ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze

Master Thesis

Acoustic Emission Sensing in Wireless Sensor Networks

Josua Hunziker

5th May 2011

Supervisors:

Dr. Jan Beutel
Dr. Stephan Gruber

Professor:

Prof. Dr. Lothar Thiele

Abstract

The PermaSense project is a joint computer science and geoscience project, aiming to understand the influence of climate change on permafrost. A reliable wireless sensor network (WSN) infrastructure has been developed, enabling scientists to constantly monitor the condition of certain rock walls on watch in near real time. However, as the network infrastructure is optimized for energy efficient operation and low data rates, only slow physical processes can be monitored.

This thesis explores the possibilities of acquiring sensor data at higher sampling rates in the context of the existing PermaSense architecture, specifically focusing on the application of acoustic emission (AE) measurements on rock walls. To this end, the architectural concept of "processing sensors" is introduced and a prototype AE sensing system is presented, integrating seamlessly into the existing PermaSense WSN infrastructure. Specifically, the prototype's architecture and design are discussed, followed by a system evaluation based on laboratory experiments.

The evaluation results show that the AE sensing system designed successfully integrates AE measurements into PermaSense. Furthermore, variations in measurement quality depending on the ambient temperature are characterized qualitatively. Further steps include deploying the developed system in the field in order to gain experience in the targeted operating environment, as well as quantitatively analyzing and eliminating temperature dependent measurement errors.

Acknowledgements

First and foremost, I would like to thank Dr. Jan Beutel for his excellent support, the constant feedback and all the insightful discussions throughout this project. You challenged me to dig deeper in every aspect. I also thank Dr. Stephan Gruber for co-advising this thesis, providing the geoscientific focus and being such a patient and grateful, yet demanding "customer". Furthermore, I would like to thank Prof. Dr. Lothar Thiele for having given me the opportunity to contribute to the research activity of the Computer Engineering group in several student projects throughout the last few years. Many thanks also to the following people who supported me in various aspects: Tonio Gsell, Roman Lim, Christoph Walser and Mustafa Yücel (Computer Engineering Group), Michael Lerjen (Communication Theory Group), Manuel Löhr and Joachim Sell (Physical Acoustics Corporation) as well as Dr. Mauro Ciappa (Integrated Systems Laboratory).

I would also like to express my gratitude to my family and friends for all their constant support and care. My dear parents: You have invested so much in my life – no thank could ever measure it up. Romina, my fiancée: You are a constant source of joy and strength in my life. Nobody else could fill what you are to me.

I thank God, my Father, my King. Your truth sets free!

Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

The PermaSense project [1] is a joint computer science and geoscience project, aiming to understand the influence of climate change on permafrost. A special focus lies on the stability of rock walls in alpine environments, where natural hazards pose a constant danger to habitants and infrastructure. PermaSense makes use of state-of-the-art ultra low-power sensor nodes that form a multihop wireless sensor network (WSN). Data samples measured are transmitted over the WSN to a core station, where the data is relayed to a backend server using a WLAN link and the public internet infrastructure.

In the context of such an environmental monitoring system, low energy consumption is a crucial requirement for the wireless sensor nodes. The sensor node platform currently used in PermaSense can sense and transmit data for 3-5 years from a single battery. In order to achieve this high energy efficiency, platform architecture, operating modes and network protocol have been optimized to allow for a very low average power consumption by utilizing the sensor node's sleep mode capabilities as extensively as possible. This setup has proven to be very reliable and stable during the continuous operation of PermaSense measurement campaigns at Matterhorn and Jungfraujoch for more than three years at the time of writing.

In addition to the current measurements of rock temperature, rock humidity and crack movement, interest arose from the geoscientifical side to capture physical processes that expose faster variations. One phenomenon of par-

ticular interest are acoustic emissions (AE) during crack and ice formation. These acoustic emissions are generally very short in the order of a few milliseconds and exhibit frequencies of multiple kHz. As the architecture and operating mode of the existing sensor nodes are optimized to capture much slower processes and thus produce at most a few samples per minute, they are not suitable for measuring and processing such fast processes. Therefore, new concepts are needed in order to allow the study of AE and other fast phenomena in the context of PermaSense.

In this thesis, a sensor platform architecture concept that is capable of performing high data rate measurements in a low-power WSN environment is proposed. Furthermore, an implementation of this concept that is tailored to measure, process and transmit AE data and fully integrates into the existing PermaSense WSN infrastructure is presented in detail.

1.2 Contributions

The contributions of this thesis are the following:

- An architectural concept for performing high data rate measurements in a low-power WSN environment.
- A proof-of-concept implementation tailored for AE measurements in PermaSense, consisting of a hardware platform ("AEBoard") and a distributed software system running in this platform ("PermAE").
- An operational performance analysis and comparison to a commercial AE measurement system.

1.3 Related Work

The PermaSense project [1], its findings and developments form the basic context for the work presented in this thesis. The PermaSense system architecture has been described in [2, 3]. It has proved to deliver high-quality data reliably and over an extended period of time. However, the PermaSense architecture has so far primarily been used for measurements with a relatively low data rate [4]. This thesis aims to extend the PermaSense ecosystem, in order to support applications that require higher sampling rates without compromising the standards of quality and reliability achieved so far. There are also efforts to integrate high-accuracy GPS measurements into PermaSense, facing similar challenges but taking different approaches for data reduction [5, 6, 7].

High data rate WSN applications in general and processing of acoustic data specifically have previously been studied in publications such as [8, 9]. Especially the latter publication by Simon et al. inspired some concepts that can be found in this thesis, specifically the idea that a significant part of the necessary data processing is done directly on the sensor node, thereby reducing the amount of data to transmit over the network significantly. Werner-Allen et al. presented another concept for data reduction in [10], transmitting raw samples selectively based on a value metric for the measured data.

There are also projects that dealt with AE sensing in WSN specifically. Grosse et al. presented a 4 channel AE device for WSN deployments in [11, 12]. Upon detection of AE activity, a number of samples is recorded and then transmitted over the WSN, in the meanwhile stopping data acquisition. Bachmaier introduced another WSN AE device in [13], not sampling any AE data at all but only estimating AE activity based on a fixed threshold. Recently, Lédeczi et al. presented a WSN based AE monitoring system for bridges in [14], saving power by residing in sleep mode for most of the time and only waking up and sampling upon increased AE activity that is indicated by a low-fidelity sensor. In all these projects, the AE signal is not continuously sampled in order to save energy and thus prolonging the system lifetime. However, as the processes that produce AE activity in rockwalls are not yet well understood, this thesis follows a different approach. In order to provide as much insight as possible, we focus on characterizing as many AE events as possible at a continuously high sampling rate. Considering these requirements, power consumption has been optimized as a secondary goal only.

1.4 Outline

Chapter 2 discusses the concepts of AE sensing and WSNs in greater detail. Chapter 3 describes the initial AE sensing experiments that initiated this thesis and derives the specifications for an AE sensing system in PermaSense from these preliminary results. In Chapter 4, our general concept for high data rate measurements in a low-power WSN environment is presented. Chapters 5 and 6 document the AEBoard hardware platform developed as well as PermAE, a distributed software system that runs on the AEBoard and is able to measure and transmit AE data efficiently. The overall system performance of AEBoard and PermAE is then evaluated in Chapter 7. Finally, Chapter 8 summarizes the presented work and proposes further steps based on this thesis.

2

Basic Concepts

This chapter aims to give a brief overview over the two basic concepts underneath this thesis: Wireless Sensor Networks (WSNs) as well as Acoustic Emission (AE) sensing. It further provides references to comprehensive sources that cover these topics in-depth.

2.1 Wireless Sensor Networks

Wireless communication has become an ubiquitous technology these days. Not only do personal mobile devices like smartphones and tablet computers use wireless communication interfaces as their primary channels, also in industry and safety-critical applications, wireless technology is about to change traditional patterns of human-machine interaction. Concepts like “The Internet of Things” [15] or “Ubiquitous Computing” [16] build on the existence of small independent computing platforms that are connected to their environment and thus form the basic infrastructure for a "smart environment".

The term “Wireless Sensor Network” commonly refers to a distributed system, consisting of many light computing platforms that communicate over a wireless channel. Each of these so-called “sensor nodes” is sensing phenomena of its physical environment and forwards these measurements over the network. In many common WSN architectures, all data harvested in the network are collected at one or multiple distinct “sink nodes”, where the data is either being forwarded over a different communication link, or can be accessed directly.

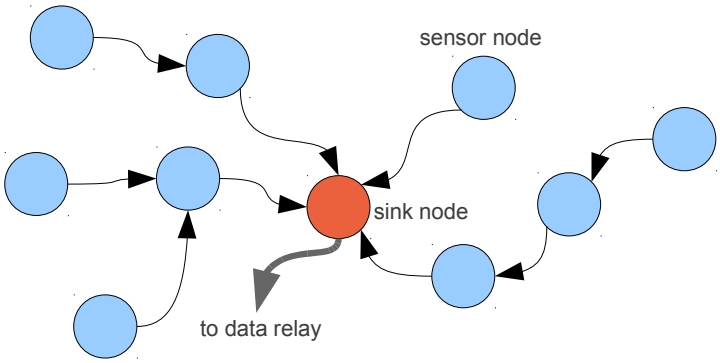


Figure 2.1: Basic principle of a multihop WSN

This approach of placing low-weight sensors unobtrusively in the environment allows to measure and monitor phenomena very closely and in a natural environment, as the nodes can be tailored and placed in such a way, that they have minimal impact on the process that is to be captured [17, 18].

There is a broad palette of applications for WSNs, ranging from military applications over environmental monitoring, health applications, monitoring of infrastructure and buildings as well as a wide range of industrial prospects [19]. Their generally low cost, ease of deployment and low maintenance requirements make WSNs an area of active research and high industrial interest.

Nevertheless, developing and deploying reliable WSNs is still a challenging task. One key constraint on autonomous wireless platforms is energy - the amount of energy available to the system ultimately determines the maximum time of unattended operation and sensing. Although there exist a variety of concepts for energy harvesting on WSN nodes (e.g., [20]), still most of the platforms in practical use nowadays use batteries as their sole power supply. This requires a stringent power management on all nodes, often resulting in putting the node into a low-power mode for most of the time, implying frequent disconnection and reconnection of nodes in the network. The resulting very dynamic network structure requires novel approaches for network management and routing, as classical network protocols often assume rather static network topologies with infrequent changes. Thus, multiple WSN-tailored network protocols have evolved over the past decade, e.g. Dozer [4], PEGASIS [21] or APTEEN [22], that aim to allow for an energy-efficient WSN communication.

2.1.1 The PermaSense WSN Architecture

The PermaSense WSN infrastructure is tailored for maximum reliability even under harsh environmental conditions. Sudden temperature changes, snow,

ice, rockfall and other natural impacts drastically increase the probability of temporary or permanent disconnection of a node, compared to, e.g., an indoor deployment with rather controlled environmental conditions. Fault tolerance is therefore a key requirement on all scales.

