuitively by designing appropriate task graph configurations.

5.2.2 Titan-based Implementation

Titan is applied in the demonstrator. It acts as a control unit: Changing activities are recognised and the networked smart objects of the demonstrator get reprogrammed with the task graph of the next state. To make Titan perform this actions, tasks were newly implemented and a finite state machine (FSM) integrated into Titan.

5.2.2.1 Implementation of Tasks

The functionality of the nodes in a network is defined by the set of task in the local Task Pool. Each task defines a structure to save its state and configuration data. This context is loaded from the task context queue (FIFO queue) when the task executes. Execution is triggered by a packet available at the task's input port and the task enters its application specific section. Tasks usually process data streams but can operate on a single packet as well. If the task has at least one output port the output packet is sent to a second FIFO queue in Packet Memory.

TinyOS represents an efficient system as general basis. But Titan and its applications must continue with efficient designs. This includes the choice and implementation of algorithms, i.e. an efficient implementation of the tasks is intended. Therefore operations with lower computational costs are favoured (refer to chapter 4, section 4.3). Titan's tasks best use incremental algorithms which consecutively process the data streams in such a way as to avoid buffering much data. This leads to a higher CPU usage but saves RAM memory or allows for complexer operations (with more tasks on the same node), respectively. Execution is spread over longer time, thus no schedulability problem will occur when a window is finished.

In the following the characteristics of the implemented tasks are described in more detail. New tasks were created or existing tasks get enhanced. Tasks are programmed in nesC, in addition an interface must be provided for each task on the Java side to handle parameters that are passed to the tasks.

- **Sensor and actuator tasks**
  - *Accelerometer task* The Accelerometer task is designed as a wrapper. This way, it can be used for different types of acceleration sensors by including the appropriate accelerometer driver. The specific modules for our hardware platforms use the driver for the ADXL330 3-axis accelerometer and is configured with the number of samples per packet and the sample rate of the sensor.
    Multiple samples (each consisting of three 16-bit integer values for the X-, Y- and Z-coordinate) should be put into a single packet to reach maximum utilisation, i.e. up to 4 samples can be packed at a maximum packet size of 24 bytes. The downside of a packet with multiple samples is that the information about the time of sampling for the individual samples gets lost. If there occur significant time delays, due to lost packets for example, synchronisation of concurrently processed data will be affected.
    Just a single sample per packet is sent in the current application and the sensor is sampled at 10Hz.
  - *Voice task* The parameters of the Voice task consist of the volume, the voice type and the speed the voice synthesiser should operate at. Additionally, the selection of the task's behaviour or the operating state, respectively, must be set.
    The main challenge of integrating the voice synthesiser into a Titan task is proper initialisation. All tasks available on a node get initialised once when the operating system boots after code was loaded on the microcontroller. After that, the task is configured and shortly started. At the end of operation, e.g. before reconfiguration, the task is terminated and frees dynamic memory with its context. The Voice task requests the UART at initialisation. Configuration and application sections of the task can not use the voice synthesiser until the request is granted. Start and termination of the task only call functions that resume or suspend the voice synthesiser. The underlying driver is responsible that the voice synthesiser keeps speaking and works its buffer off even if the task network is meanwhile being reprogrammed.

There are some tasks which have a structure similar to the tasks described above. They also rely on the developed drivers and are only available on the respective nodes. Tasks *LightSensor* and *PressureSensor* also read out a sensor. PressureSensor faces the problem of correct initialisation. The *Switch* task adds a further actuator. Tasks *LightSensor* and *PressureSensor* also read out a sensor. PressureSensor faces the problem of correct initialisation. The *Switch* task adds a further actuator.

- **Data processing tasks**
  - *Magnitude task* The Magnitude task computes the squared magnitude of the input data. The number of coordinates for which the magnitude should be calculated and the number of calculated magnitudes packed into a single output message can be adjusted. The Magnitude task normalises the data. It uses an offset and a factor value for scaling (cf. section 4.1.1). As Magnitude reduces the raw data to a single magnitude value it is mostly used at a early stage in a task graph.
  - *Variance task* The Variance task computes the squared variance over a defined length of data using an incremental approach. Instead of calculating the variance only when needed at the task's output, variance gets continuously computed. Variance takes the two parameters window size and window shift. Several other data processing tasks, such as Sum or Mean, are organised similarly and also operate on those parameters. Window size specifies over how many data items the sliding window goes. Window shift sets the number of data items the sliding window is shifted before the next value is calculated. This way, overlapping windows can be generated.
    The Variance task is the critical task in terms of achievable (integer) precision of the system. During variance calculation the input data get multiplied and summed up. The trade-off consists of keeping integer values as large as possible to increase resolution while avoiding overflows of the sum of products.
  - *ZeroCrossings task* The ZeroCrossings task counts the number of crossings inside a sliding window. This number is incremented each time the input signal has completely crossed a predefined band around the average of the current window. The sliding window is continuously shifted according to the selected window shift. The band is specified by a lower and an upper threshold. The idea of this band is to remove the influence on the ZeroCrossings due to signal noise.
  - *Merge task* The Merge task merges packets coming in at several input ports and creates a new single output stream by concatenating the input values.
  - *Threshold task* The Threshold task does a thresholding of the input data. It accepts a single or multiple thresholds. If several data items are contained in a packet all get thresholded or binned, respectively, by the Threshold task.
  - *Fuzzy task* The Fuzzy task makes decisions in function of the input values. The precise decision rule must be adjusted for the specific application. The only parameter of the task selects the application and thus the corresponding "Fuzzy" function.

  - *SimilaritySearch task* The SimilaritySearch task describes a similarity search algorithm. The algorithm is specified in section 4.1.3 and ported to Titan by the implementation of this task. The Similarity Search classifier must be trained on meaningful training data. Once the parameters of the classifier, the centroids, the standard deviations as well as the radius, are estimated they are passed to the SimilaritySearch task.

  Further tasks that were implemented for feature calculation are: *Covariance*, *Mean* and *Sum*. Covariance mainly differs from the Variance task in having two inputs, Mean is similar to Sum and the Sum task is basically contained in parts of the Variance task.

- **Special tasks: the Communication task and the Sink task**

  - *Communication task* Unlike the other tasks the Communication task is instantiated by the Network Manager. It is automatically inserted into a task graph whenever the graph is partitioned into several subnetworks on different physical nodes. Hence each node has its own Communication task which is connected to the Communication tasks of those nodes that contain immediate successor or predecessor tasks.
    Communication tasks are found to be critical: they can lead to starvation of a task. If several tasks output packets to a Sink task radio can be occupied by the output of one task for the most time. As consequence data from other task outputs are not transmitted.
  - *Sink task* The Sink task is the task that completes a task graph. It must run on node 0, the node that provides the network managing capabilities. In our case, the PC runs the Network Manager and node 0 connects to the PC over USB.

5.2.2.2 From Tasks to Real Applications

Titan aims to discover changes in a WSN and to take the appropriate actions. Automatic reconfiguration of a network and active redistribution of tasks does need an intelligent control unit. Titan's network manager is supposed to dynamically adapt the task networks in a system autonomously.
Up to now, Titan has been operating in passive mode, i.e. no automatic reconfigurations were provided. The network manager only checked if the nodes were alive. Our current approach is a first step in direction of a reliable automatic system that monitors and reconfigures a WSN: a FSM is integrated in the Titan framework to control the network nodes.

The FSM receives data messages from the network over data communication interface. The messages are the events that trigger the FSM. Each message is evaluated according to the number of the input port of the Sink task at which the message is received, according to the message length and data contained in the message and according to the current state of the FSM. Depending on these values the FSM makes a decision. Either it remains in the current state or a state transition is invoked. At a state transition the FSM automatically initiate a reconfiguration of the network and a new task graph is loaded to the nodes.

The FSM provides a modular solution. Depending on the application a specific FSM can be inserted into Titan. Each time when the network is reprogrammed the FSM loads a new task graph to the nodes of the network, i.e. the functionality of each state is described by its own task graph. Task graphs can be defined in configuration files (ordinary text files), which are loaded through Titan's configuration reader.

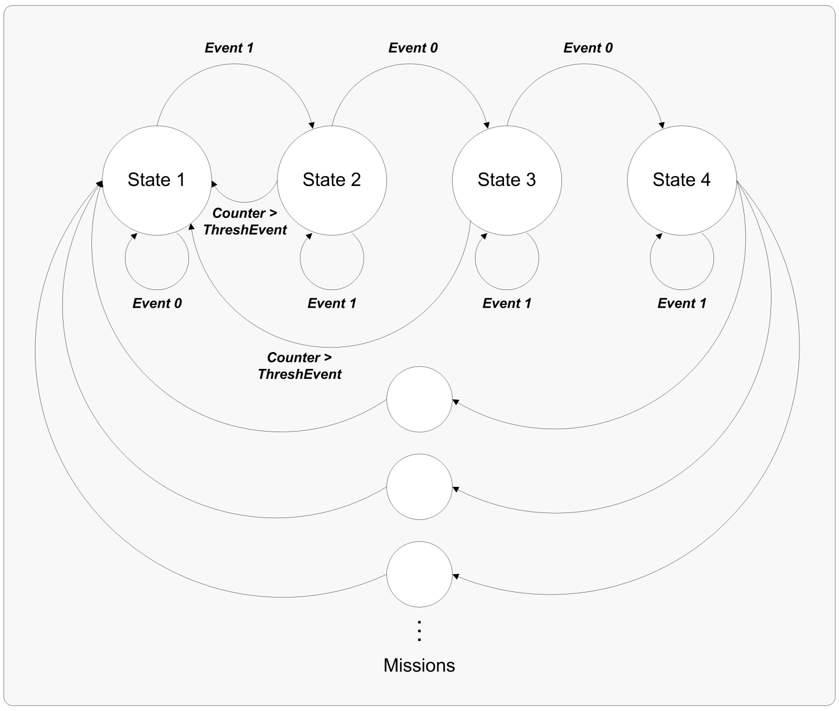


Figure 5.2: Finite State Machine. A detailed description of states 1 to 4 is provided by Figure 5.3. If in states 2 or 3 no activity is recognised for a certain time (for example, the number of incoming events can be counted during the timeouts) the state machine will return to the initial state. The states can implement the missions which have been proposed in section 2.2.

Titan and the FSM are used in our demonstrator setup to reconfigure the Switch Modules, which might be built in the foam dice for example, and the Voice Module. Figure 5.2 shows the design of the FSM for the demonstrator application. First the game is started and the subsequent course is determined by shaking and rolling the dice (states 1 to 4). Then one out of several missions could be selected due to the achieved score on the dice. When the mission is completed the game can start again.

Figure 5.3 presents the task graphs that are executed in the network during the first four states. Each task graph basically implements a recognition algorithm. If the network is reconfigured and a new task graph is loaded the recognition algorithm gets dynamically updated.
The first state contains the task graph "Foam Dice Listen". During this state all free dice wait until and listen for an action. As soon as exactly two dice get shaken the second task graph "Foam Dice Shake" is programmed. Here the discrimination between correlated and uncorrelated shaking is performed. When correlation in shaking the dice was detected by the classifier the player is informed to roll the dice and the third state is entered. In "Foam Dice Throw" the system waits until the dice remain unmoved. Then the next task graph "Foam Dice Score" is loaded which reads out the total score thrown. While designing the single task graphs again minimisation of network communication is an issue. Communication can be reduced by local activity aggregation, such that features rather than raw data are transmitted. Optimal distribution of the tasks on the nodes in the network can be achieved manually or by execution of an elaborate network compiler.

5.3 Conclusions from SW Design

A certain time was used to get familiar with the architecture of the Titan framework. Titan's basic concepts must be learned during the project.

Titan was extended by new tasks. Some tasks rely on the developed hardware and drivers and provide Titan with additional sensing and actuation. Other tasks are more directed to signal processing and activity recognition. A total of up to 15 tasks were extended or newly created.

A FSM was added to the Titan framework. It enables control of a system running Titan. Activities recognised on the single nodes of a network serve as events which drive the FSM. State transitions invoke dynamic reconfiguration by loading a new task graph automatically to the WSN nodes.

Titan was applied in a demonstrator setup. The system got started, nodes could be reconfigured and different task graphs loaded consecutively.

The main challenges of WSN, which were stated at the beginning of this chapter, could be approached. An efficient usage of resources was achieved by efficient implementations of the single tasks. In addition, the dynamic nature of Titan allows an efficient use of memory. Further, system's adaptability is provided by Titan, as adaptation to changes in the network and its environment is one of Titan's main functionalities. Concurrency was guaranteed by the underlying TinyOS operating system. Eventually, the concept of task graphs, which were loaded to the WSN nodes, provides an intuitive way of designing an application.