The core WSN of PermaSense consists of sensor nodes based on the "TinyNode" platform by Shockfish SA [23, 24]. These TinyNodes are equipped with a custom-built sensor interface board (SIB), that connects to various sensors measuring rock temperatures, rock humidity, crack movement etc. The sensor nodes run software based on the TinyOS operating system [25] and communicate over the ultra low-power Dozer network protocol [4], which is configured to schedule communication slots every 30 seconds. All data is forwarded to a single sink node, which is part of the so-called "CoreStation". The CoreStation is basically an embedded PC platform, featuring WLAN and GPRS downlinks to forward all data to the data backend, which is based on the GSN software [26] and features additional control, management and supervision interfaces. Furthermore, sensors with higher data volume such as high resolution photo cameras [27] and GPS receivers are also connected to the backend over separate network links.

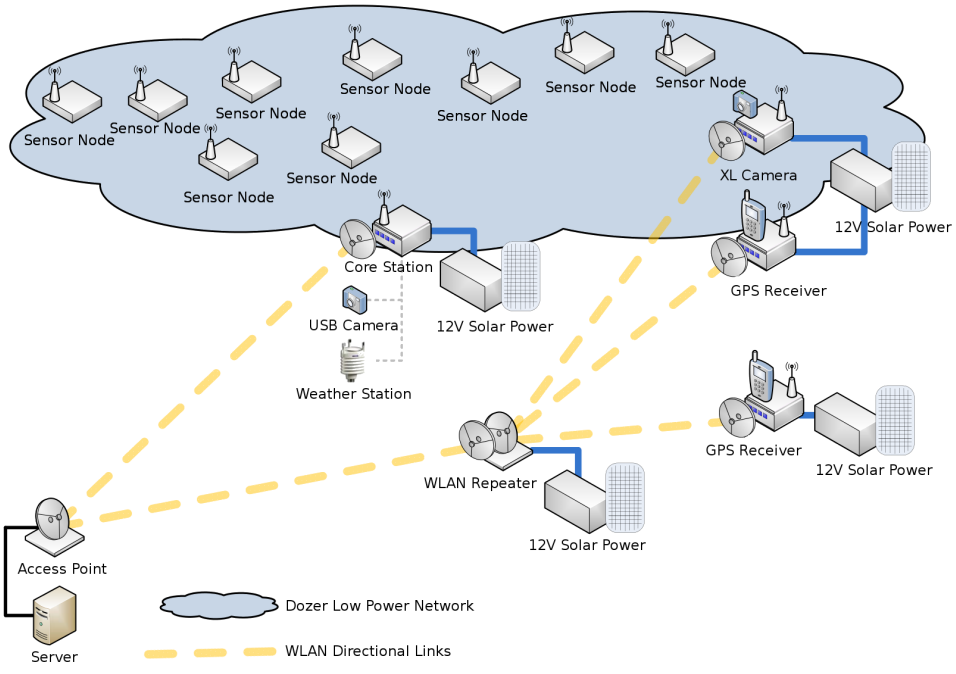


Figure 2.2: The PermaSense network architecture

In case of normal operation, samples arrive at the data backend on the order of a few minutes after measurement, assuming that no network links are congested. Additionally, each sensor node in the WSN features a 1 GB SD

storage, where measured sensor data can be buffered in case of connection loss (e.g., in case of the sensor being snowed in). Once the network connection can be re-established, the accumulated data packets are flushed to the backend.

The sensor nodes run on a single Li-$SOCl_2$ battery which enabled them to operate for more than three years. Larger components, like the core stations as well as the high resolution cameras, are powered by a regenerative power supply.

For more details about the current PermaSense architecture, please refer to [2, 3, 28].

2.2 Acoustic Emission Sensing

It is commonly known from daily experience, that materials often emit sounds under mechanical stress or when internal structures break. Generally speaking, these (often inaudible) breaking sounds are referred to as "Acoustic Emissions" (AE):

> "Acoustic Emission (AE) refers to the generation of transient elastic waves produced by a sudden redistribution of stress in a material. When a structure is subjected to an external stimulus (change in pressure, load, or temperature), localized sources trigger the release of energy, in the form of stress waves, which propagate to the surface ... Sources of AE vary from natural events like earthquakes and rockbursts to the initiation and growth of cracks, slip and dislocation movements, melting, twinning, and phase transformations in metals. In composites, matrix cracking and fiber breakage and debonding contribute to acoustic emissions." [29]

AE sensing technology began to evolve in the middle of the 20^{th} century. It is a so-called non-destructive testing (NDT) method, providing information about ongoing stress processes in the material sample under test. In contrast to other NDT methods, which are mostly applied before or after the material is being stressed, AE sensing captures processes *while* they are taking place. This makes AE sensing particularly interesting for applications that require constant monitoring of certain structures, e.g. alarm systems for pipeline leaks. Often, this method is used to detect material failure at a very early stage of damage and before the structure fails completely ([30], chapter 1).

2.2.1 Sensor Technology

AE signals are often being captured with piezoelectric sensors. These sensors are directly attached to the surface of the material sample under test, converting dynamic motions at the surface into an electrical signal.

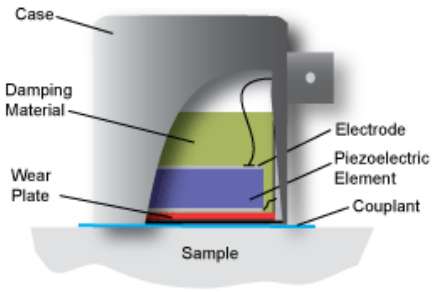


Figure 2.3: Piezoelectric AE sensor (schematic representation). [29]

When using piezoelectric sensors, it must be considered that they are normally operated at resonance and therefore do not allow broadband detection of AE signals. Nevertheless, the frequency ranges of the expected signals are often fairly well known, making it possible to choose the right sensor before the experiment. Thus, their ease of use as well as their high sensitivity make piezoelectric sensors the instrument of choice for many AE applications, even though more sophisticated AE sensor concepts exist (e.g. sensors based on laser systems or on fiber optics).

Before A/D conversion, the electrical signal at the output of a piezoelectric sensor normally needs to be filtered and amplified. This process is also known as "signal conditioning". It is obvious, that these components of an acquisition system are very sensitive and heavily influence the overall measurement quality. The conditioned signal is then sampled and all further processing is then performed digitally.

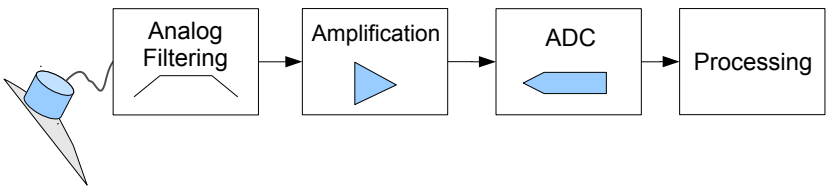


Figure 2.4: AE data acquisition chain

2.2.2 Data Processing and Signal Parameter Analysis

As most modern data analysis systems, today's AE systems almost exclusively perform the necessary signal detection, processing and characterization tasks in the digital domain. AE activity normally occurs rapidly and randomly, resulting in distinct "pulses" of oscillation in the measured AE signal. Conventionally, a threshold is defined by the user in order to distinguish AE activity from background noise. Once the AE signal crosses this threshold, an AE event is recognized. The event is considered to be terminated when the threshold has not been exceeded for a certain, user-defined amount of time (i.e., "hit-lockout time", "dead time" or "posttrigger time"). The AE signal between the first threshold crossing and the end of the posttrigger time is also referred to as "AE Hit" or "AE event". As especially in the early days of AE testing technology, all of the signal processing had to be done in the analog domain, some common signal parameters have evolved that still form the basis for many of today's AE testing and analysis applications. These "classical" AE signal parameters are ([30], chapter 4):

1. **Event Count/Hit Count**: A signal that exceeds the threshold and causes the AE system to accumulate data. Hit counts are often used to show the overall AE activity over a certain period of time.

2. **Length/Duration**: The amount of time between the first threshold crossing of an event and the end of the posttrigger time.

3. **Amplitude**: The maximum signal amplitude within an event.

4. **Rise Time**: The time interval between the first threshold crossing and the time of the peak amplitude.

5. **Pulse Count/Count**: The number of times an AE event signal exceeds the threshold value.

6. **Energy**: There is no single agreed definition of AE signal energy. In this thesis, we define the energy of an AE event as the *signal energy* contained in the event, i.e., the sum of the squared sample amplitudes. Another popular energy definitions is the measured area under the rectified signal envelope.

Figure 2.5 illustrates these aforementioned features of an AE event.

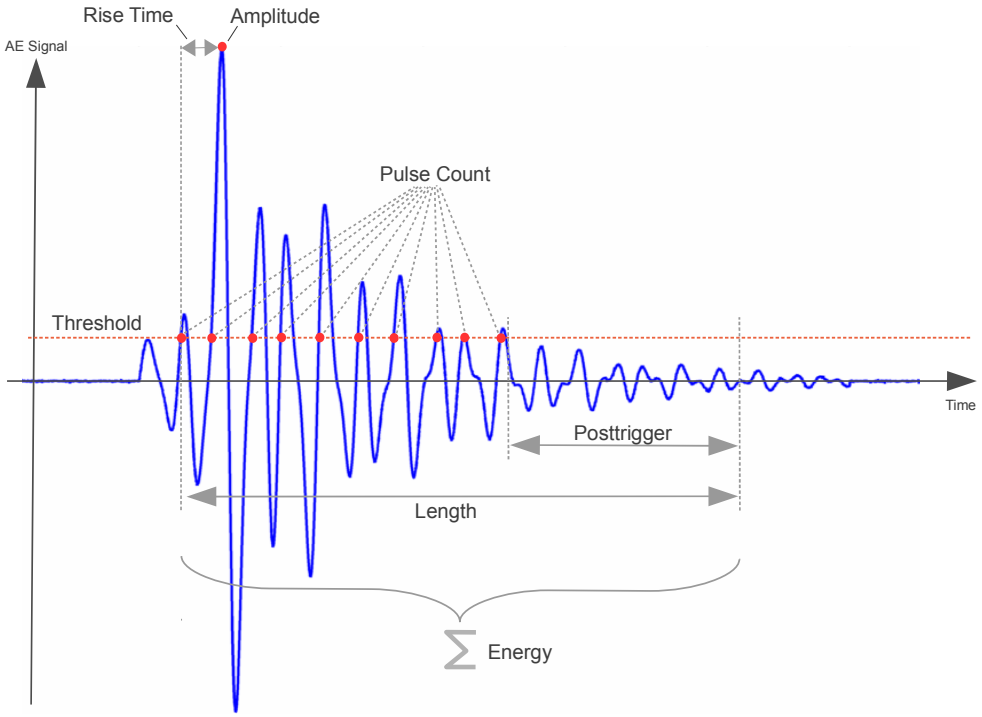


Figure 2.5: AE event parameters

From these basic parameters, further event parameters may be derived:

- The **Average Frequency** is calculated by dividing "Pulse Count" by "Length".
- The **RA Value** is derived from "Rise Time" divided by "Amplitude". This parameter can be used to classify cracks.

2.2.3 AE Sensing in Geosciences

AE sensing originally arose in the context of material sciences and is nowadays particularly popular in the field of civil engineering for structural health monitoring. Nevertheless, this method has a much broader scope of applications, e.g., in geosciences for studying stability and structural displacements in rock structures. In contrast to seismology, where large, long events at low frequencies (i.e., earthquakes) are studied, AE measurements in rock focuses on much smaller, shorter events that can give information about displacements in the order of a few micrometers ([30], chapter 11). In mining, scientists started to monitor rock pillars with AE measurements in the middle of the 20^{th} century. The development of these practical experiments was

accompanied by laboratory experiments, where rock samples were hydraulically stressed and monitored for AE at the same time. Also nowadays, AE in rock structures under stress are still an area of active geoscientific research (see e.g., [31, 32, 33]).

The formation of ice within rock is an important driver of near-surface frost weathering [34] and rock damage at the depth of several meters [35], and in steep terrain, this process may be crucial for the slow preconditioning of rock fall from warming permafrost areas [36]. However, the transfer of corresponding theoretical insight and laboratory evidence to natural conditions characterized by strong spatial and temporal heterogeneity is nontrivial. To prepare corresponding characterization of rock fracture in natural conditions, AE induced by natural thermal cycling and freeze/thaw in a high altitude rock face are studied within the project PermaSense. This complements well previous investigations within PermaSense (e.g., [37, 38]). To the knowledge of the author, no comparable study of AE in alpine rock walls exists so far.

3
System Specification

This chapter describes the experiments and intentions, that led to the specification of an AE sensing platform for PermaSense finally presented in Section 3.4.

3.1 Preliminary Experiments

As no reference experiments of AE measurements in alpine rock walls exist, preliminary experiments had to be carried out in order to get a feeling for the type of AE activity that has to be expected during more in-depth studies. For this purpose, AE experiments with conventional laboratory equipment have been carried out at the Jungfraujoch site in April 2010.

3.1.1 Installation

Six piezoelectric AE sensors have been installed at different positions in the rock wall. The installation positions were selected in order to capture varieties in rock structure and expected humidity, i.e., some sensors were mounted at dry and compact expositions, whereas others were placed at cracked positions featuring a constant flow of water. The sensors were connected to a laboratory computer running the AEWin software from Physical Acoustics Corporation [39] and AE data has been acquired for a period of four days. Both event parameters as well as the raw event waveforms have been captured.

The results of these measurements were very promising and reinforced interest in monitoring AE activity in rock walls over longer periods of time [40]. Thus, the development of a custom AE sensing platform that integrates into the existing PermaSense infrastructure was initiated, whereof a first prototype is presented in this thesis.

Figure 3.1: AE sensor positions for preliminary experiments at Jungfraujoch

3.1.2 Data Analysis from a Technical Perspective

In order to define the basic requirements and constraints for a system implementation, the data gathered in the aforementioned experiment has been analyzed from a technical perspective.

All sensor channels were configured with a fixed threshold of 35 dB and posttrigger value of 1 ms. The signals were sampled with a quantization resolution of 24 bits at a sampling rate of 1 MHz.

Event Rate

Depending on the sensor position, the number of events occurring differed significantly. We could also observe that events often occur in bursts: long periods of low AE activity follow on shorter periods with a much higher activity level.

For the specification of an AE measurement system, the following event rate measures are therefore of particular interest:

- The *maximum event rate* defines the processing an storage bandwidth

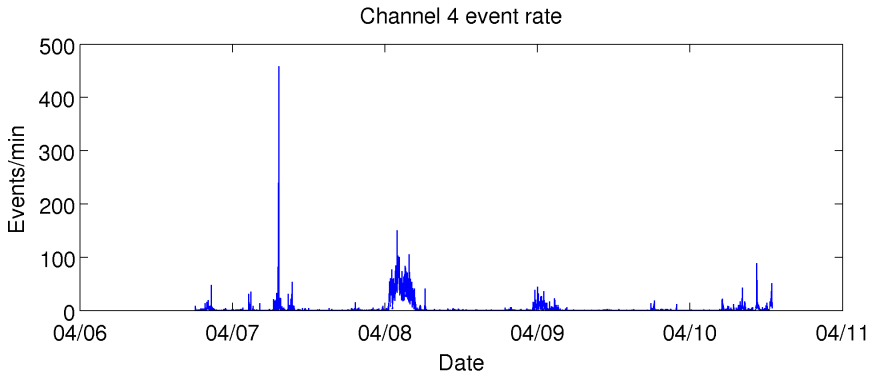


Figure 3.2: Observed event rate of channel 4 during the preliminary experiments.

(in events per second) an acquisition system must offer in order to be able to parametrize all occurring events, regardless of the current level of AE activity.

- The *average event rate* defines the required communication bandwidth (in events per second), given there's enough buffer storage available to compensate for event bursts.

For both of these values, the worst case that occurred during the preliminary experiments was chosen for the system specification in order to ensure normal operation under heavy load. As for the number of occurred events, the sensor at position number four showed by far the most intense AE activity. During the four days of experimentation, a peak burst of 300 events in one second was observed. On a minute scale, the peak event rate was at 480 events per minute (c.f. Figure 3.2). However, generally the activity peaks did not exceed 100 events per minute. The average event rate over the whole experiment was in the order of only a few events per minute. Again, sensor position four showed the highest overall activity level with an average of 3 events per minute.

Generally, the event rate observed is highly dependent on data acquisition parameters, i.e., threshold and posttrigger time: Choosing a higher threshold will decrease the number of events observed while also decreasing the overall system sensitivity. Increasing the posttrigger time will concatenate events, that would have been detected as distinct events otherwise. A suitable compromise has to be found for every deployment, depending on the activity characteristics expected.

Frequency Bandwidth

The frequency bandwidth of the observed signals depends heavily on the sensor devices used for measuring. As piezoelectric sensors generally show distinct resonances in their frequency response, they already filter the mechanical input signal significantly. It is therefore important to know the type of AE activity that is expected to be observed. As the speed of sound in (fractured) rock is relatively low (i.e., about 3600 m/s) compared to, e.g., steel due to its lower density, the expected AE frequencies are rather low, i.e., below 100 kHz ([30], chapter 11). In our preliminary experiments, piezoelectric sensors of the type R6α by Physical Acoustics Corporation were used [41]. The sensors' frequency response is measured in V/(m/s), i.e., output voltage as a function of the rock's speed of movement. The peak resonance of the R6α sensors lies at a frequency of 55 kHz, with a loss of less than 10 dB (V/(m/s)) in a range of 35 to 100 kHz. This also matches the observed average event frequencies on Jungfraujoch.

Event Length

Event length, together with the sampling rate, determines memory buffer size requirements in order to store event samples. In the preliminary experiments, the observed event length was generally in the order of 1 up to 5 ms. However, also event lengths of up to 180 ms have been detected. Again, this parameter heavily depends on the set threshold and posttrigger time: A higher threshold or a longer posttrigger time would have resulted in more, but shorter events.

Amplitude Dynamic Range

The highest event amplitudes observed in the preliminary experiment were 104 dB. As for parametrization, every signal or pulse not exceeding the threshold of 35 dB is classified as noise, the dynamic range of the observed signal is about 70 dB.

The event amplitudes' dynamic range is important in order to specify the necessary ADC resolution. As a result of quantization, the signal to noise ratio of an ideal ADC with a resolution of N bits is given as

$$\mathrm{SNR}_{\mathrm{dB[MAX]}}(N) = 6.021\,\mathrm{dB} \cdot N - 1.763\,\mathrm{dB} \quad [42]$$

Thus, a sampling resolution of at least 12 bits is needed to cover the given dynamic range, as $\mathrm{SNR}_{\mathrm{dB[MAX]}}(12) = 70.5\,\mathrm{dB}$.

In order to get a better impression of the effects of lowering the bit resolution, the captured waveforms have been resampled and reparametrized. Figure 3.3

compares the parametrization errors to the original parametrization when lowering the sampling resolution from 24 bits to 20, 16 and 12 bits. It clearly shows that a signal quantization of 12 bits introduces significant errors in the signal parametrization process, whereas a resolution of 20 bits performs almost equally to the full resolution of 24 bits. A quantization resolution of 16 bits offers a good trade-off between complexity and introduced error, as still for 85% of the expected events, all parameters show an error of less than 10%. The theoretical dynamic range of an ideal 16 bit ADC is given as $\mathrm{SNR}_{\mathrm{dB[MAX]}}(16) = 94.6\mathrm{dB}$.

Sampling Rate

The second important parameter determining the error introduced by sampling and quantizing an analog signal is the sampling rate used. Higher sampling rates improve the temporal resolution of the measurement and allow to capture higher signal frequencies, but on the other hand thereby also increase the required signal processing capabilities. As already discussed in Section 3.1.2, the expected AE signal frequencies are in the range of up to 100 kHz. According to the Nyquist-Shannon sampling theorem [43], the sampling frequency must be at least twice the highest frequency occurring in the measured signal in order to capture the signal unambiguously (i.e., without aliasing effects). Sampling at the Nyquist rate would allow to reconstruct the measured signal perfectly, given that it is strictly bandlimited to the specified bandwidth. Nevertheless, as signal reconstruction would introduce significant processing overhead compared to parametrizing the sampled signal, higher sampling rates are needed in order to minimize the error introduced. Figure 3.4 shows the effect of downsampling the measured events, originally sampled at 1 MHz.

Obviously, sampling at the Nyquist rate of 200 kHz introduces significant errors to the event parametrization. As expected, deeper analysis also showed that this error is higher for events with average frequencies close to 100 kHz and lower for events with comparably low average frequencies. Sampling at 900 kHz already introduces notable errors, while lowering the sampling rate from 900 to 500 kHz, the loss of parametrization precision is comparably small. Thus, a sampling rate of 500 kHz seems to perform reasonably well, again introducing parametrization errors of less than 10% for more than 85% of all occurring events.