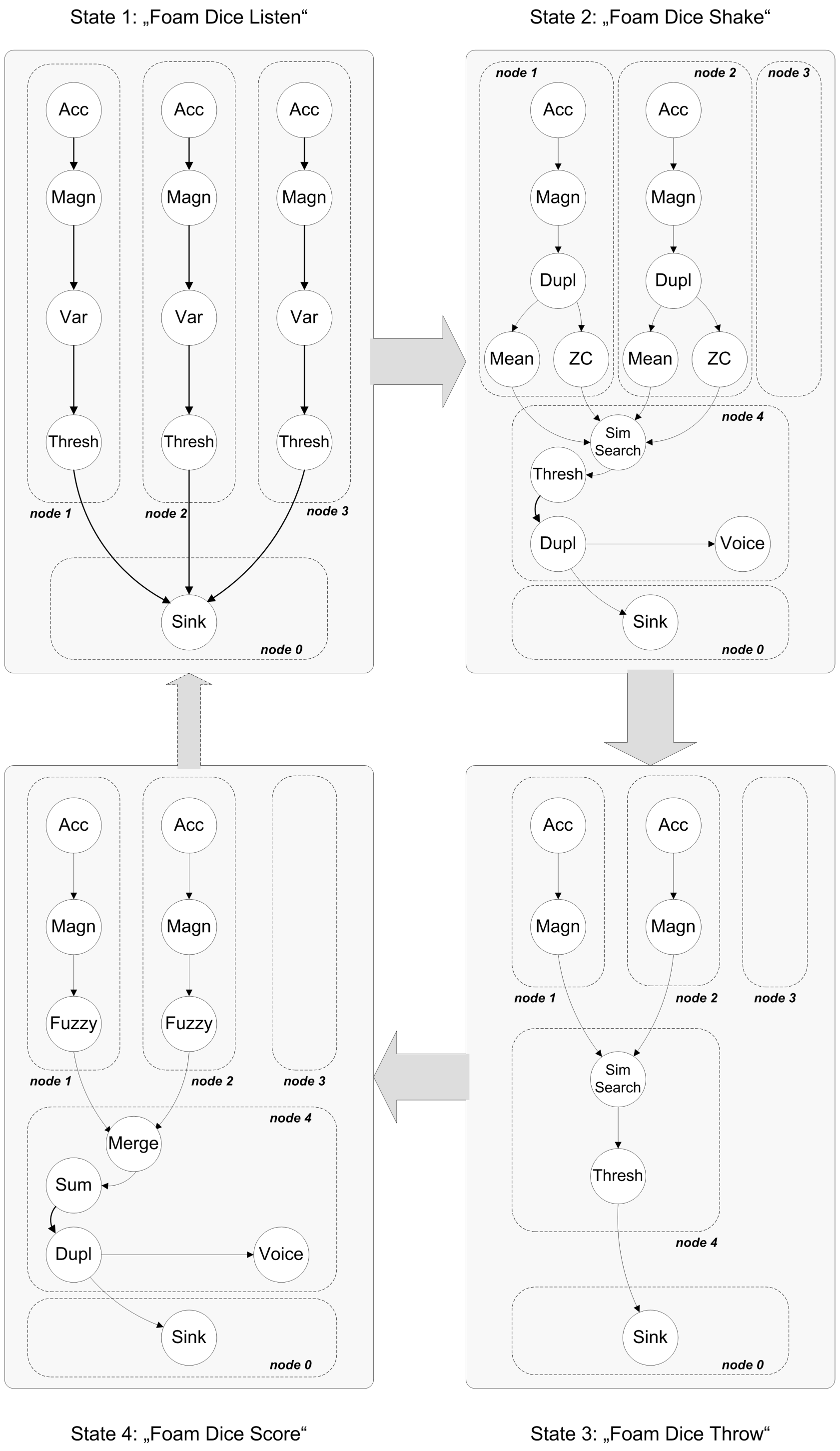


Figure 5.3: Task graphs for the implementation of the first 4 states. "Foam Dice Listen" waits until two dice are moved together. "Foam Dice Shake" detects a correlated shaking movement using the simple feature set with the features mean and zero crossings and the Similarity Search classifier. "Foam Dice Throw" detects the end of throwing the dice. "Foam Dice Score" calculates the total score. After state 4 has been executed a mission may be selected or the operation can restart with the initial state.

Chapter 6

Conclusion

In a first step scenarios for possible future applications in the WSN and activity recognition fields were worked out. Based on the state-of-the-art in these fields ideas were concentrated for a *demonstrator* that presents the operation of dynamic reconfiguration of WSN in combination with activity recognition.

With the *Switch Module* and the *Voice Module* two working hardware modules were built. The modules meet the requirements of WSN embedded devices: they provide efficient computation and communication while operating at low power to achieve a long lifetime. TinyOS 2.0 was ported to the modules and drivers have been written for the sensors and the actuators. The two modules provide Titan with a hardware platform including new tasks and form a base for further study of smart objects and activity recognition in an interactive real-time system for ambient intelligence.

The recognition of the activities “rolling” and “shaking” of a smart object was analysed in the context of WSN. The focus in evaluation lay in the discrimination of correlated and uncorrelated simultaneous shaking. The goal was to arrive at a reduction of computational costs and of costs for communication.
The reduction in the need for communication was achieved by using a set of simple features individually on each sensor node rather than calculating features from sensor data of several different nodes. Instead of computing a feature on the raw data several simple features were calculated before communication and only the features were transmitted over radio to the classifier. Reductions in data rates of 2 up to 20 depending on the efficiency of message handling can result. The reduction of communication costs enables power savings and prolongs the lifetime of the system.
Efficiency in computation was obtained by calculation of simple features that try to avoid multiplications and 32bit-operations as much as possible. In addition, the feature set is chosen to be as small as practicable, i.e features were only included to the set as long as a significant improvement in recognition rates was noticeable.

Evaluations showed that correlated and uncorrelated shaking can be distinguished by a Similarity Search classifier with an accuracy of 73.5% using a feature set that consists of only two feature pairs, mean and zero crossings, at a sampling rate of 20Hz and a window size of 1s. Introduction of FFT features did not significantly enhance the recognition rate. A set consisting of three feature pairs, the mean and two FFT bands, resulted in an accuracy of nearly 76%.

Titan was tested, developed further and applied in the demonstrator. Titan's task pool was extended with multiple tasks. Tasks to access the components of the hardware modules, tasks for signal processing, feature computation and classification were connected to tasks graphs that realise the activity recognition algorithm. Finally, a Finite State Machine was integrated into the Titan framework to dynamically reconfigure the task network for control of the demonstrator application.

This project provides an extensive study of smart objects and their application in wireless sensor networks and activity recognition. Initially, ideas for scenarios and future applications based on smart objects and activity recognition were evaluated. Then, two prototypes of networked embedded devices were built providing for efficient operation and low power consumption. Activity recognition of shaking movements performed with smart objects was analysed. Finally, the recognition algorithms were implemented in the Titan framework on the smart objects. The resulting system allows for online activity recognition and enables continuous adaptation to changes in activities by dynamic reconfiguration.

Chapter 7

Outlook

7.1 Suggestions for Improvement

Hardware Design

- *Boost Converter TPS61202* The boost converter included in the current Voice Module does not work properly. The boost converter is not elementary for a functional system because the V-Stamp's audio system can be driven at lower voltages. As a loss in audio quality is assumed by the lower supply voltage, the boost converter circuit needs particular consideration in case of a redesign of the Voice Module.

- *LEDs* The LEDs on the Switch Module and Voice Module are connected to the Tmote Mini and to GND. Thus the LEDs are normally switched on, which results in a negative logic. Currently the methods of the LED-interface are applied inversely. The problem can be fixed by changing the definitions in the LED driver of the TelosB platform. In the case of a new layout, the LED should be connected to VCC (instead of GND).

- *Power Management* The Switch and the Voice Module both have been designed to achieve low power consumption. System components were selected, power switching and battery power monitoring were included with this objective. MSP430 microcontroller has several power states. TinyOS can manage the power states of the microcontroller and of peripheral devices. Implementation of power down modes for the modules by using TinyOS's power management capabilities[1] would be the next important step.

- *Pressure Sensor MS5540B* The air pressure sensor is interfaced by an unusual SPI interface, which complicates calibration of the sensor. Code for sensor calibration is provided in the driver but not yet fully tested.

[1]see TEP 108, TEP 112, TEP 115 on http://www.tinyos.net

Activity Recognition

- *Evaluation* MIFS only provided tendencies of promising feature sets. Alternative feature selection methods could result in better performance.
  In dealing with recognition, overfitting of the classifier is always an issue. Evaluations (especially the training of the classifier) could be extended to larger data sets to reduce the risk of overfitting.

- *Classification* In the project we focused on binary decisions, having a NULL class and one further class "Correlated Shaking". It would be interesting to extend the set of classes, such that several activities, such as shaking, throwing, rolling, carrying the dice, or putting the dice to a wardrobe, could be detected. For classification an extension to a more powerful classifier, such as linear discriminant analysis (LDA) [49], could improve the recognition results.

- *Sensor Fusion* Evaluation of activity recognition could be extended to the other sensors on the module. Sensor fusion of the acceleration, light and pressure sensor data or a combination of multiple accelerometers could be analysed.

Software Design

- *Number formats* Titan's tasks work on data streams. Incoming and outgoing data have their own format, e.g. a 16-bit unsigned integer. At the moment each task works with the data format that fits best its application and sets the data type of output data when sending a new message. Some tasks already check incoming packets for different data types. The number format used by the tasks must be standardised to make the system more reliable and user-friendly. Another approach would be to handle different data types of incoming data in every task. This way the application-specific part of a single task could be implemented independent from the data type used by other tasks.

- *COM Task* The COM task has been a bottleneck. It can cause starvation, i.e. certain tasks do not get their packets whereas others are continuously transmitting.

- *Time Synchronisation* To avoid synchronisation problems when processing data from multiple nodes, a timestamp could be included into Titan's packets. Although adding an overhead accuracy of recognition algorithms could be considerably improved. However, a time synchronisation protocol must be introduced into the network.

7.2 Future Work

Titan's goal is to provide a full service discovery. This includes the detection of failing nodes, recompilation and reconfiguration of the task network to adapt the system. Adaptations could serve for optimisation, e.g. if a node states low battery power Titan will reduce the set of tasks running on that node or exchange the tasks by less critical ones for the case the node will fail. New concepts in partition algorithms could enable a more sophisticated distribution of tasks on the available nodes in the network. Thus tasks are distributed more equally and the network is more balanced, i.e. a node will never work to full capacity while another remains idle.
If a PC is connected to a network it could be interesting to investigate the partition between SW and HW, i.e. which tasks should be performed on the nodes and which on the PC.
On the other hand, further development could aim at the complete integration of Titan on embedded devices in the network. If service discovery is also performed directly on the WSN nodes (assumed that certain nodes in the network provide enough resources) the system running Titan becomes completely real-time, which is beneficial to provide for human interaction for example.

Another problem related to dynamic reconfiguration of WSN is the proper handling of multiple network managers in the system. To determine which nodes belong together information of grouped objects could be used. Further analysis of grouping of objects can offer methods to assign nodes to different network managers and to establish subnetworks with common behaviour.

The combination of smart objects and wearable systems is another field for future work. Wearable systems can take advantage of information provided by smart objects integrated in the environment. Additional context information can simplify the recognition task. Furthermore, smart objects can temporarily become part of Body Area Networks which then may be updated by the context of the smart objects. An example for such a smart object is Titan running on a mobile phone.

In the context of wearable systems, the voice synthesiser module might be an interesting feedback device which could be tested in further applications. Interaction by voice offers an alternative to displays and would especially be appropriate for wearable systems.

Acknowledgement

I would like to express my appreciation to my supervisor C. Lombriser whose advice and weekly discussions made this project possible. He was a great help in solving problems related to the Titan framework. I would like to acknowledge U. Egger for his support, especially for giving an introduction into PCB design with Altium Designer. His know-how in hardware design assisted me greatly.

Bibliography

[1] C. Lombriser, D. Roggen, M. Stäger, and G. Tröster. Titan: A tiny task network for dynamically reconfigurable heterogeneous sensor networks. In *15. Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, pages 127–138, 2007.

[2] P.J. Marron, D. Minder, and the Embedded WiSeNts Consortium. *Embedded WiSeNts Research Roadmap*. Logos Verlag Berlin, November 2006.

[3] H. Junker, P. Lukowicz, and G. Tröster. Continuous recognition of arm activities with body-worn inertial sensors. In *Proceedings of the Eighth International Symposium on Wearable Computers (ISWC'04)*, pages 188–189, 2004.

[4] C. Lombriser, N.B. Bharatula, D. Roggen, and G. Tröster. On-body activity recognition in a dynamic sensor network. In *2nd International Conference on Body Area Networks (BodyNets)*, 2007.

[5] T. Stiefmeier, H. Junker G. Ogris, P. Lukowicz, and G. Tröster. Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. In *10th IEEE International Symposium on Wearable Computers (ISWC)*, 2006.

[6] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ACM SIGPLAN Notices*, volume 35, pages 93–104, November 2000.

[8] P. Levis. *TinyOS Programming*, October 2006.

[9] J.M. Kahn, R.H. Katz, and K.S.J. Pister. Emerging challenges: Mobile networking for t't'smart dust". Technical report, University of California, Berkeley, USA, 2000.

[10] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA'02*, pages 88–97, 2002.

[11] K. Martinez, A. Riddoch, J. Hart, and R. Ong. Glacsweb: A sensor web for glaciers. In *EWSN 2004*, 2004.

[12] R. Beckwith, D. Teibel, and P. Bowen. Pervasive computing and proactive agriculture. In *PERVASIVE 2004*, 2004.

[13] F. Michahelles, P. Matter, A. Schmidt, and B. Schiele. Applying wearable sensors to avalanche rescue. *Computers and Graphics*, 27(6):839–847, 2003.

[14] H. Baldus, K. Klabunde, and G. Müsch. Reliable set-up of medical body-sensor networks. In *EWSN 2004*, 2004.

[15] M. Beigl, H. W. Gellersen, and A. Schmidt. Mediacups: Experience with design and use of computer-augmented everyday artefacts. *Computer Networks*, 35(4):401–409, 2001.

[16] D. Roggen, N.B. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. From sensors to miniature networked sensorbuttons. In *Proc. of the 3rd Int. Conf. on Networked Sensing Systems (INSS06)*, 2006.

[17] S.S. Intille, K. Larson, E.M. Tapia, J.S. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson. Using a live-in laboratory for ubiquitous computing research. In *Proceedings of PERVASIVE 2006*, pages 349–365, 2006.

[18] L.E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Third International Conference of Ubiquitous Computing (Ubicomp)*, page 116, 2001.

[19] J. Lester, B. Hannaford, and G. Borriello. Are you with me? – using accelerometers to determine if two devices are carried by the same person. In *2nd International Conference on Pervasive Computing (Pervasive)*, pages 33–50, 2004.

[20] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *4th International Conference on Pervasive Computing (Pervasive)*, pages 144–161, 2007.

[21] R. Marin-Perianu, M. Marin-Perianu, P. Havinga, and H. Scholten. Movement-based group awareness with wireless sensor networks. Technical report, University of Twente, Enschede, The Netherlands, 2006.