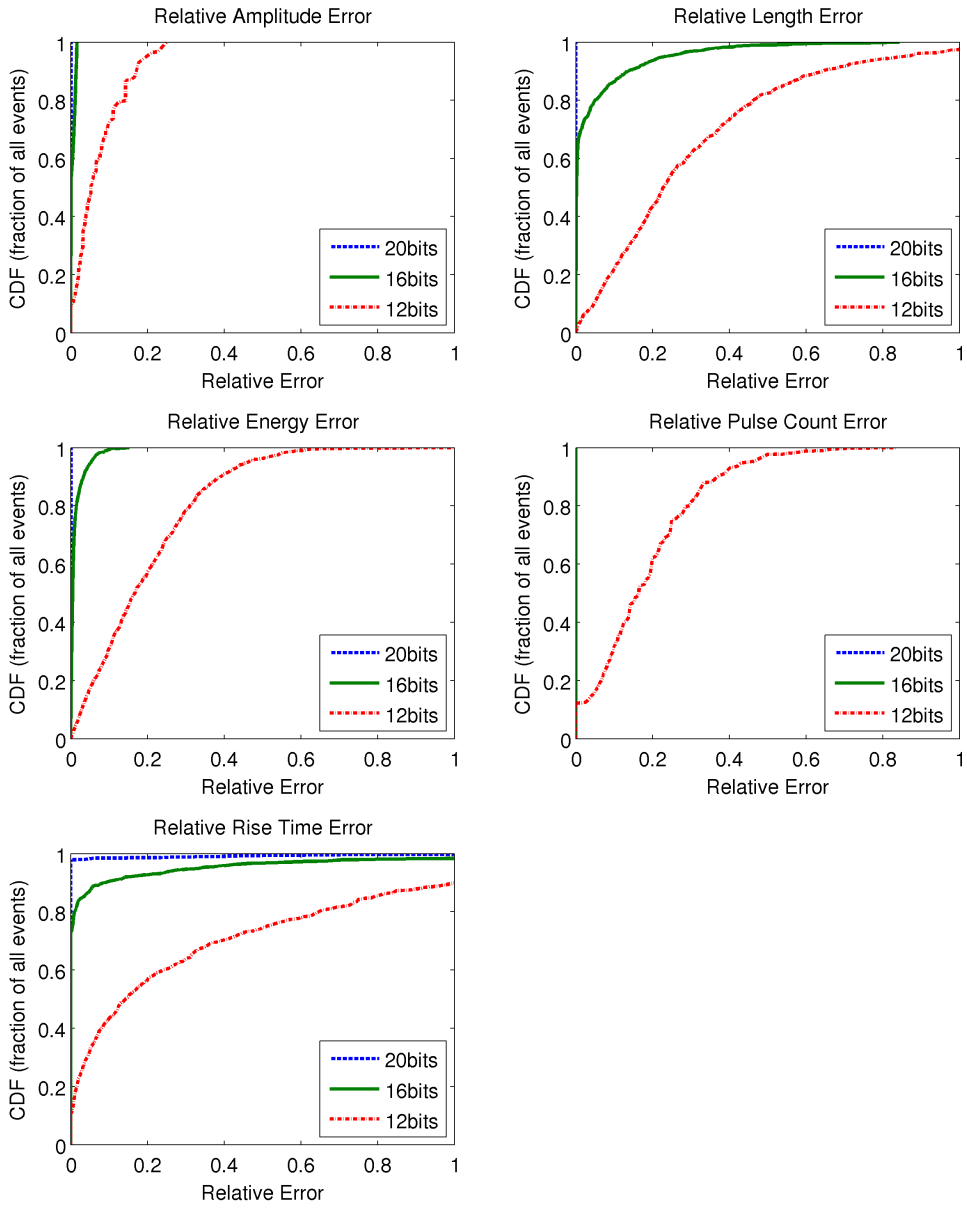


Figure 3.3: The effect of lowering the ADC bit resolution on the captured AE events. The reparametrization results are compared to the original result with a full resolution of 24 bits.

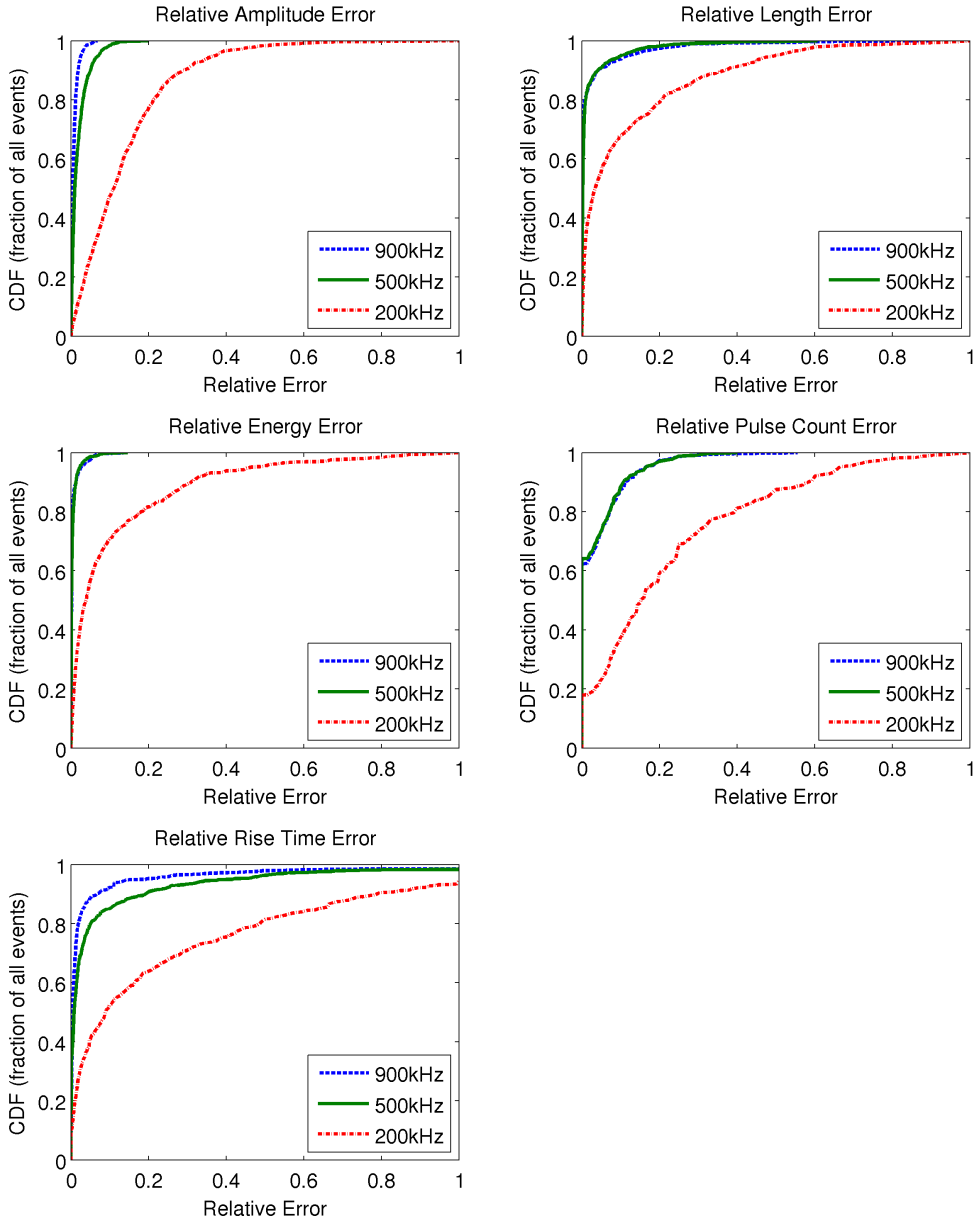


Figure 3.4: The effect of downsampling the captured AE events. The reparametrization results are compared to the original result sampled at 1 MHz.

3.2 Intented Experimental Setup

Having studied and analyzed the Jungfraujoch data [40], the geoscientific PermaSense partners defined an experimental setup for studying AE activity. Specifically, the measured AE activity shall be correlated with the already existing temperature and moisture measurements, giving further insight into the role of water in, e.g., crack formation.

As rock moisture and temperature are measured in multiple depths, it is also desirable for the AE platform to sense not only general activity, but also give a notion of how deep the event source was located. A planar location of the events is however of smaller interest, as signal attenuation in fractured rock is comparably high and thus AE signals are not expected to travel over distances in the order of meters or more. Therefore, event location requirements have been limited to linear depth location, requiring at least two AE sensors mounted in different depths. AE source depth can then be estimated by measuring the arrival time difference of an AE signal at the two sensors. Furthermore, the event rise time at both sensors also gives information about the source location, i.e., the longer the rise time, the further was the AE source located.

Measuring the arrival time difference between the two sensors requires exact time synchronization in the order of microseconds: As the speed of sound in rock is about 3000 m/s and the sensors will be mounted at a distance of approximately one meter, the arrival time differences will be in the order of less than one millisecond. As in the Dozer based PermaSense network, no exact time synchronization can be achieved between two sensor nodes, both AE sensors belonging to one location need to be read and processed by a single node in order to achieve the necessary temporal precision. Figure 3.5 therefore sketches the intended experimental AE measurement setup.

3.3 Network Bandwidth Considerations

So far, the data streams measured in PermaSense have been transferred to the backend completely, meaning that each and every measured sample has finally been stored in the PermaSense database and could be retrieved from there to be analyzed. Naturally, this approach offers the highest degree of reliability as well as flexibility in data analysis, as all processing steps can be done offline (i.e., not in real time) and non-destructively, i.e., without loosing any data. Currently, an average PermaSense node generates less than 100 Bytes of payload data per minute. When considering the findings and requirements from Sections 3.1 and 3.2 however, it is easy to see that a reasonable AE sensing platform will produce much more data:

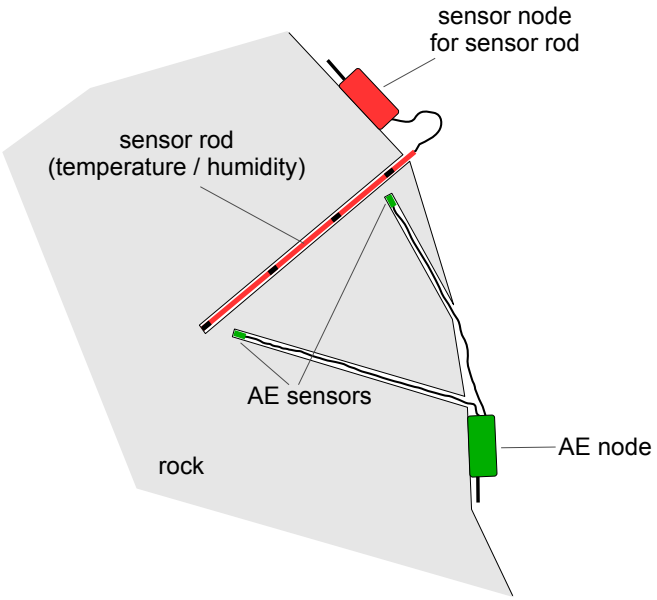


Figure 3.5: Intended experimental setup of an AE measurement location

$$\begin{aligned}
\text{Bandwidth}_{\text{all samples}} &= N \cdot f_s \cdot w \\
&= 2 \cdot 500 \cdot 10^6\,\text{s}^{-1} \cdot 16\,\text{bits} \\
&= 16 \cdot 10^6\,\text{bits/s} \\
&= 2\,\text{MB/s} \\
&= 120\,\text{MB/min}
\end{aligned}$$

where N denotes the number of channels, f_s denotes the sampling frequency and w denotes the bit width per sample. Thus, given that the AE sensing board should integrate into the existing PermaSense WSN infrastructure, streaming all measured AE data samples to the backend is not feasible.