[22] O. Amft, C. Lombriser, T. Stiefmeier, and G. Tröster. Recognition of user activity sequences using distributed event detection. In *European Conference on Smart Sensing and Context (EuroSSC)*, 2007.

[23] D. Bannach, O. Amft, K.S. Kunze, E.A. Heinz, G. Tröster, and P. Lukowicz. Waving real hand gestures recorded by wearable motion sensors to a virtual

car and driver in a mixed-reality parking game. In *Computational Intelligence and Games (CIG)*, pages 32–39, 2007.

[24] L. Bao and S.S. Intille. Activity recognition from user-annotated acceleration data. In *PERVASIVE 2004*, volume 3001, pages 1–17, April 2004.

[25] T. Marrin. Possibilities for the digital baton as a general-purpose gestural interface. Technical report, MIT Media Laboratory, Cambridge, USA, 1997.

[26] A. Wilson and S. Shafer. Xwand: Ui for intelligent spaces. *letters chi*, 2003.

[27] M. P. Johnson, A. Wilson, B. Blumberg, C. Kline, and A. Bobick. Sympathetic interfaces: Using a plush toy to direct synthetic characters. *CHI 99*, 1999.

[28] R. Crepaldi, A. F. Harris, R. Cooper, R. Kravets, G. Maselli, C. Petrioli, and M. Zorzi. Managing heterogeneous sensors and actuators in ubiquitious computing environments. In *SANET 07*, pages 35–42, 2007.

[29] B. Brumitt and J.J. Cadiz. Let there be light: Examining interfaces for homes of the future. In *Human-Computer Interaction: INTERACT‘01*, 2001.

[30] R. Aylward, S.D. Lovell, and J.A. Paradiso. A compact, wireless, wearable sensor network for interactive dance ensembles. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN‘06).*

[31] F. Schenkel. Wireless gateway, 2007.

[32] E.A. Heinz, K. Kunze, S.Sulistyo, H. Junker, P. Lukowicz, and G. Tröster. Experimental evaluation of variations in primary features used for accelerometric context recognition. In *Ambient Intelligence: First European Symposium, EUSAI 2003*, volume 2875, pages 252–263, October 2003.

[33] O. Amft, H. Junker, and G. Tröster. Detection of eating and drinking arm gestures using inertial body-worn sensors. In *Proceedings of the 2005 Ninth IEEE International Symposium on Wearable Computers*, pages 160–163, October 2005.

[34] T. Huynh and B. Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*, volume 121, pages 159–163, October 2005.

[35] M. Graf. Wearable sensor package 3. Master’s thesis, ETH Zürich, 2005.

[36] M. Strohbach and H. Gellersen. Smart clustering – networking smart objects based on their physical relationships. In *5th IEEE International Workshop on Networked Appliances*, pages 151–155, 2002.

[37] D. Bannach, K. Kunze, P. Lukowicz, and O. Amft. Distributed modular toolbox for multi-modal context recognition. In *Proceedings of the 19th International Conference on Architecture of Computing Systems (ARCS)*, pages 99–113, 2006.

[38] G.C. Carter. Coherence and time delay estimation. In *Proceedings of the IEEE*, volume 75, pages 236–255, February 1987.

[39] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, 1994.

[40] D. Christen. Feature selection for body-worn sensors, 2007.

[41] O. Beucher. *Wahrscheinlichkeitsrechnung und Statistik mit MATLAB: Anwendungsorientierte Einführung für Ingenieure und Naturwissenschaftler.* Springer, 1st edition edition, 2005.

[42] W.H. Kruskal and W.A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47:583–610, 1952.

[43] C.J. Veenman and M.J.T. Reinders. The nearest sub-class classifier: a compromise between the nearest mean and nearest neighbor classifier. *IEEE Transactions on PAMI*, 27(9):1417–1429, 2005.

[44] W. Li. Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60(5/6):823–837, 1990.

[45] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *Proceedings of Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pages 147–163, 2002.

[46] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pages 138–149, 2003.

[47] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1–11, June 2003.

[48] D. Gay, P. Levis, and D. Culler. Software design patterns for tinyos. In *ACM Transactions on Embedded Computing Systems (TECS)*, volume 6, pages 93–104, September 2007.

[49] P. Lukowicz, J. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. Recognizing workshop activity using body worn microphones and accelerometers. In *Second International Conference, PERVASIVE 2004*, pages 18–32, April 2004.

Appendix A

HW Design

A.1 Test Protocols

When running the hardware tests for the modules the system can be powered from a power supply (and not from battery).

A.1.1 Switch Module

1. PCB:
   - Preparation work: cut PCBs
   - Test: visual inspection, check diameter of drill holes
2. Power Supply:
   - Preparation work: prepare cables/wires, mount power connector and Schottky diode D2
   - Test: check connections
3. Tmote Mini:
   - Preparation work: solder Tmote Mini module and JTAG connector
   - Test: check connections, measure voltage levels
   - Result: Module should be seen from JTAG connector, voltage levels at connector must be correct (GND, VCC, ...)
4. LED:
   - Preparation work: solder LED D1 and R1
   - Test: check connections, measure voltage levels, load and run test program "BlinkingLED" (LED is switched on/off)
5. Actuator "Switch":
   - Preparation work: solder switch and R6
   - Test: check connections, measure voltage levels, load and run test program "BlinkingCandle" (external circuit is driven and LEDs of the lamp is switched on/off)
   - Result: find the highest switching frequency possible
6. Chip antenna:
   - Preparation work: solder antenna
   - Test: check connections, measure voltage levels, load and run test program "SendingHelloWorld" (TmoteSky Module receives Messages and forwards them over the serial port; the messages are displayed on the terminal)
7. Sensor "Battery Level":
   - Preparation work: solder R2 and R3

- Test: check connections, measure voltage levels, load and run test program “BatteryCheck” (LED displays if battery is connected or read out the battery level and let it display)

8. Sensor “Light”:

   - Preparation work: solder APDS9003, C6, C7 and R5
   - Test: check connections, measure voltage levels, load and run test program “SensingLight” (LED displays if it’s light or dark, or read out the brightness and let it display)
   - Result: light sensor is calibrated (measure in the dark, measure direct below the light), noise level measured

9. Sensor “Acceleration”:

   - Preparation work: solder ADXL330, C1, C2, C3 and C5
   - Test: check connections, measure voltage levels, load and run test program “SensingAcceleration” (read out the values of the 3 axes and let it display)
   - Result: acceleration sensor is calibrated (scale using acceleration of gravity, or known constant acceleration, e.g. fast elevator), noise level measured

10. Sensor “Pressure”:

    - Preparation work: solder MS5540B, C4 and R4
    - Test: check connections, measure voltage levels, load and run test program “SensingPressure” (read out the temperature and pressure values and let them display)
    - Result: pressure sensor calibrated (scale with barometric pressure values from a weather station), noise level measured

A.1.2 Voice Module

1. PCB:
   - Preparation work: cut PCBs
   - Test: visual inspection, check diameter of drill holes
2. Power Supply (1):
   - Preparation work: prepare cables/wires, mount power connector, Schottky diode D3
   - Test: check connections
3. Power Supply (2):
   - Preparation work: mount LDO-regulator and solder C10 and C11
   - Test: check connections, measure voltage level
   - Results: known input voltage range, output level of 3.3V (or 3V), known input and output current range
4. Power Supply (3)[1]:
   - Preparation work: mount boost converter and solder L1, C12, C13, C14 and R18, R19
   - Test: check connections, measure voltage level
   - Results: known input voltage range, output level of 5V, known input and output current range, power dissipation, noise
5. Tmote Mini:
   - Preparation work: solder Tmote Mini module and JTAG connector
   - Test: check connections, measure voltage levels
   - Result: Module should be seen from JTAG connector, voltage levels at connector must be correct (GND, VCC, . . .)
6. LED:
   - Preparation work: solder LEDs D1, D2 and R1, R2
   - Test: check connections, measure voltage levels, load and run test program “BlinkingLED” (LEDs are switched on/off)
7. Chip antenna:

[1]This step must be replaced due to malfunction of the boost converter. Instead of *4. Power Supply (3)* the boost converter is omitted and bridged with a wire directly connecting the input voltage from the pad at C12 to TPS61202’s output pin at pad C13. This way, a voltage of 3.7V is applied to the input pin V_{PA} of the V-Stamp.

- Preparation work: solder antenna
- Test: check connections, measure voltage levels, load and run test program "SendingHelloWorld" (TmoteSky Module receives Messages and forwards them over the serial port; the messages are displayed on the terminal)

8. Sensor "Battery Level":
   - Preparation work: solder R3 and R4
   - Test: check connections, measure voltage levels, load and run test program "BatteryCheck" (LED displays if battery is connected or read out the battery level and let it display)
9. Sensor "Light":
   - Preparation work: solder APDS9003, C6, C7 and R6
   - Test: check connections, measure voltage levels, load and run test program "SensingLight" (LED displays if it's light or dark, or read out the brightness and let it display)
   - Result: light sensor is calibrated (measure in the dark, measure direct below the light), noise level measured
10. Sensor "Acceleration":
    - Preparation work: solder ADXL330, C1, C2, C3 and C5
    - Test: check connections, measure voltage levels, load and run test program "SensingAcceleration" (read out the values of the 3 axes and let it display)
    - Result: acceleration sensor is calibrated (scale using acceleration of gravity, or known constant acceleration, e.g. fast elevator), noise level measured
11. Sensor "Pressure":
    - Preparation work: solder MS5540B, C4 and R5
    - Test: check connections, measure voltage levels, load and run test program "SensingPressure" (read out the temperature and pressure values and let them display)
    - Result: pressure sensor calibrated (scale with barometric pressure values from a weather station), noise level measured
12. Actuator "V-Stamp":
    - Preparation work: first solder C8, C9 and R7, . . . , R17; then solder the phone jack, finally solder the multipoint connectors

- Test: check connections, measure voltage levels, load and run test program “playSound” (text is sent to the mote in a package and converted to speech, a piece of music is stored on the V-Stamp and the signal is set to play that music)

A.2 Schematics

A.2.1 Schematics of the Switch Module

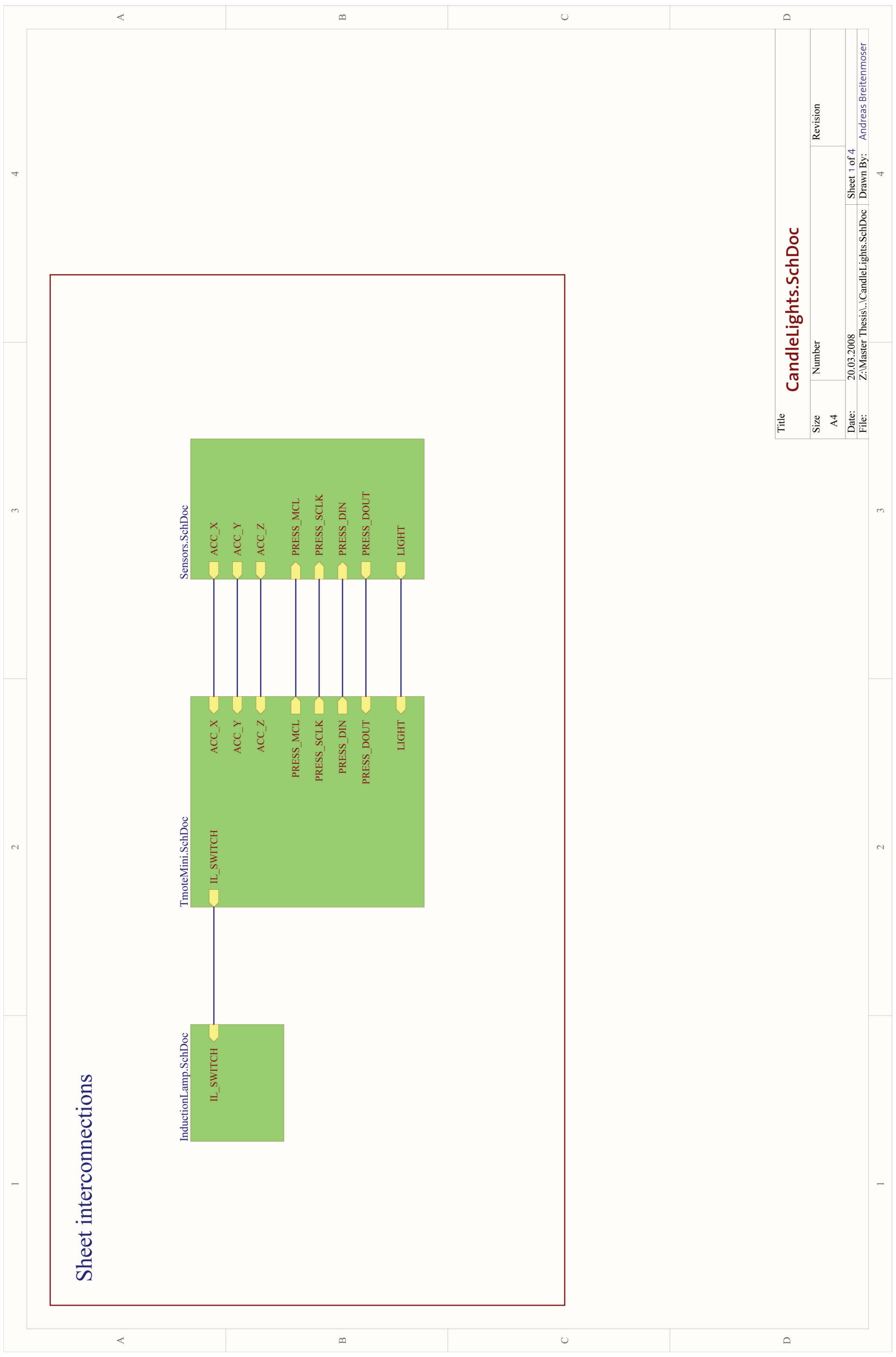
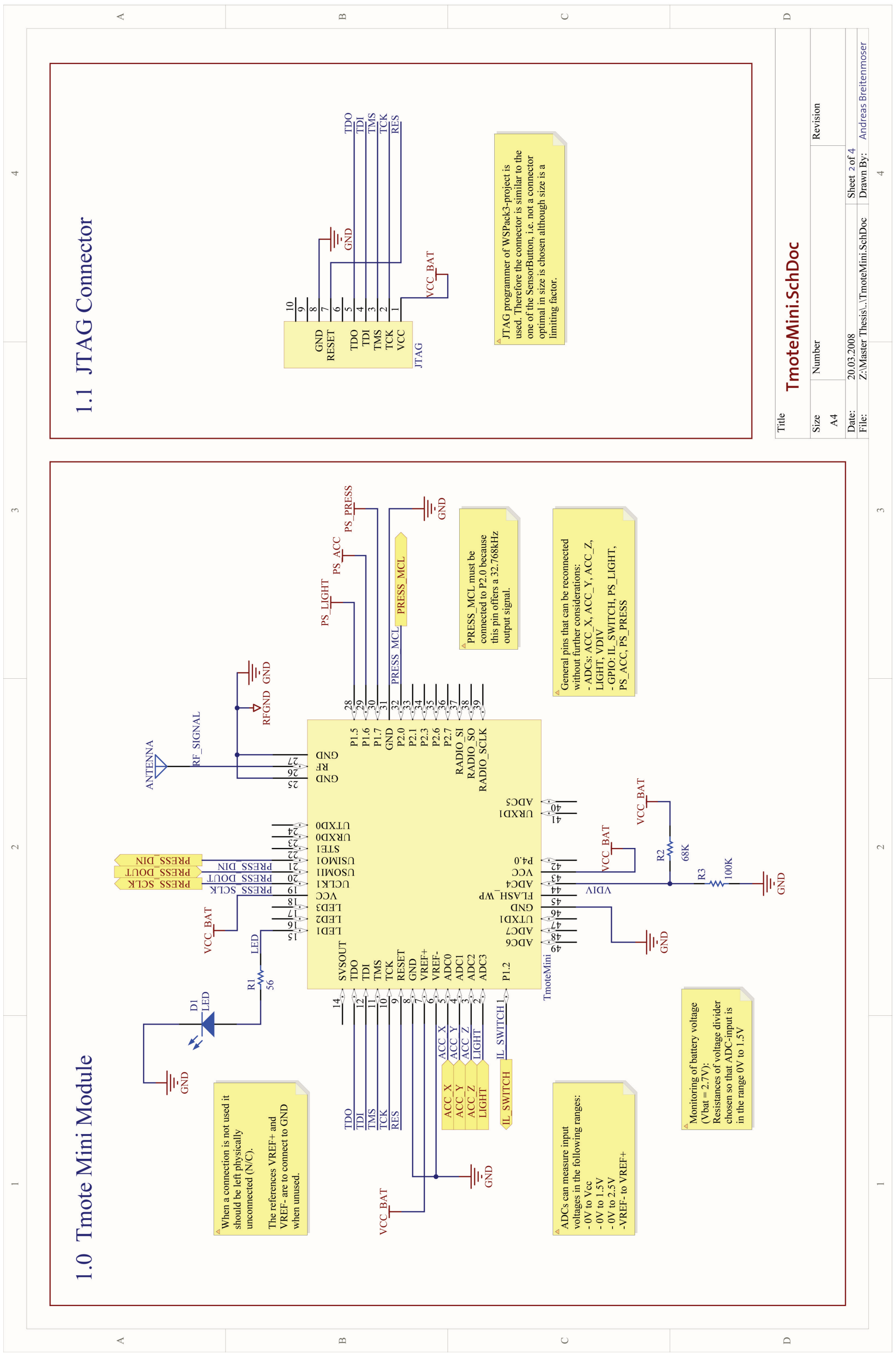


Figure A.1: Switch Module: sheet interconnections.



JTAG programmer of WSPack3-project is used. Therefore the connector is similar to the one of the SensorButton, i.e. not a connector optimal in size is chosen although size is a limiting factor.

PRESS_MCL must be connected to P2.0 because this pin offers a 32.768kHz output signal.

General pins that can be reconnected without further considerations:
- ADCs: ACC_X, ACC_Y, ACC_Z, LIGHT, VDIV
- GPIO: IL_SWITCH, PS_LIGHT, PS_ACC, PS_PRESS

When a connection is not used it should be left physically unconnected (N/C).

The references VREF+ and VREF- are to connect to GND when unused.

ADCs can measure input voltages in the following ranges:
- 0V to Vcc
- 0V to 1.5V
- 0V to 2.5V
- VREF- to VREF+

Monitoring of battery voltage (Vbat = 2.7V):
Resistances of voltage divider chosen so that ADC-input is in the range 0V to 1.5V

Figure A.2: Switch Module: Tmote Mini.

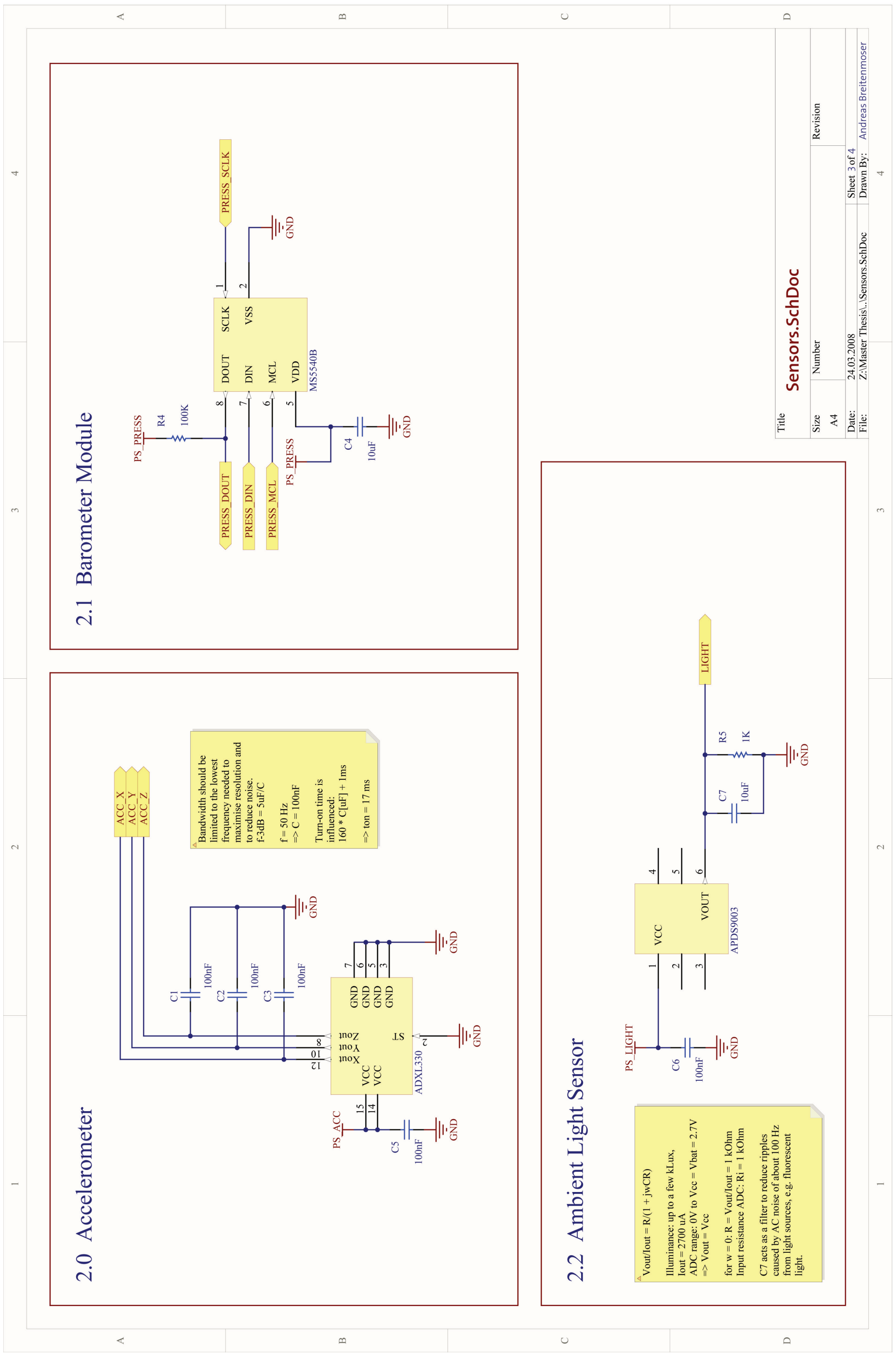
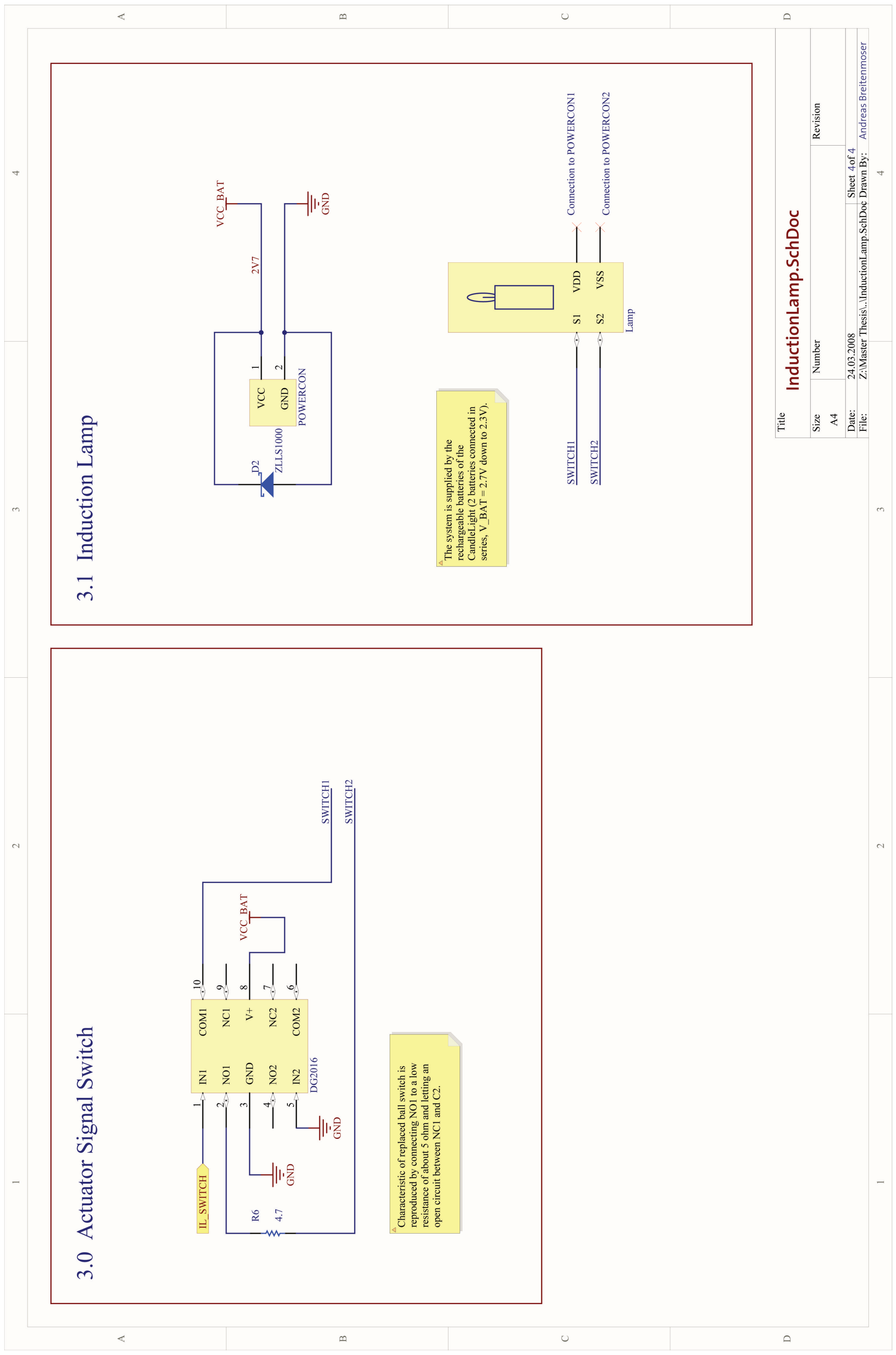

2.0 Accelerometer
2.1 Barometer Module
2.2 Ambient Light Sensor
Sensors.SchDoc
Title
Size A4
Number
Revision
Date: 24.03.2008
File: Z:\Master Thesis\..\Sensors.SchDoc
Sheet 3 of 4
Drawn By: Andreas Breitenmoser
MS5540B
SCLK
VSS
DOUT
DIN
MCL
VDD
PRESS_SCLK
PRESS_DOUT
PRESS_DIN
PRESS_MCL
PS_PRESS
R4 100K
C4 10uF
GND
ADXL330
Xout
Yout
Zout
ST
VCC
GND
ACC_X
ACC_Y
ACC_Z
PS_ACC
C1 100nF
C2 100nF
C3 100nF
C5 100nF
Bandwidth should be limited to the lowest frequency needed to maximise resolution and to reduce noise.
f-3dB = 5uF/C
f = 50 Hz
=> C = 100nF
Turn-on time is influenced:
160 * C[uF] + 1ms
=> ton = 17 ms
APDS9003
VCC
VOUT
LIGHT
PS_LIGHT
R5 1K
C6 100nF
C7 10uF
Vout/Iout = R/(1 + jwCR)
Illuminance: up to a few kLux, Iout = 2700 uA
ADC range: 0V to Vcc = Vbat = 2.7V
=> Vout = Vcc
for w = 0: R = Vout/Iout = 1 kOhm
Input resistance ADC: Ri = 1 kOhm
C7 acts as a filter to reduce ripples caused by AC noise of about 100 Hz from light sources, e.g. fluorescent light.