A first step of reducing the amount of data to be transferred would be to only forward raw samples when an event (i.e., a signal that exceeds a set threshold) has been detected. Again referring to the findings from Section 3.1, we would expect an average event rate of about two events per minute, each event having a length of about 2 ms. In this case, the required network bandwidth for serving one AE sensor node would be

$$\begin{aligned}
\text{Bandwidth}_{\text{event samples}} &= \lceil \text{Bandwidth}_{\text{all samples}} \cdot f_e \cdot l_e \rceil \\
&= \lceil 16 \cdot 10^6 \,\text{bits/s} \cdot \frac{2}{60\,\text{s}} \cdot 2 \cdot 10^{-3}\,\text{s} \rceil \\
&= 134\,\text{Bytes/s} \\
&= 8040\,\text{Bytes/min}
\end{aligned}$$

with f_e denoting the expected event frequency and l_e denoting the expected event length. Obviously, only forwarding event data samples would reduce the required network bandwidth significantly. When aggregating data streams at a sinknode or over multiple WSN hops however, the required bandwidth for one point to point link is much higher than the amount of data produced by a single node as Figure 3.6 illustrates:

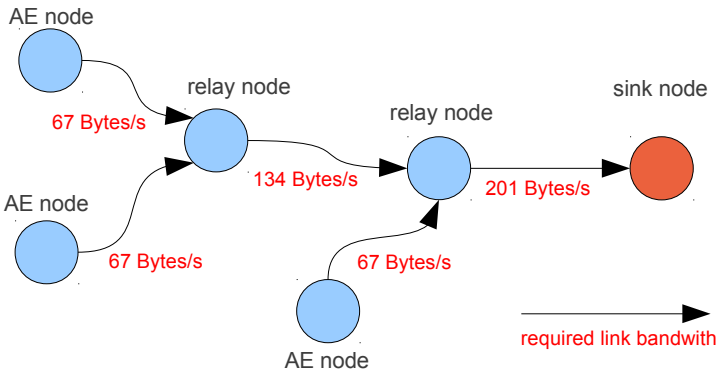


Figure 3.6: Increasing link bandwith requirements in multihop WSNs

Thus, the probability of congesting the PermaSense WSN with AE data streams is still high, making the transmission of raw samples undesirable from a technical perspective.

However, when only considering the transmission of the event parameters discussed in Section 2.2.2, the required bandwidth can again be reduced drastically:

$$\begin{aligned}
\text{Bandwidth}_{\text{event parameters}} &= f_e \cdot s_p \\
&= \frac{2}{60\,\text{s}} \cdot 23\,\text{Bytes} \\
&= 46\,\text{Bytes/min}
\end{aligned}$$

where f_e again denotes the expected event frequency and s_p denotes the maximum payload size of one Dozer packet in PermaSense, assuming that all event parameters fit into one Dozer packet. Even with the worst case average

event rate of three events per minute, only 69 Bytes of data per minute would be produced. Thus, the required network bandwidth for an AE sensing platform that only transmits the event parameters is of the same order as for the existing sensors. This approach seems to be the most promising in order to achieve a seamless and tight integration into the existing PermaSense infrastructure.

3.4 Functional System Specification

The findings from Sections 3.1 to 3.3 formed the basis to define the following functional specification for a PermaSense AE sensor node:

- The AE sensor node shall integrate seamlessly into the existing PermaSense WSN infrastructure. Specifically, it shall communicate its data over the Dozer WSN.
- Two piezoelectric sensors shall be sampled with a sampling rate of 500 kHz and a quantization resolution of 16 bits. Input signals with frequencies between 20 kHz and 100 kHz shall be captured.
- The sensor data shall be processed directly on the sensor node. The processing must thereby respect the following priorization:

  1. Count the occurring events and transmit the cumulative number over a period of two minutes. This counter value must be accurate and up to date under all circumstances, i.e., also under high AE activity load.
  2. Parametrize the occurring events and transmit the event parameters. If AE activity load is too high to parametrize all events, parametrization may be done on a best effort basis.
  3. Store the raw event samples on an internal storage device for later offline inspection and analysis if resources are not busy with counting and parametrizing events.

  The processing must behave robust under high AE activity, i.e., if not all detected events can be parametrized and / or stored, the system must not run into an illegal state and return to normal operation when AE activity decreases.
- The two channels' measurements must be temporally correlatable with an accuracy of one sample.
- The threshold and posttrigger values shall be adaptable in operation, i.e., with command messages sent to the AE sensor node.

- The most important system voltages as well as on board temperature and humidity shall be measured and transmitted to the backend every two minutes.
- Energy consumption shall be minimized as far as possible. Multiple power supply domains shall be switchable by the user in order to optimize the system's energy efficiency. The system shall run at a supply voltage of 12 V DC. If desired, a photovoltaic power supply may be used instead of batteries.
- The AE sensor must feature an operating range from -40 to +65°C, with a max. change rate of 5°C per minute.

4

High Data Rate Sensing in Low Bandwidth WSNs

Before going into implementation details of the developed AE sensing platform, this chapter introduces some general design concepts. Specifically, a generic architectural concept for sensing data with high sampling rates in a low bandwidth WSN environment is presented, that may also be used for applications other than AE sensing.

4.1 Problem Statement

Most WSN architectures and protocols are optimized to maximize unattended lifetime. As many sensor nodes are battery powered, energy is often a key factor limiting the independent operating timespan. However, especially in environmental monitoring, the physical processes captured often expose relatively slow temporal variations, requiring only low sampling rates in the order of a few samples per minute to measure them adequately. Thus, many WSN operating approaches aim to save energy by putting the sensor nodes into some low power mode for most of the time, regularly waking them up for performing short data sampling and/or communication sequences. This inherently also decreases the available network bandwidth and node reactivity.

Integrating measurements of processes that - due to their faster varying nature - require continuous sampling at higher rates thus produce data in the

order of multiple MB/second thus requires novel architectural and operational approaches. As the available bandwidth does not allow to stream all measured data through the WSN, some form of data reduction must take place directly on the sensor node itself. Besides compressing the data to transmit over the network with generic statistical data compression algorithms such as Huffman coding [44], some more specific approaches are possible that are discussed in the following.

4.2 Raw Data Reduction Strategies

The simplest possibility to reduce the required bandwidth for high data rate measurements is to reduce the amount of raw data samples that is transmitted over the WSN. To this end, two basic approaches are possible which are presented in the following sections.

4.2.1 Duty-Cycling

Bandwidth requirements for high data rate measurements can be lowered by reducing the timespan, during which the phenomenon of interest is sampled, i.e., only sampling for short periods with intermediate longer periods where no sampling takes place. E.g., consider an application where sensor data is sampled with a sampling rate of $f_s = 1\,\text{kHz}$ and a quantization resolution of $w = 8\,\text{bits}$. Thus, during sampling, $b_r = f_s \cdot w = 1\,\text{kBytes/second}$ of raw data are being produced. When reducing the sampling period to one minute per hour, the average required bandwidth is only $\frac{b_r}{60} = 16\,\text{Bytes/second}$; the required bandwidth could be reduced by the same factor as the sampling time was reduced. The advantage of this method lies in its exact predictability of the bandwidth requirements, comparable to measurements with much lower constant sampling rates.

In PermaSense, the current efforts to integrate GPS measurements are based on a duty-cycling approach (c.f. [6, 7]).

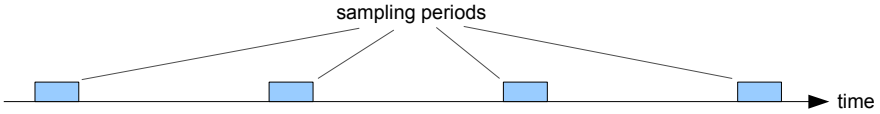


Figure 4.1: Basic principle of sampling time reduction: Periodical short periods of sampling reduce the average data rate.

4.2.2 Raw Data Classification

Although providing excellent predictability of bandwidth requirements, sampling time reduction is only suitable for special measurement cases. Often, it is unknown in advance *when* data should be sampled, as the processes of interest (e.g., AE activity) occur spontaneously and would not be captured by periodic short measurement sequences. Thus, as already depicted in Section 3.3, the amount of transmitted raw data may also be reduced by some sort of data classification, basically deciding for each sample whether it is worthy to be sent over the network or not. E.g., in the case of AE measurements, this decision process is fairly simple, as all samples that belong to an event are forwarded and all others are discarded. Similar simple decision processes might be defined for other phenomena, minimizing the required on-node processing power.

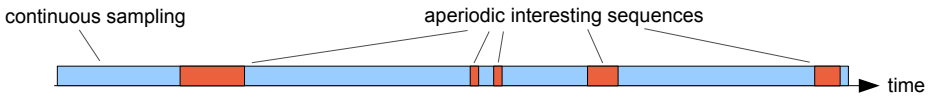


Figure 4.2: Basic principle of raw data classification: Data is sampled continuously, but only interesting sequences are forwarded.

The network bandwidth required with this approach can generally only be estimated, as it depends on the fraction of samples that need to be forwarded, which is unknown for spontaneous processes. With p_f denoting the expected fraction of samples to forward and b_r denoting the raw data rate being produced when sampling, the required network bandwidth is equal to $b_r \cdot p_f$. For the specific case of AE sensing, a more detailed example is presented in Section 3.3.

4.3 Direct Data Aggregation

As already discussed in Section 3.3, for certain applications, reducing the amount of transmitted raw data is still not sufficient in order to meet given network bandwidth constraints of a low power WSNs. Either valuable data would be discarded, or the WSN would quickly suffer congestion. Thus, for such applications it is desirable to perform substantial parts of the data processing directly on the sensor node itself, aiming to reduce the amount of transmitted data as much as possible without discarding information of interest. Basically, no raw samples are transmitted over the network at all, but merely meaningful aggregates thereof that can be used for further analysis. In the case of AE sensing for example, the events may be detected and

parametrized directly on the sensor nodes, leaving only the AE parameters to be transmitted over the WSN.

Again, as the processes of interest are assumed to occur spontaneously, the required network bandwidth can only be estimated based on expected activity rates. A bandwidth calculation example for the case of AE sensing is presented in Section 3.3. Naturally, direct data aggregation especially makes sense if local processing can reduce the data by multiple orders, as it is the case for AE event detection and parametrization.