Figure A.3: Switch Module: sensors.



The system is supplied by the rechargeable batteries of the CandleLight (2 batteries connected in series, V_BAT = 2.7V down to 2.3V).

Characteristic of replaced ball switch is reproduced by connecting NO1 to a low resistance of about 5 ohm and letting an open circuit between NC1 and C2.

Figure A.4: Switch Module: actuator.

A.2.2 Schematics of the Voice Module

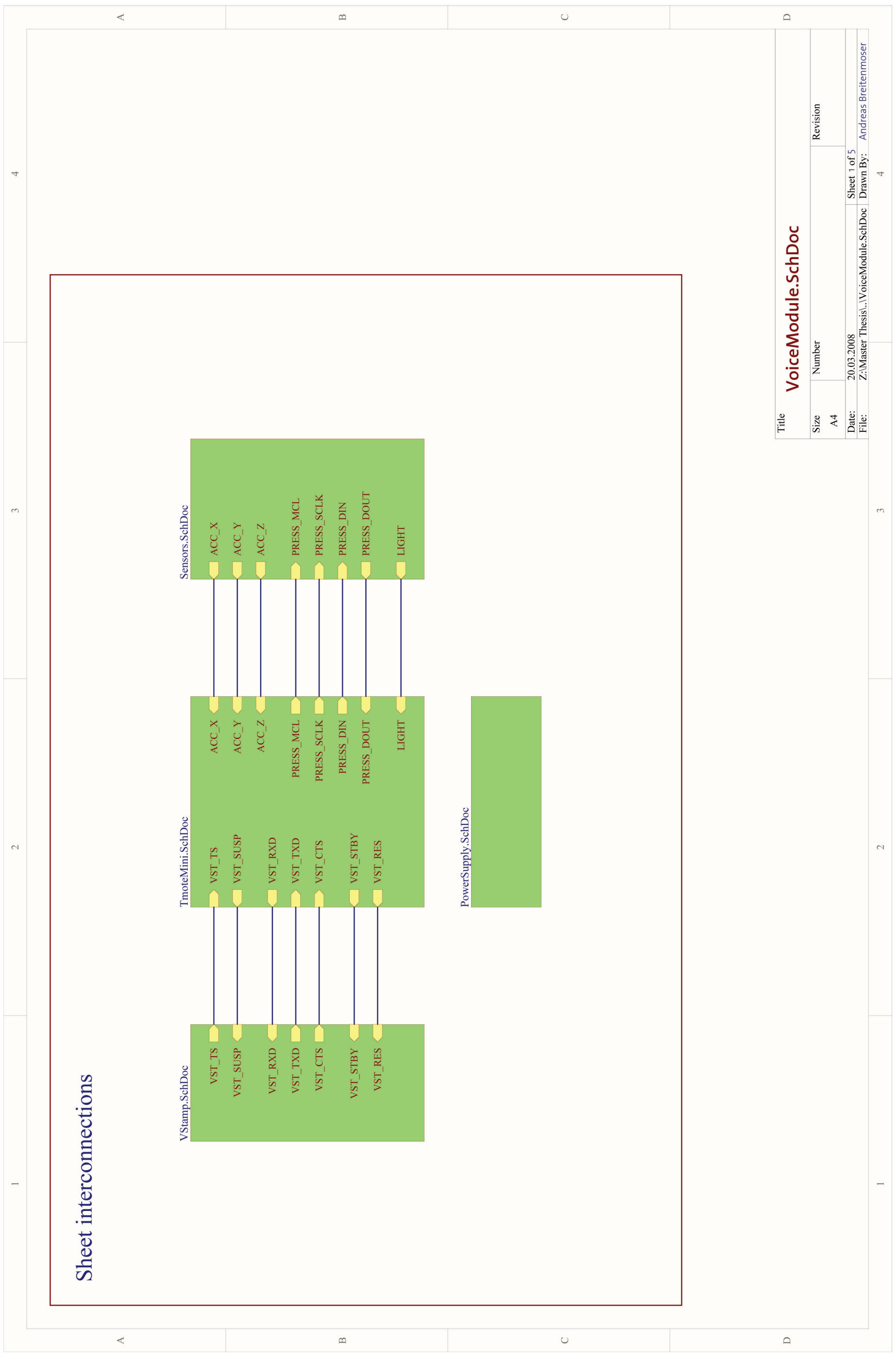
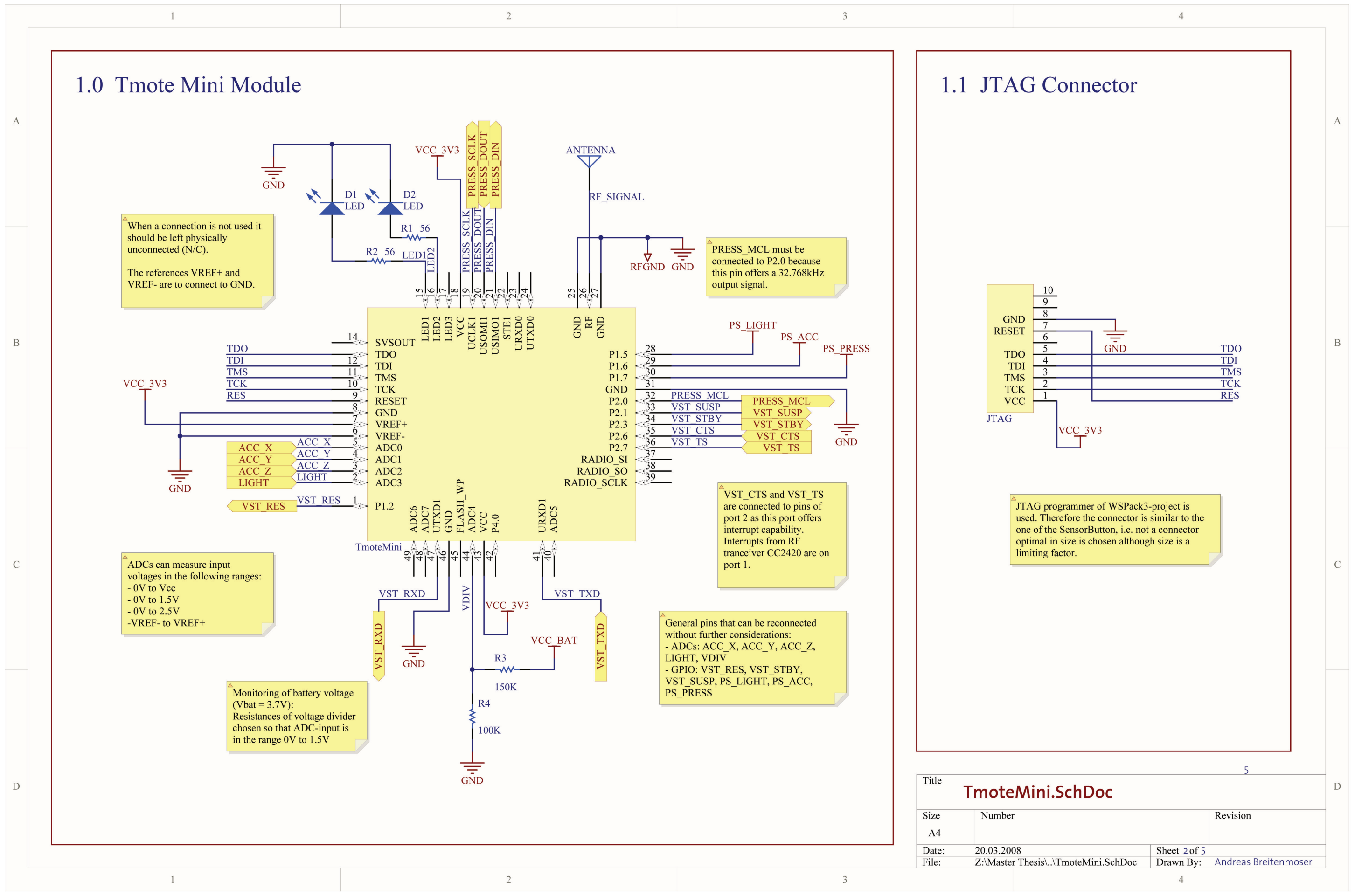


Figure A.5: Voice Module: sheet interconnections.



When a connection is not used it should be left physically unconnected (N/C).

The references VREF+ and VREF- are to connect to GND.

ADCs can measure input voltages in the following ranges:
- 0V to Vcc
- 0V to 1.5V
- 0V to 2.5V
-VREF- to VREF+

Monitoring of battery voltage (Vbat = 3.7V):
Resistances of voltage divider chosen so that ADC-input is in the range 0V to 1.5V

PRESS_MCL must be connected to P2.0 because this pin offers a 32.768kHz output signal.

VST_CTS and VST_TS are connected to pins of port 2 as this port offers interrupt capability. Interrupts from RF tranceiver CC2420 are on port 1.

General pins that can be reconnected without further considerations:
- ADCs: ACC_X, ACC_Y, ACC_Z, LIGHT, VDIV
- GPIO: VST_RES, VST_STBY, VST_SUSP, PS_LIGHT, PS_ACC, PS_PRESS

JTAG programmer of WSPack3-project is used. Therefore the connector is similar to the one of the SensorButton, i.e. not a connector optimal in size is chosen although size is a limiting factor.

Figure A.6: Voice Module: Tmote Mini.

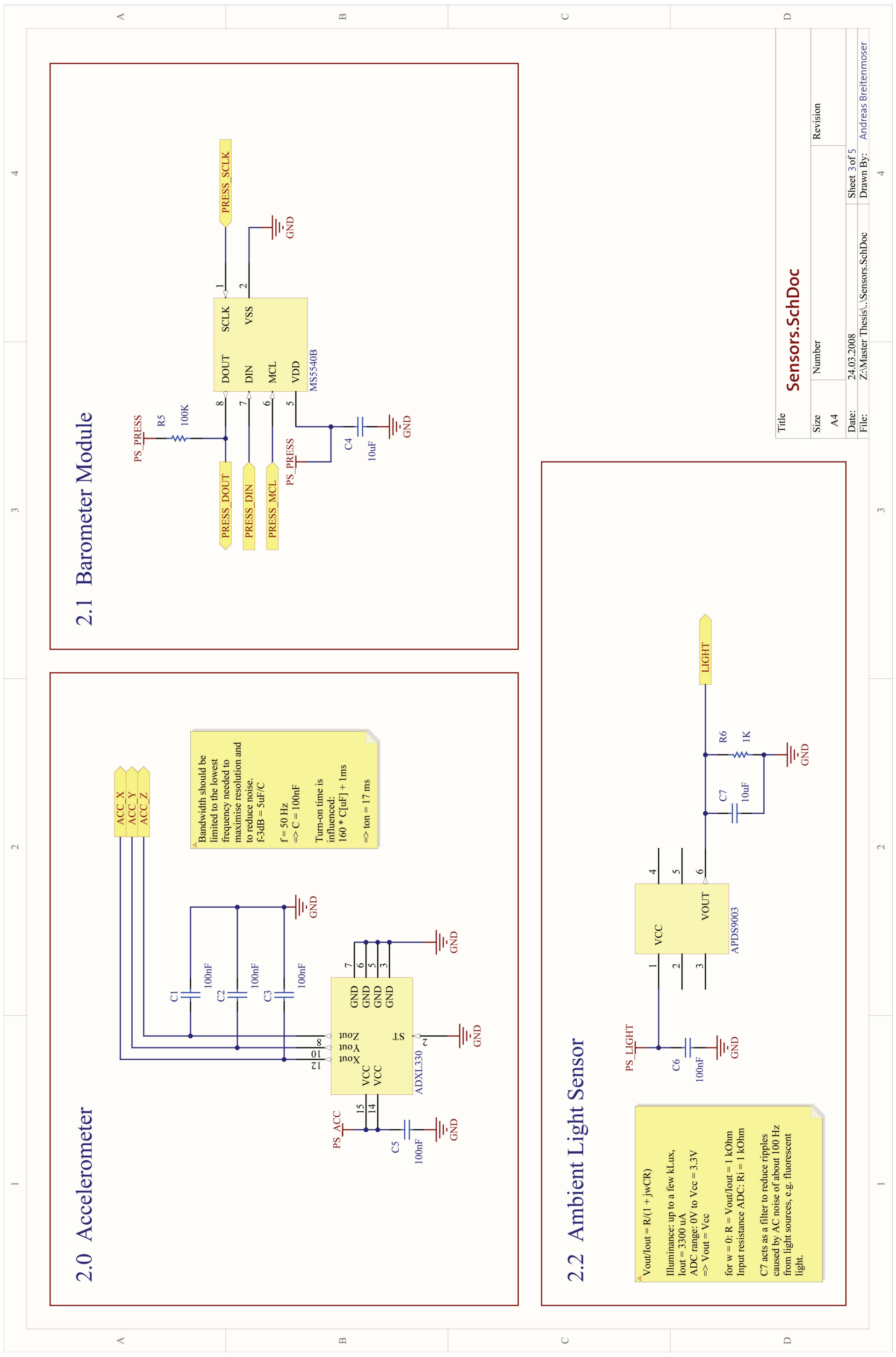

2.0 Accelerometer
2.1 Barometer Module
2.2 Ambient Light Sensor
ACC_X
ACC_Y
ACC_Z
C1 100nF
C2 100nF
C3 100nF
C5 100nF
PS_ACC
ADXL330
Xout Yout Zout ST
VCC VCC
GND GND GND GND
GND
Bandwidth should be limited to the lowest frequency needed to maximise resolution and to reduce noise.
f-3dB = 5uF/C
f = 50 Hz
=> C = 100nF
Turn-on time is influenced:
160 * C[uF] + 1ms
=> ton = 17 ms
PS_PRESS
R5 100K
PRESS_DOUT
PRESS_DIN
PRESS_MCL
PRESS_SCLK
DOUT DIN MCL VDD SCLK VSS
MS5540B
C4 10uF
LIGHT
R6 1K
C7 10uF
PS_LIGHT
C6 100nF
VCC VOUT
APDS9003
Vout/Iout = R/(1 + jwCR)
Illuminance: up to a few kLux, Iout = 3300 uA
ADC range: 0V to Vcc = 3.3V
=> Vout = Vcc
for w = 0: R = Vout/Iout = 1 kOhm
Input resistance ADC: Ri = 1 kOhm
C7 acts as a filter to reduce ripples caused by AC noise of about 100 Hz from light sources, e.g. fluorescent light.
Title Sensors.SchDoc
Size A4
Number
Revision
Date: 24.03.2008
Sheet 3 of 5
File: Z:\Master Thesis\..\Sensors.SchDoc
Drawn By: Andreas Breitenmoser

Figure A.7: Voice Module: sensors.

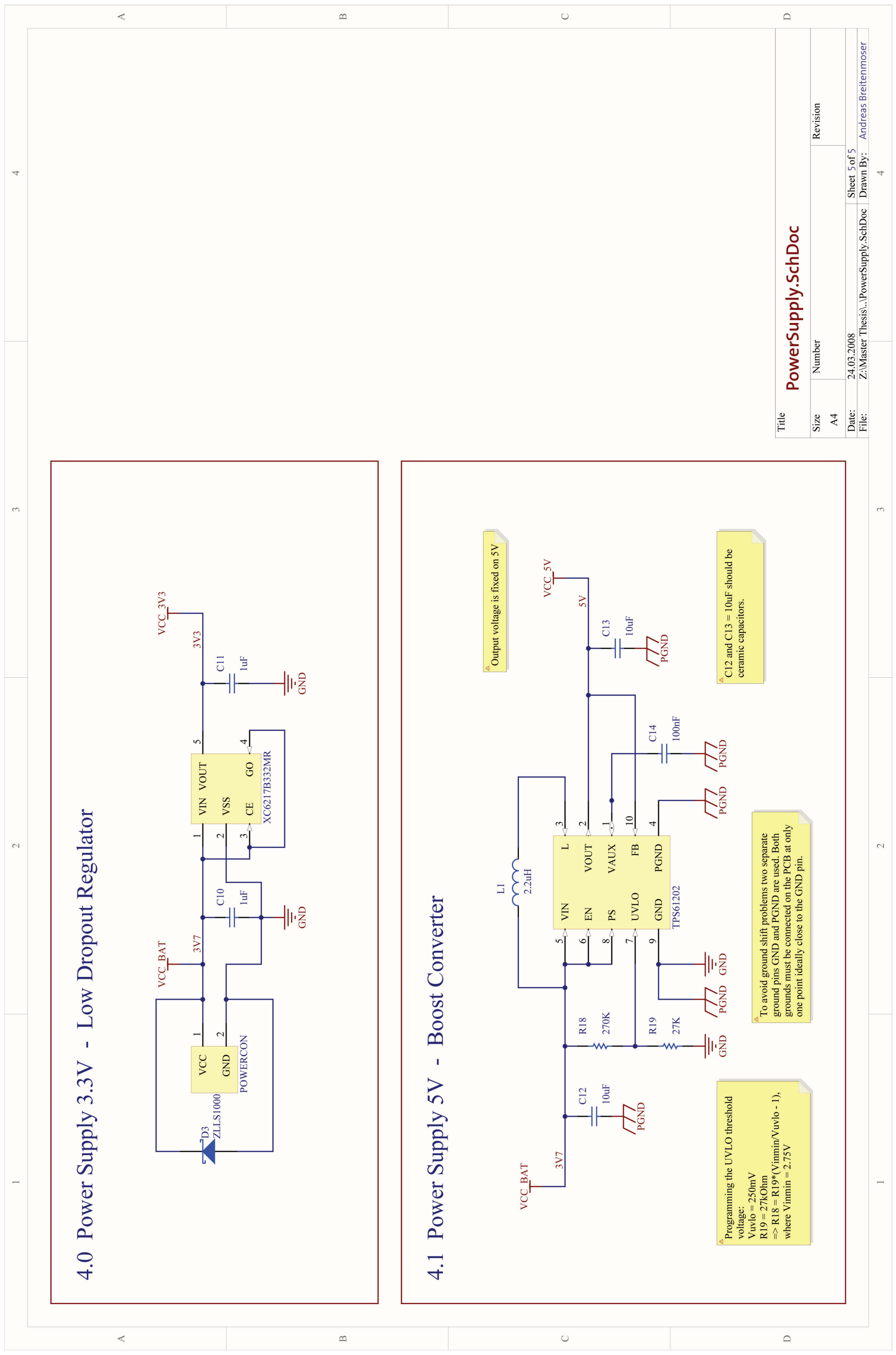

4.0 Power Supply 3.3V - Low Dropout Regulator
VCC_BAT
3V7
VCC_3V3
3V3
C10
1uF
C11
1uF
GND
VIN
VOUT
VSS
CE
GO
XC6217B332MR
VCC
GND
POWERCON
D3
ZLLS1000
4.1 Power Supply 5V - Boost Converter
Output voltage is fixed on 5V
VCC_5V
5V
C13
10uF
PGND
C14
100nF
C12 and C13 = 10uF should be ceramic capacitors.
L1
2.2uH
VIN
EN
PS
UVLO
GND
L
VOUT
VAUX
FB
PGND
TPS61202
R18
270K
R19
27K
C12
10uF
VCC_BAT
3V7
To avoid ground shift problems two separate ground pins GND and PGND are used. Both grounds must be connected on the PCB at only one point ideally close to the GND pin.
Programming the UVLO threshold voltage: Vuvlo = 250mV R19 = 27kOhm => R18 = R19*(Vinmin/Vuvlo - 1), where Vinmin = 2.75V
Title
PowerSupply.SchDoc
Size
A4
Number
Revision
Date: 24.03.2008
Sheet 5 of 5
File: Z:\Master Thesis\..\PowerSupply.SchDoc
Drawn By: Andreas Breitenmoser

Figure A.8: Voice Module: power supply.

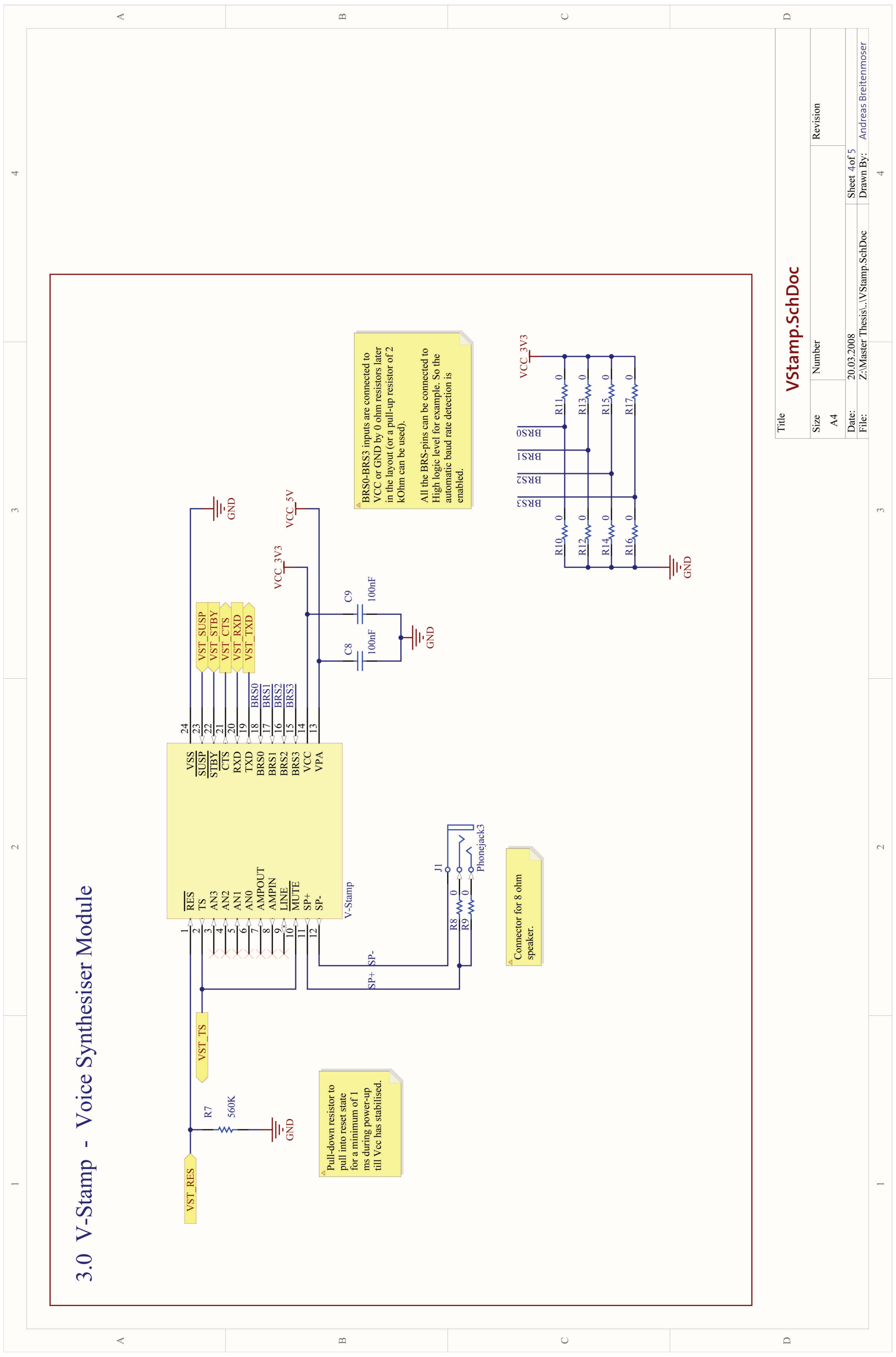
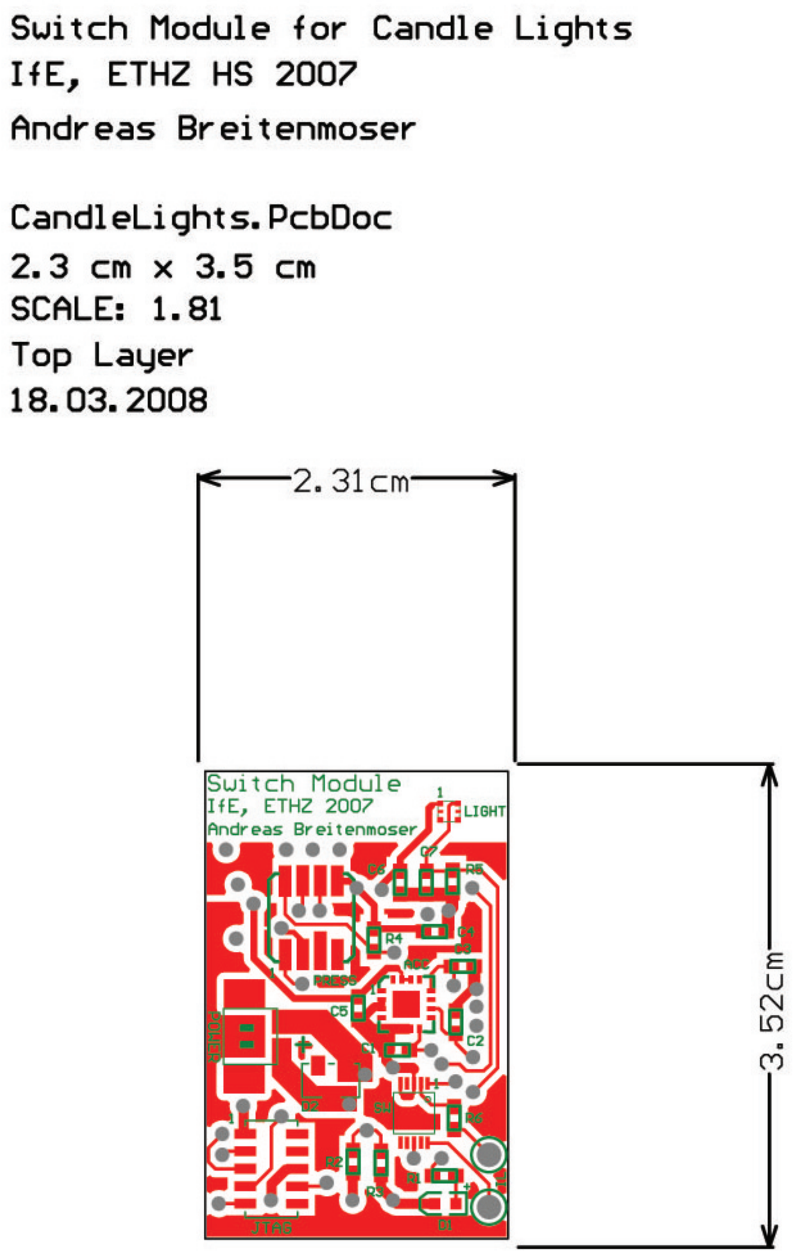

3.0 V-Stamp - Voice Synthesiser Module
BRS0-BRS3 inputs are connected to VCC or GND by 0 ohm resistors later in the layout (or a pull-up resistor of 2 kOhm can be used).
All the BRS-pins can be connected to High logic level for example. So the automatic baud rate detection is enabled.
Pull-down resistor to pull into reset state for a minimum of 1 ms during power-up till Vcc has stabilised.
Connector for 8 ohm speaker.
V-Stamp
RES
TS
AN3
AN2
AN1
AN0
AMPOUT
AMPIN
LINE
MUTE
SP+
SP-
VSS
SUSP
STBY
CTS
RXD
TXD
BRS0
BRS1
BRS2
BRS3
VCC
VPA
VST_SUSP
VST_STBY
VST_CTS
VST_RXD
VST_TXD
VST_TS
VST_RES
R7 560K
GND
VCC_3V3
VCC_5V
C8 100nF
C9 100nF
R8 0
R9 0
J1
Phonejack3
SP+
SP-
R10 0
R11 0
R12 0
R13 0
R14 0
R15 0
R16 0
R17 0
Title VStamp.SchDoc
Size A4
Number
Revision
Date: 20.03.2008
Sheet 4of 5
File: Z:\Master Thesis\..\VStamp.SchDoc
Drawn By: Andreas Breitenmoser

Figure A.9: Voice Module: V-Stamp.