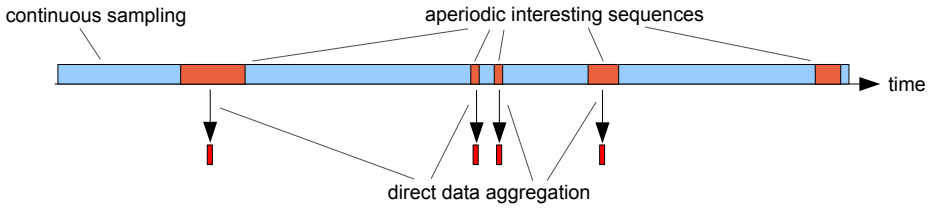


Figure 4.3: Basic principle of direct data aggregation: Data is sampled continuously, interesting sequences are first aggregated and then forwarded.

In order to effectively employ such an approach, a precedent solid understanding of the phenomenon studied is required: Opposed to the analysis of raw data that has arrived at the backend, data aggregation happens very early in the process. This prohibits the application of any other form of analysis on the samples measured than the predefined aggregation functions. Nevertheless, for certain high data rate applications, this may be the only approach to make a successful WSN integration feasible.

However, the low power optimization trend in WSN development did not only concern the operating mode of the sensor nodes: Also the sensor node hardware itself is optimized to operate very power efficiently. This generally also implies that the processing capacity of most sensor nodes is very limited, tailored to sense and forward data and at most perform some very basic aggregation operations. In the context of direct data aggregation for high data rate measurements, common sensor nodes simply lack the computation power to perform the necessary signal processing tasks. On the other hand, building a heterogeneous WSN that integrates commodity sensor nodes as well as higher performance platforms presumably implies a lot of hassle with cross-platform compatibility issues, complicating network management and monitoring. As an approach to address this dilemma, we propose the concept of "processing sensors".

4.3.1 Processing Sensors

The concept of processing sensors basically attempts to insert an additional layer of abstraction between data acquisition and data transmission, significantly reducing the amount of data to forward and thereby virtually reducing the sensor's data rate. This is achieved by adding an additional processing element between the physical sensor and the sensor node platform, serving as data acquisition controller, data processing instance and sensor data interface at the same time. Ideally, this data processing platform keeps its own state and is only loosely coupled to the sensor node, separating concerns as much as possible in order to allow virtually independent operation. Figure 4.4 illustrates this basic principle:

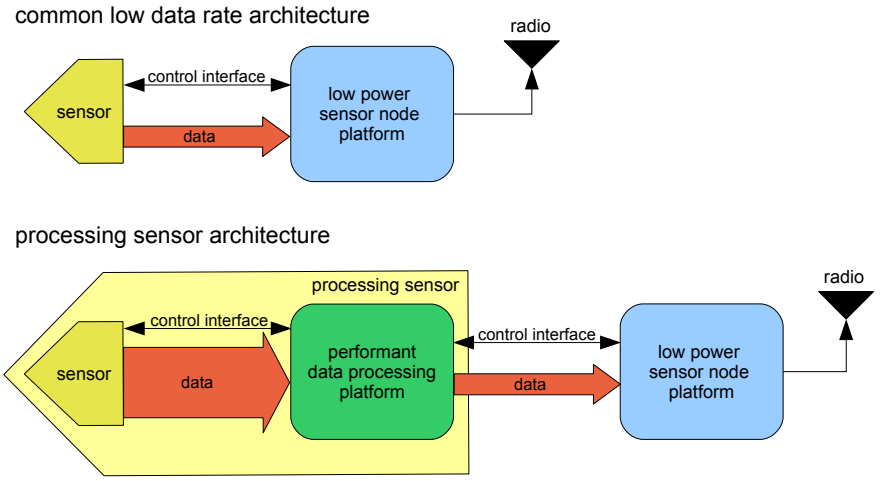


Figure 4.4: The processing sensor architecture. From a sensor node platform's perspective, a processing sensor does not differ much from any other low data rate sensor.

The approach of processing sensors offers the following advantages over integrating higher performance computing platforms directly into the WSN:

- **Manageability**: Deploying and operating a WSN reliably is a difficult task. It becomes even harder, if the network consists of a heterogeneous mix of different platforms. Relying on a single WSN optimized platform therefore eases network management significantly.
- **Robustness and speed of development**: A command based point to point data interface is simpler to implement and less prone to cross-platform compatibility issues than a full WSN network stack.
- **Modularity**: As long as the data interface between the sensor node platform and the processing element stays unaltered, the two compo-

nents may be developed and improved independently. Changes in the WSN platform code or hardware do not affect the processing sensor platform and vice versa.

- **Predictability**: As each platform keeps and manages its own state, there's virtually no interference between data acquisition/processing and network communication. This makes it easier to predict data acquisition and processing performance as it is not influenced by network link quality and other factors related to communication.

Naturally, introducing a second processing element on a sensor node generally increases its power consumption. It is therefore important to carefully trade computation power (i.e., power consumption) versus the amount of data to transmit (i.e., network bandwidth requirements) for each individual application.

5 AEBoard Hardware Design

This chapter presents an AE sensing hardware platform called "the AEBoard", custom designed for performing AE measurements in PermaSense (c.f. Chapter 3).

5.1 Basic System Architecture

The AEBoard's system architecture basically follows the approach of a processing sensor (c.f. Section 4.3.1): An intelligent sensor reduces the amount of data to transmit drastically, such that existing WSN hard- and software can be reused as if a low-data rate sensor was interfaced.

Specifically, the AEBoard features a TinyNode WSN platform as system master, controlling all sensors as well as taking care of the WSN communication. The TinyNode receives AE parameter data over its serial interface from a processing AE sensor, consisting of a slave microprocessor and the actual AE sensing circuitry. Furthermore, two SD memory cards are included: One SD card is attached to the TinyNode to serve as network packet backlog memory in case of missing network connectivity or AE data bursts. The other SD card may be used by the slave processor to store raw AE data of occurring events (c.f. Section 3.4). Figure 5.1 sketches this basic architecture.

Additionally, a combined sensor for temperature and humidity has been attached to the TinyNode for system health monitoring as well as measurement

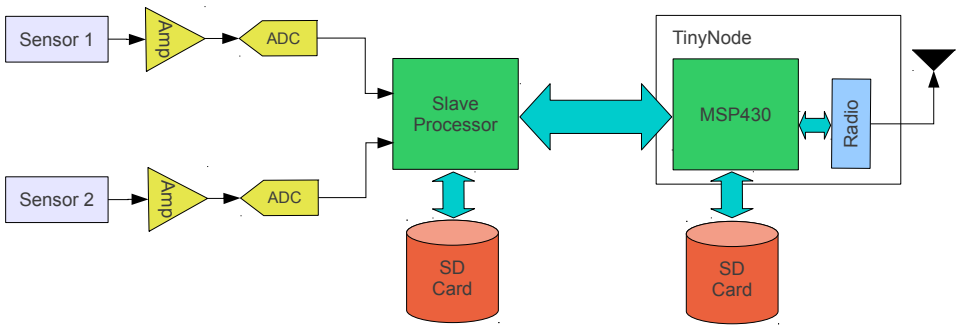


Figure 5.1: AEBoard architectural concept

calibration. The TinyNode also monitors several voltage levels and supply currents with its internal ADCs and is able to power cycle the slave processor (c.f. Section 5.4).

5.2 AE Sensor Key Components

As no similar measurements have been done in PermaSense so far, the AEBoard's processing sensor has been designed from scratch. The following sections describe the various considerations that were taken into account when selecting the sensor's key components.

5.2.1 Slave Processor

The slave processor platform is the heart of a processing sensor: It controls the data acquisition (DAQ) process, performs significant parts of the data analysis and interfaces the actual WSN platform to forward data. Furthermore, the AE sensor slave processor has to store raw event data on its SD card. Thus, besides being able to perform the necessary signal processing in real time, the slave processor must be able to coordinate multiple internal and external data streams, as Figure 5.2 illustrates.

Therefore, a key requirement for the slave processor platform are powerful communication interfaces and a memory architecture that offers enough size and bandwidth to serve all communication interfaces virtually in parallel. Ideally, the slave processor offers true parallelism for certain tasks (e.g., parallel communication and signal processing).

Based on these requirements, candidate devices for the slave processor have been evaluated. Three platform variants have been considered:

- Digital signal processor (DPS) platforms

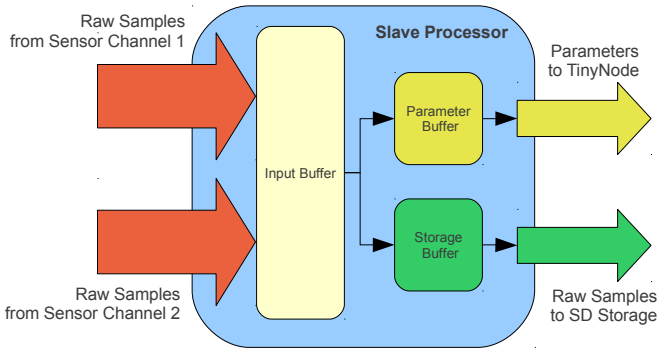


Figure 5.2: Slave processor data streams.

- Programmable logic devices, i.e., several PLDs and FPGAs
- General purpose microprocessor platforms (e.g., ARM Cortex based platforms)

Table 5.1 compares the three most promising devices (one of each platform variant) in detail.

Out of the three components presented, the ARM Cortex M3 based STM32F103RD has been selected for the following reasons:

- The STM32 offers comparably many communication interfaces, all of them offering direct memory access (DMA) capabilities. This allows the processing core to spend more capacity for signal processing instead of managing communication tasks.
- The included 64 kBytes SRAM offers enough space for buffering the various data streams.
- Native SDIO support offers high writing performance to the SD card.
- An FPGA based solution would have offered more flexibility, true hardware parallelism as well as assumably lower power consumption. However, there is no FPGA know-how in PermaSense so far. Thus, as the AEBoard shall be productively integrated into PermaSense as soon as possible, a microprocessor based approach seemed to be more compatible with further project developments.
- The STM32 microprocessors are well available at comparably low prices.

	ADSP2186M	**IGLOO AGL600**	**STM32F103RD**
General			
Type	DSP	FPGA	Microprocessor
Manufacturer	Analog Devices	Actel	STMicroelectronics
CPU			
Core	ADSP-2100	ARM Cortex M1	ARM Cortex M3
f_{max}	75 MHz	60 MHz	72 MHz
Bus width	16 bits	32 bits	32 bits
Clocking	external	external / external	external
Memory			
RAM type	SRAM	Dual-Port SRAM	SRAM
RAM size	40 kBytes	13.5 kBytes	64 kBytes
ROM type	n/a	Flash	Flash
ROM size	n/a	1 kBit	384 kBytes
Power			
Supply voltage	2.5 V - 3.6 V	1.2 V - 1.5 V	2.0 V - 3.6 V
Low power modes	1	2	4
Wakeup source	n/a	ext Pin	ext Pin
Est. Power at 60 MHz	20 mW	15 mW	20 mW
Connectivity			
SPI ports	2	configurable	3
USARTs	0	configurable	5
SDIO	0	configurable	1
max GPIOs	8	235	51
Miscellaneous			
Operating temp.	-40°C to +85°C	-40°C to +85°C	-40°C to +85°C
Price per unit	16 CHF	75 USD	12 CHF

Table 5.1: Comparison of three potential slave processor platforms. All technical values base on the corresponding datasheets [45, 46, 47].

Some preliminary tests with the STM32Discovery evaluation kit further verified that the system specifications can be met with the STM32 slave processor.

5.2.2 ADCs

When digitally capturing and processing analog data, the ADC naturally heavily influences the measurement quality achieved. Thus, choosing an appropriate ADC is crucial for good measurement quality. For the AEBoard, besides offering a quantization resolution of 16 bits at a sampling rate of 500 kHz, the ADCs have to operate between -40°C to +85°C and offer low power consumption.

Many available low-power ADCs either only offer a resolution up to 12 bits or don't achieve the sampling rates required. While for audio applications with sampling rates up to 96 kHz a broad variety of 16 bit ADCs exists, products in the 16 bit / 500 kHz class are rather rare. Finally, two candidate ADCs could be identified, which are compared in Table 5.2.

	ADS8327	**AD7980**
General		
Manufacturer	Texas Instruments	Analog Devices
Conversion		
Maximal sampling rate	500 kHz	1000 kHz
Resolution	16 bits	16 bits
Miscellaneous		
Typ. power dissipation at 500 kHz	10.6 mW	3.5 mW
Operating temp.	-40°C to +85°C	-40°C to +125°C
Interface	SPI	SPI
Price per unit	30 CHF	40 CHF

Table 5.2: Comparison of two potential ADCs. All technical values base on the corresponding datasheets [48, 49].

Because of its lower power consumption as well as the possibility to increase the sampling rate above 500 kHz if necessary, the AD7980 ADC from Analog Devices has been chosen for the AEBoard. Its output format is compatible to the well-known SPI protocol, which is well supported by the STM32.

5.3 Input Signal Conditioning Circuitry

Before sampling the AE sensor's output voltage, several signal conditioning tasks have to be performed in order to ensure optimal sampling quality:

- Filter the input signal with a bandpass filter in order to minimize

unwanted noise.

- Adapt the signal's voltage range in order to utilize the ADC's full dynamic range.

- Drive the ADC with a low-impedant signal source for maximal quantization accuracy.

Piezoelectric sensors generally produce relatively weak output signals that are not suitable to drive cables longer than a one or two meters. Therefore, the AE signals have to be amplified very close to the sensor. The Physical Acoustics Corporation offers both active sensors that already have the necessary amplifier built into the housing (e.g., the PK6I sensor [50]) and passive sensors that need to be connected to an in-line amplifier after at most two meters of cable (e.g., the R6α sensor [41] together with a IL-LP-6S preamplifier). In both cases, the preamplifier is supplied with electric power over the signal line by lifting it to a DC bias voltage of 4 to 7 V. The sensor output signal then rides on this supply voltage, featuring a maximal amplitude that is equal to the supply voltage.

On the other hand, the AD7980 ADC requires the constant reference voltages V_{ref} and GND. It then samples input voltages between GND and V_{ref} with a resolution of 16 bits. Input voltages smaller than GND or higher than V_{ref} are saturated at the respective output value [49].

Thus, the signal conditioning circuitry must bandpass the input signal to the specified frequency range of 20 kHz to 100 kHz, and adapt its voltage range such that the maximal ADC input voltage lies at V_{ref} and its minimum lies at GND potential. For this purpose, an active filter has been designed, using the ADA4841 operational amplifier by Analog Devices [51] which is recommended to drive the AD7980 ADC. The DC supply voltage is removed at the filter input by a serial capacitor, whereas a 50 Ω serial resistor decouples the signal from the supply voltage regulator. The signal conditioning circuitry's simplified schematics are shown in Figure 5.3.

Circuit simulation with LTSpice [52] was used to adequately dimension the passive components. Figure 5.4 shows the simulated filter frequency response.

For this simulation, the input source was simply modelled as an AC source with an offset voltage of 4.5 V instead of exactly modelling the sensor / preamplifier source. Thus, the 4.5 V bias voltage affects the input signal, resulting in an overall -6 dB damping in the simulation results. The LTSpice simulation files are also contained on the enclosed CD (c.f. Appendix G).

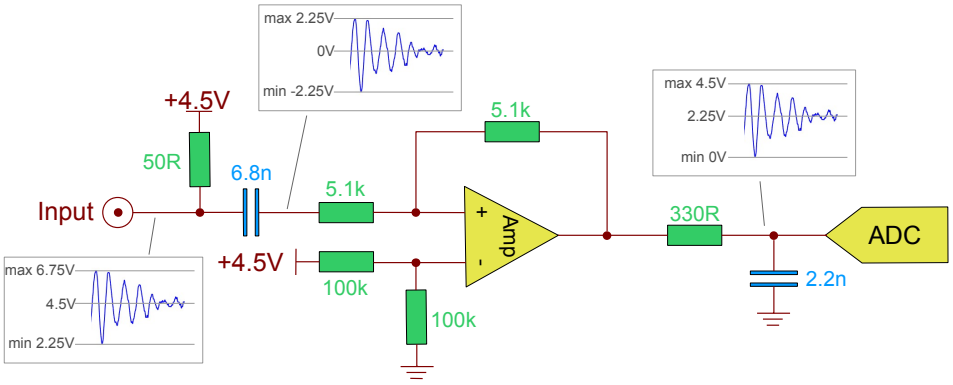


Figure 5.3: Simplified schematics of the signal conditioning circuitry for one AE channel

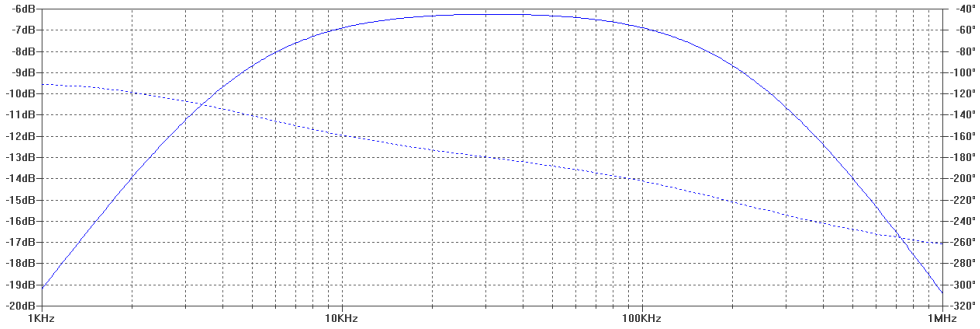


Figure 5.4: Signal conditioning circuitry frequency response, simulated with LTSpice. The solid line shows the filter amplification, which is relevant to define the usable frequency range. The dotted line shows the less important filter phase.

5.4 Power Concept

The AEBoard features various components requiring electrical power supply. In order to minimize the overall power consumption, the input voltage for each component should be as low as low as possible in it's specified supply range. Thus, power consumers on the AEBoard have been divided into the following supply voltage categories:

- **4.5 V**: The AE sensors need a supply voltage V_s of at least 4 V, additional 0.5 V have been added to provide enough headroom. Consequently, the operational amplifiers also have to be powered with the same voltage in order to support the required output range of 0 V to V_s.
- **2.8 V**: SD cards require a minimum supply voltage of 2.7 V [53]. Adding 0.1 V of headroom, the STM32 as well as the TinyNode need to be powered with at least 2.8 V in order to meet the requirements for SD card hosts.
- **2.5 V**: The AD7980 ADC requires a supply voltage of exactly 2.5 V. Thus, it requires a separate power supply circuit. Furthermore, it also requires a reference voltage V_{ref} that is equal to the maximum amplifier output, i.e., 4.5 V, as well as a digital input / output level voltage V_{IO} of 2.8 V.

Additionally, in order to gain finer control over the AEBoard's operation and save power when necessary, the three following possible operating modes have been defined:

- **Standby**: Only the TinyNode's power supply is turned on. The slave processor as well as the DAQ circuitry are cut from power supply completely.
- **Data transfer**: In addition to the Standby mode, the slave processor is supplied with power. This mode allows for data exchange between the TinyNode and the STM32, but no AE data can be sensed.
- **AE sensing**: All components are powered and operational.

Consequently, as the actual sensing circuitry shall be completely hidden from the TinyNode, it only controls the STM32's power supply. The slave processor in turn controls all DAQ components' supply and reference voltages.

In order to ensure optimal measurement quality and minimize coupling between the two channels, each input channel is biased by a separate voltage

regulator. Furthermore, the ADC reference voltage as well as the amplifier supply voltage are generated by distinct sources.

Initial estimates assumed a constant operating current of about 40 mA in the AE sensing mode. As even high capacity batteries would discharge in a few days under this load, the AEBoard is designed to be fed by a solar power supply, providing an input voltage of 12 V. For power efficiency reasons, these 12 V are converted to an intermediate voltage of 5.5 V by a switched capacitor voltage regulator. This intermediate voltage is then converted to the required voltage levels by linear regulators. Figure 5.5 sketches this power concept.

Due to a voltage drop of approximately 0.5 V at the power switch transistors used, the TinyNode and its SD card are fed with a supply voltage of 3.3 V to ensure the required 2.8 V for the STM32 after the power switch. For system health supervision purposes, voltage and current are measured at various points: Voltages and currents in the prescaling and digital master domain are sensed by the TinyNode, voltages in the sensing domain are supervised by the slave processor.

5.5 AEBoard Prototype

The hardware concept described has been implemented in an AEBoard prototype. In addition to the features already mentioned, the following components and features have been added:

- **JTAG programming and debugging ports** for both the TinyNode and the STM32. The 20 pin port for the STM32 is thereby compatible with the ST-LINK programmer by STMicroelectronics, the TinyNode's 14 pin port is compatible with various MSP430 JTAG programmer devices.
- **Reset buttons** for both the TinyNode and the STM32.
- **Serial console connector** for the STM32 which could be used to output debug messages to a PC. It is pin compatible with the TTL-232R-3V3 USB TTL serial cable from FTDI chip [54].
- **Three debug LEDs** for the STM32.
- **Serial communication debug LEDs** that indicate data flow between the TinyNode and the STM32.
- **Two shared GPIO connections** between the TinyNode and the STM32 in addition to the serial interface.