A.3 Layout

A.3.1 Layout of the Switch Module


Switch Module for Candle Lights
IfE, ETHZ HS 2007
Andreas Breitenmoser

CandleLights.PcbDoc
2.3 cm x 3.5 cm
SCALE: 1.81
Top Layer
18.03.2008

Figure A.10: Switch Module: top layer.

Switch Module for Candle Lights
IfE, ETHZ HS 2007
Andreas Breitenmoser

CandleLights.PcbDoc
2.3 cm x 3.5 cm
SCALE: 1.81
MidLayerTop_Vcc
18.03.2008

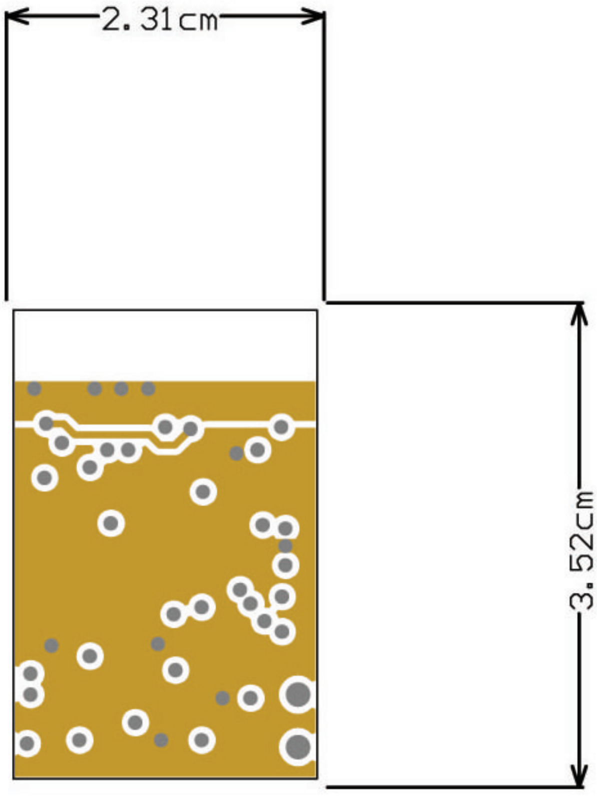


Figure A.11: Switch Module: top midlayer.

Switch Module for Candle Lights
IfE, ETHZ HS 2007
Andreas Breitenmoser

CandleLights.PcbDoc
2.3 cm x 3.5 cm
SCALE: 1.81
MidLayerBottom_GND
18.03.2008

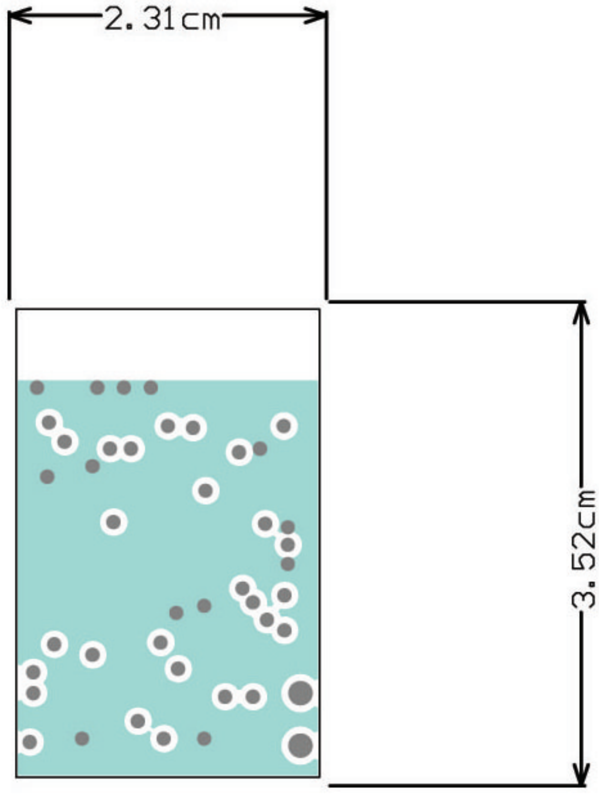


Figure A.12: Switch Module: bottom midlayer.

Switch Module for Candle Lights
IfE, ETHZ HS 2007
Andreas Breitenmoser

CandleLights.PcbDoc
2.3 cm x 3.5 cm
SCALE: 1.81
Bottom Layer
18.03.2008

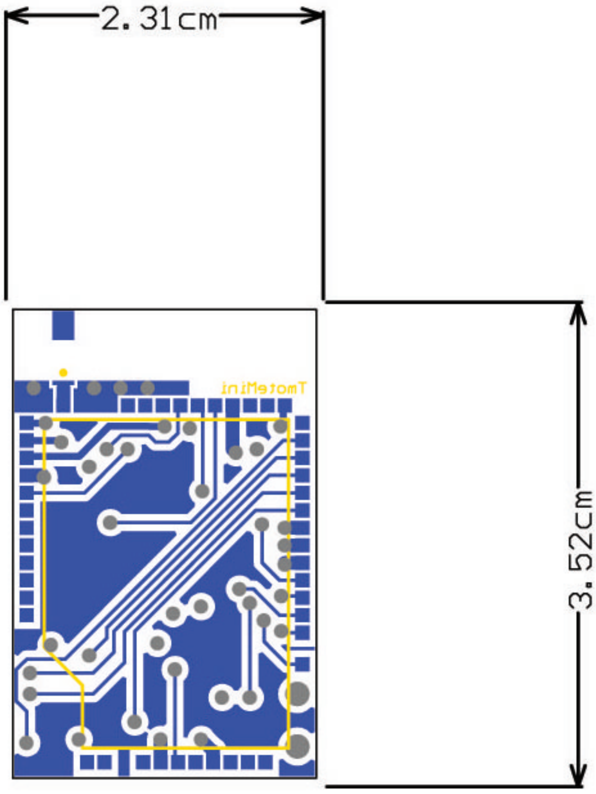


Figure A.13: Switch Module: bottom layer.

A.3.2 Layout of the Voice Module

Voice Module for Text-to-Speech Conversion
IfE, ETHZ HS 2007
Andreas Breitenmoser

VoiceModule.PcbDoc
3.43 cm x 4 cm
SCALE: 1.81
Top Layer
18.03.2008

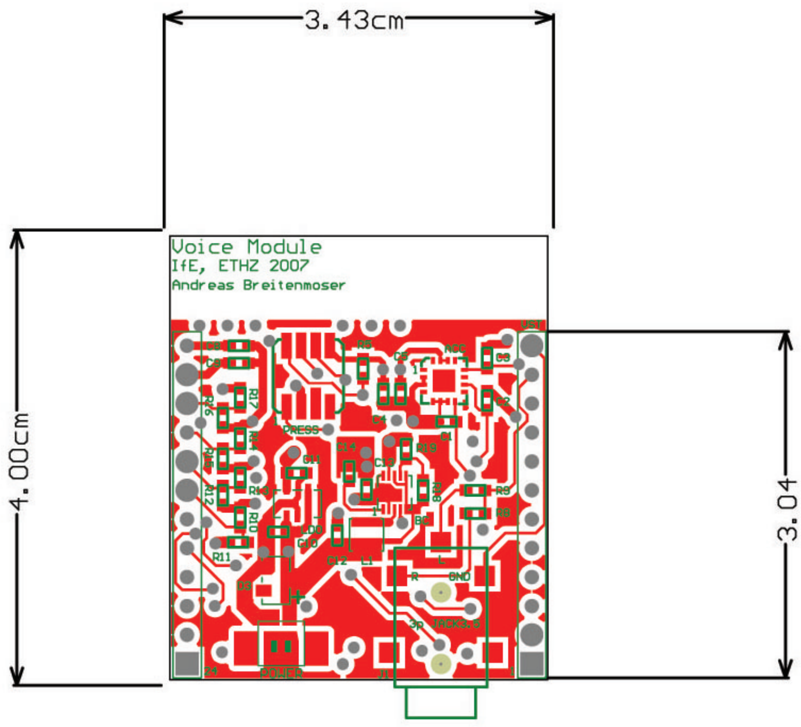


Figure A.14: Voice Module: top layer.

Voice Module for Text-to-Speech Conversion
IfE, ETHZ HS 2007
Andreas Breitenmoser

VoiceModule.PcbDoc
3.43 cm x 4 cm
SCALE: 1.81
MidLayerTop_Vcc
18.03.2008

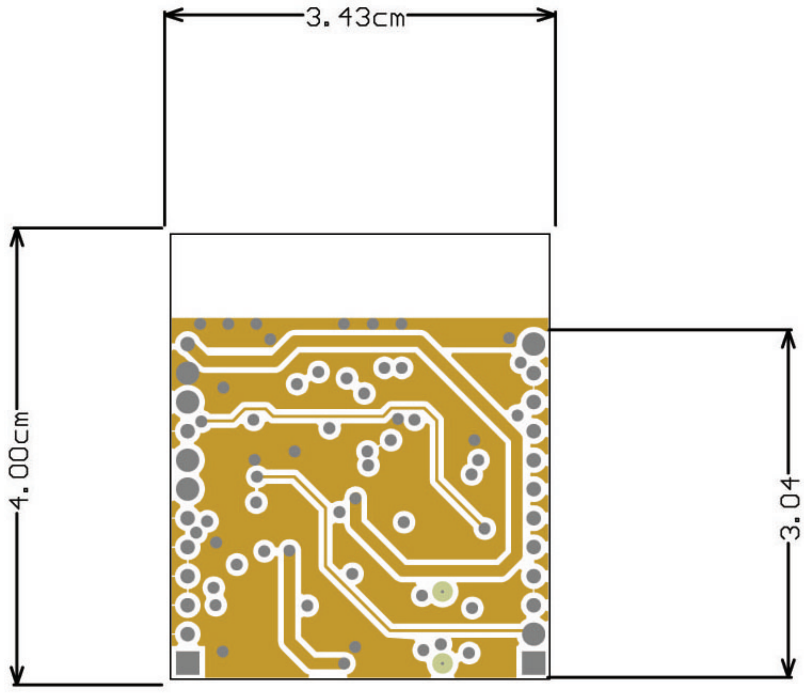


Figure A.15: Voice Module: top midlayer.

Voice Module for Text-to-Speech Conversion
IfE, ETHZ HS 2007
Andreas Breitenmoser

VoiceModule.PcbDoc
3.43 cm x 4 cm
SCALE: 1.81
MidLayerBottom_GND
18.03.2008

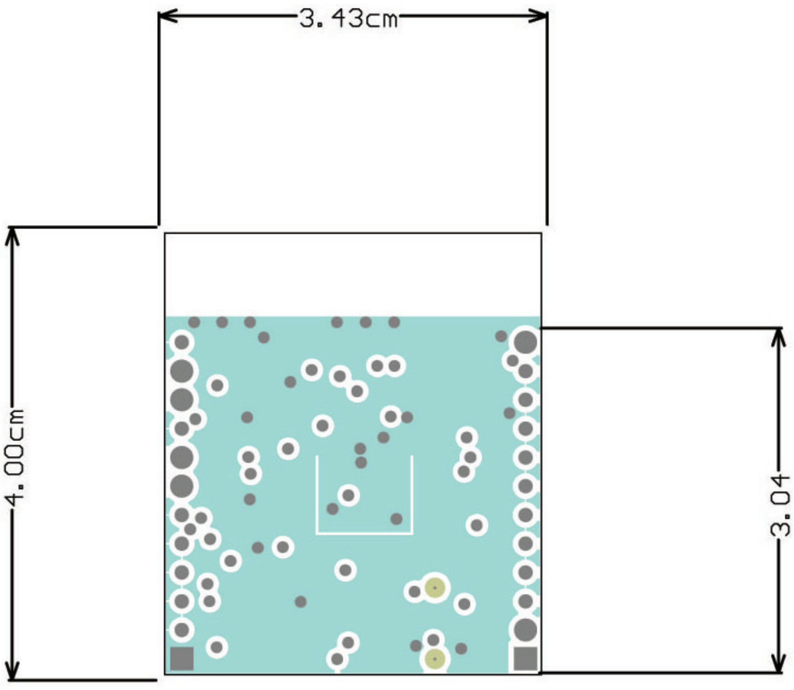


Figure A.16: Voice Module: bottom midlayer.

Voice Module for Text-to-Speech Conversion
IfE, ETHZ HS 2007
Andreas Breitenmoser

VoiceModule.PcbDoc
3.43 cm x 4 cm
SCALE: 1.81
Bottom Layer
18.03.2008

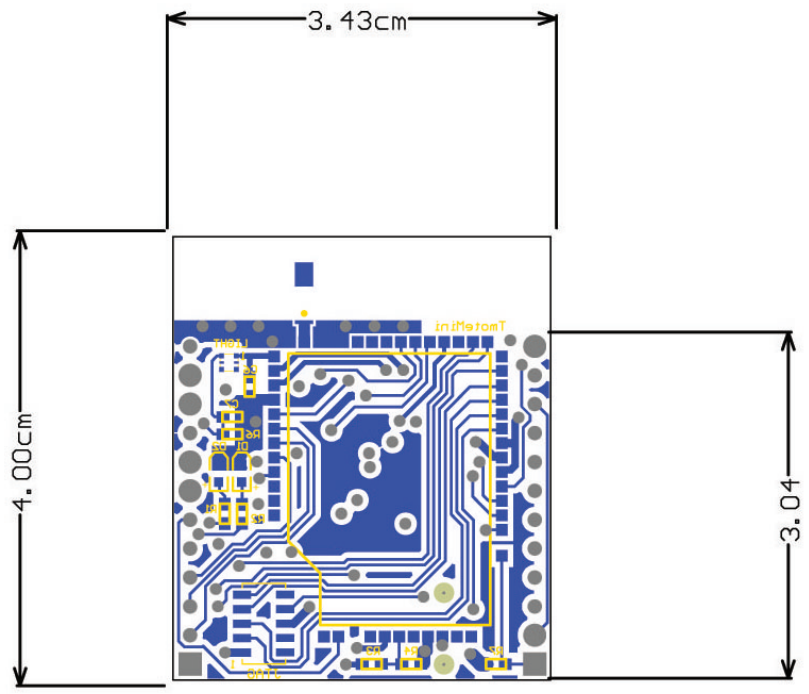


Figure A.17: Voice Module: bottom layer.

Appendix B

Activity Recognition

B.1 Experiment: Shaking the Foam Dice

1) – 3): Correlated shaking

1) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Shake the dice for periods of about 15s, *faces pressed together*, changing the orientation of the dice each time a new period begins.

2) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Shake the dice for periods of about 15s, edge or corner of one dice *pressed to the face of the other*, changing the orientation each time a new period begins.

3) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Shake the dice for several periods, *faces pressed together*, the orientation remains the same each time:

   1. Shake: shake for a longer period of about 25s.
   2. Shake: shake for a shorter period of about 5s.
   3. Shake: shake heavily (try to keep your normal shaking speed or frequency).
   4. Shake: shake slightly (try to keep your normal shaking speed or frequency).
   5. Shake: shake slowly (try to keep your normal shaking elongation or strength).
   6. Shake: shake fast (try to keep your normal shaking elongation or strength).

4) – 6): Uncorrelated shaking

4) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Shake the dice independently for periods of about 15s, holding one dice in each hand.

5) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Confusion experiment for one person: hold one dice in each hand, try to shake as similar (correlated) as possible with the two dice (imitate shaking when both dice are hold in direct contact).

6) Recording: 20Hz sampling frequency, 4000 samples, 3.33 min

   Confusion experiment for two people: each person shakes one dice, both people shake at the same time, both try to shake as similar (correlated) as possible (imitate correlated shaking of a single person).

B.2 Further Plots for Activity Recognition Evaluation

Complete search used a 3-dimensional Pareto front to determine the feature sets with best recognition rates. A broad range of feature sets was found. These sets were evaluated for varying window sizes and varying numbers of features in a set to get a general idea of the best feature mixes. As done for the simple feature sets, also a plot of the performance metrics including FFT features was generated in Figure B.1.

Promising feature sets are all Pareto optimal, i.e. no global optimum can be found. Pareto optimal feature sets can result in quite different ratios of complementary measures, such as precision and recall. Figure B.2 shows the Pareto fronts in the case of the optimum feature set with simple features (mean and zero crossings) and with simple features and FFT features (mean, FFT band 2, FFT band 3). Depending on the chosen cost function, which defines the desired performance of the classifier (higher recall or higher precision), a different feature set with different window size must be chosen. The two examples presented in Figure B.2 show either a rather steep or rather flat curve. This means that either precision or recall can be optimised without losing too much in the complementary performance.

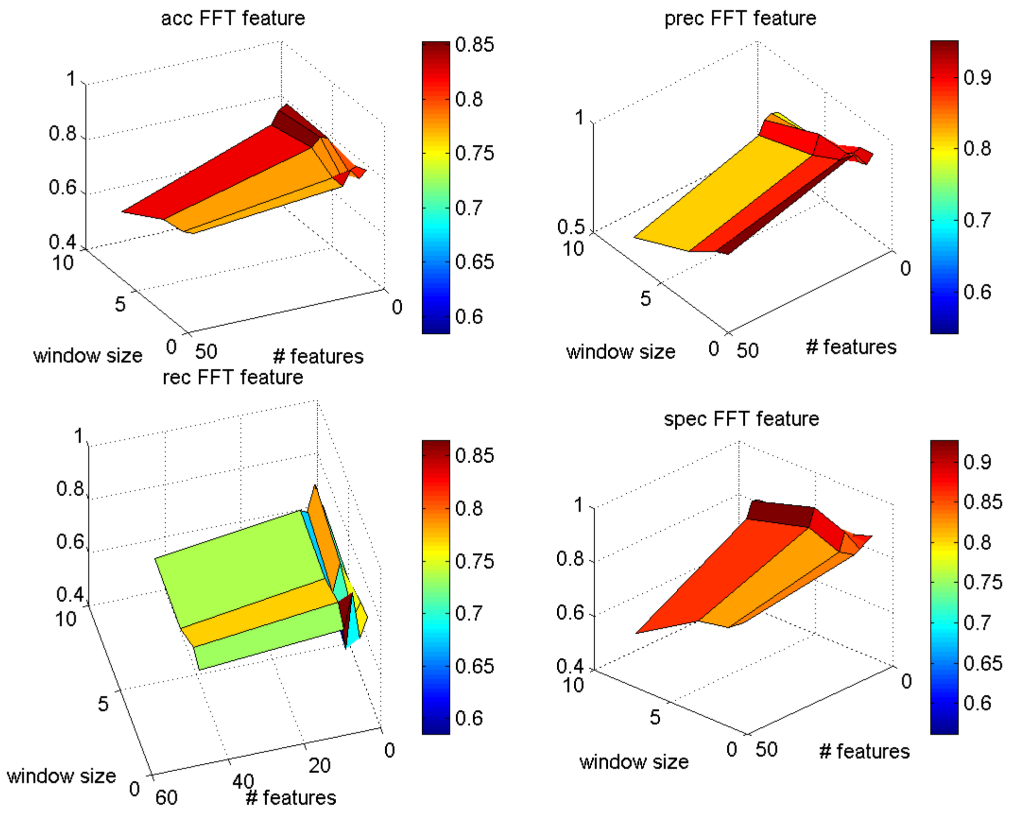


Figure B.1: Performance metric evaluation for the FFT features.

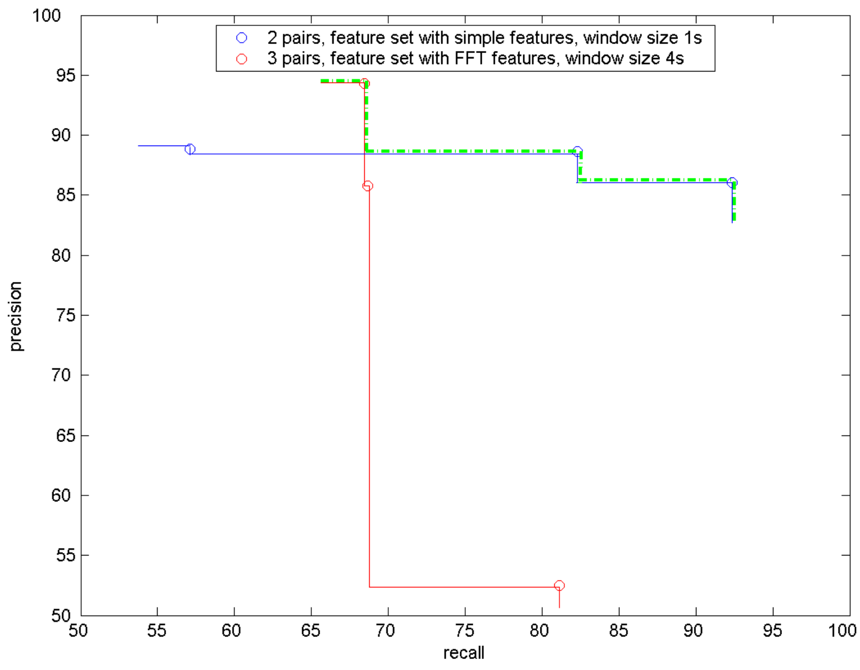


Figure B.2: Pareto fronts for the features mean and zero crossings as well as for the features mean, FFT band 2 and FFT band 3. Depending on the desired precision/recall-ratio one or the other is preferred (green dashed line).

Appendix C

How To/Quick Start Guide

C.1 Loading Code to the Switch and Voice Module

1. Open a shell and compile the program:

```
$> make tsoswitch
```

   or

```
$> make tsovoice
```

2. Open a second shell and start the msp430-gdbproxy:

```
$> msp430-gdbproxy -port=2000 msp430
```

3. Return to the first shell and load the program with msp430-gdb:

```
$> msp430-gdb build/tsoswitch/main.exe
```

   or

```
$> msp430-gdb build/tsovoice/main.exe
```

4. Now, type the following lines into the shell:

```
(gdb) set remoteaddresssize 64
(gdb) set remotetimeout 999999
(gdb) set remote memory-write-packet-size 1024
(gdb) set remote memory-write-packet-size fixed
(gdb) set remote memory-read-packet-size 1024
(gdb) set remote memory-read-packet-size fixed
(gdb) target remote localhost:2000
```

   Hint: You can put the lines above in a file named gdb.ini into your home directory. GDB will then automatically execute these lines each time it is started.

5. Erase MSP430's flash memory:

```
(gdb) monitor erase all
```

6. Load the program:

```
(gdb) load
```

7. Start the program:

```
(gdb) continue
```

C.2 Loading a Sound File to the V-Stamp Module

V-Pod development kit from RC Systems is used to connect the V-Stamp voice synthesiser directly to the PC (see Figure C.1. This allows the use of development tools as RCStudio and RCLink. Data files are downloaded into the V-Stamp module before it gets used.

There are several possibilities to communicate with the V-Stamp module from the PC over the serial connection:

- *RCStudio and RCLink* RCStudio and RCLink are application programs provided by RC Systems. Data files (e.g. sound files) and text messages can be created and tones generated with RC Studio and downloaded to the V-Stamp module. RCStudio and RCLink are available from RC Systems's homepage for free[1].

[1]http://www.rcsys.com/dnlds.htm#sw

- *HyperTerminal or command prompt* The HyperTerminal or command prompt (DOS) can be used to interface with the V-Stamp modules directly.

C.3 Application Development with Titan: How to run Titan?

Titan is started from the PC in simulation[2] or physical mode. The configuration reader allows to load a new configuration from a text file.

1. Compile Titan's TinyOS part for each module separately (a specific task combination must be selected in TitanC.nc) and load it onto the Switch Module or Voice Module (cf. section C.1).

2. Create a task graph or compose a set of task graphs (which may be reconfigured by a FSM) with the configuration text files.

3. Start Titan's Java part: Start Titan in (simulation or) physical mode

```
> titan start (sim)
```

   and load the appropriate configuration file to start the execution

```
> titan load filename_config.txt
```

[2]Controlled application development for distributed embedded systems is difficult. Simulation tools can help. The simulation of Titan is based on TOSSIM, the TinyOS mote simulator, and Python. Titan's simulator starts Python in the background. Python allows to interact with a running simulation. The Python interface calls a C++ interface that then starts TOSSIM. TOSSIM replaces TinyOS components with simulation implementations and simulates sequences of events.

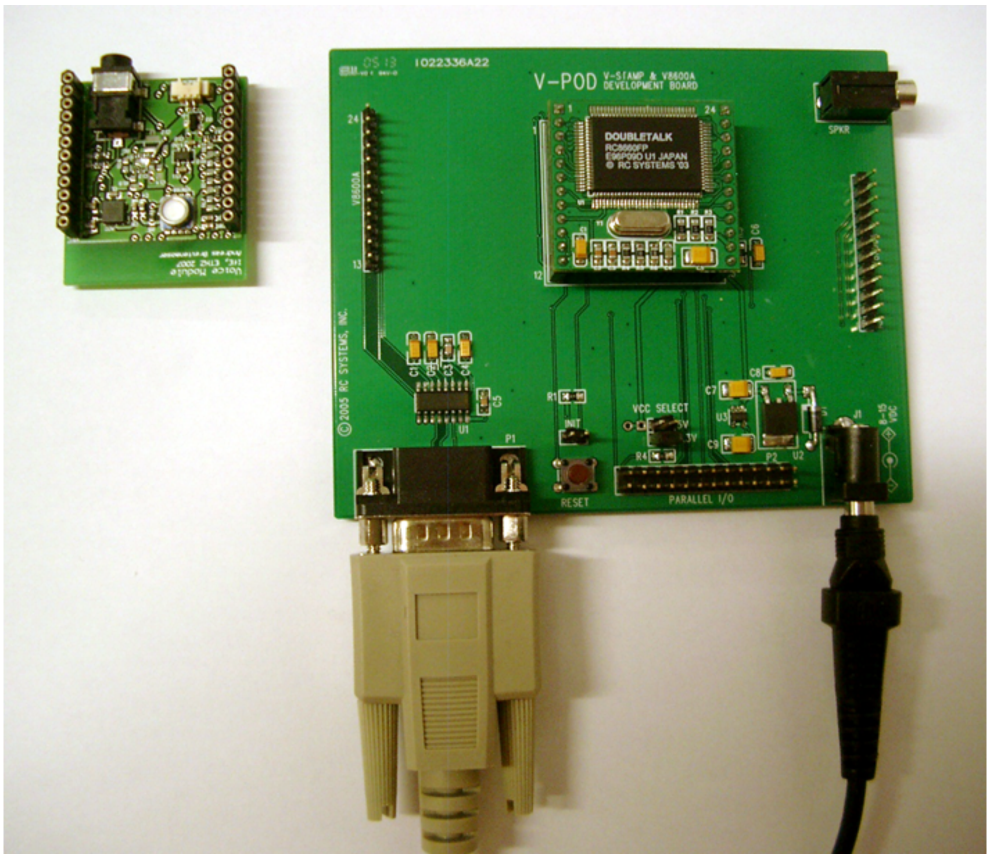


Figure C.1: V-Pod development kit. The V-Stamp voice synthesiser module is removed from the Voice Module and plugged in the socket of the V-Pod for connection to the PC.