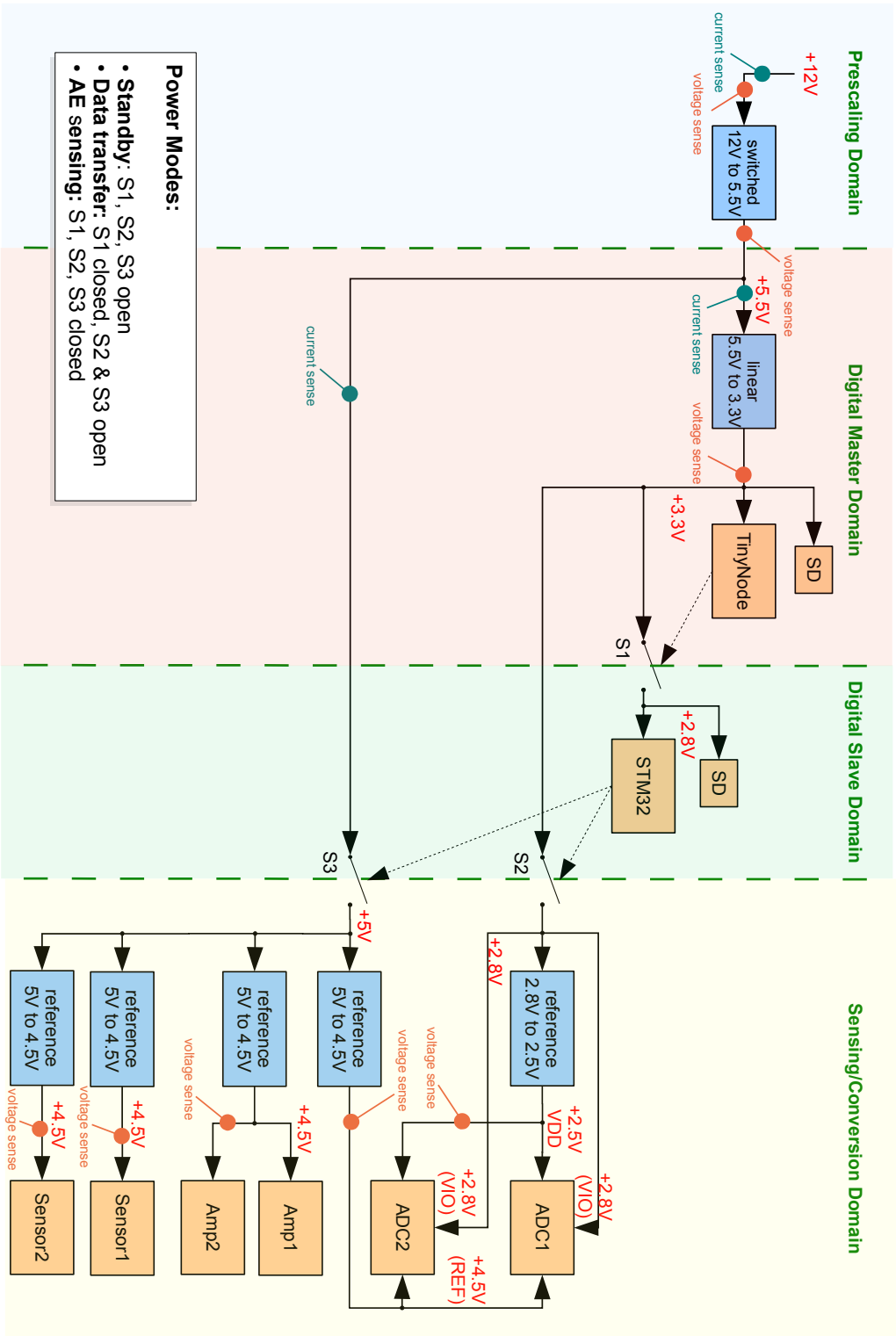


Figure 5.5: AEBoard power concept

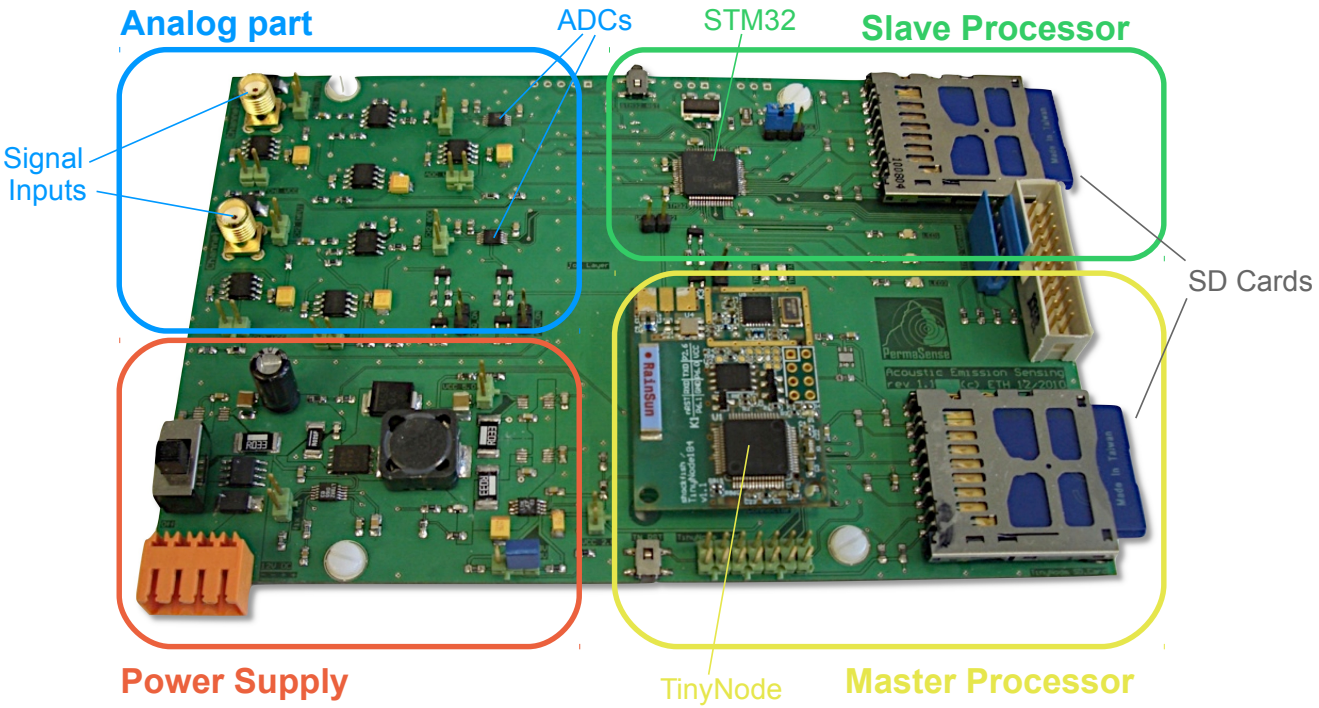


Figure 5.6: The AEBoard prototype.

- **Switchable boot mode for the STM32** to recover the processor from corrupted program memory by booting it from the factory bootloader.
- **Switchable processor supply voltage** to 2.8 V or 3.3 V. This feature has been removed for further hardware revisions, as a supply voltage of 3.3 V is required for proper functionality (c.f. Sections 5.4 and 5.5.1).
- **ESD protection diodes** at the signal inputs.
- **Test pins** at various points to measure supply and reference voltages manually.

For detailed design schematics and PCB layout figures of the so far manufactured hardware revision 1.1, please refer to Appendix E.

5.5.1 Changes for Revision 1.2

Although the first AEBoard prototype basically worked as expected, a few changes have been made for further hardware revisions (the component des-

ignators refer to the respective schematics). These changes have already been included in the latest schematics and layout files that may be found on the enclosed CD (c.f. Appendix G) or in the PermaSense SVN repository under `/trunk/pcb/projects/0011_acoustic_emission`.

- Pins 5 and 13 of the STM32's JTAG interface have been interchanged in order to allow JTAG access. This bug could be circumvented by either patching the access ribbon cable or using the ARM specific single wire debug (SWD) protocol instead of JTAG for programming and debugging.
- The current supervision ICs' supply voltage has been corrected from 5.5 V to 3.3 V. As a consequence of the wrong supply voltage, the current supervision circuits in revision 1.1 are not usable.
- An LED (D10) has been added to indicate the the presence of supply power.
- A pull-down resistor (R77) has been added to the STM32 reset circuitry, preventing spontaneous resets due to a floating reset control line.
- The TinyNode's supply voltage has been fixed to 3.3 V by removing jumper W2 as well as resistor R20, as an output voltage of 2.8 V at the linear voltage regulator U9 is not sufficient for powering the STM32's SD card after the voltage drop of 0.5 V at the switching transistor (c.f. Section 5.4).

6

PermAE Software Design

This chapter describes a distributed piece of software called "PermAE", that runs on the AEBoard and meets the system specifications described in Chapter 3.

6.1 General Concept

Consequently following the principle of a "processing sensor" (c.f. Chapter 4), PermAE is split into two virtually independent parts: One part is running on the STM32 slave processor and is responsible to control and supervise the data acquisition process, process the raw data, forward detected AE event parameters as well as status messages to the TinyNode and store raw event samples on its SD card. The other part of PermAE runs on the TinyNode and basically controls the slave processor, manages the WSN connection, forwards received AE parameter and status messages to the data sink and monitors air temperature and humidity. The two devices communicate over a UART port by a set of well defined commands and messages as presented in the following section. This serial port, together with the TinyNodes capability of power cycling the STM32, forms the only interface between these two components, allowing to develop and test many of the respective internal matters independently (c.f. Figure 5.1).

6.2 PermAE Messages

As producing data messages is the main purpose of a sensor node in PermaSense, the expected output messages are specified first. Note that from an end user's perspective, it does not matter whether a message was produced by the STM32 or by the TinyNode. Rather, a user should not even notice that two independent processors are involved.

6.2.1 PermAE Output Messages

Table 6.1 briefly outlines the messages generated by PermAE and finally stored in the backend database.

Message	Description	Contents
aedata	Information about *one* measured AE event.	Occurrence time information, event parameters, posttrigger used, threshold used and measurement channel
aestatistics	Event counter values, aggregated over two minutes.	Total events measured, total events parametrized, total raw events stored (c.f. Section 3.4)
aedaqhealthdata	AEBoard health and system status data.	Monitored supply voltages and currents, monitored reference voltages, number of free raw event blocks on the SD card, power state of the STM32 (on / off).
nodehealth	PermaDozer standard message representing the TinyNode's status	Packet count, system voltages, air temperature and humidity, SD card status and TinyNode uptime.

Table 6.1: PermAE output messages

Additionally, further standard PermaDozer messages (eventlogger, rssi, statecounter) are generated. For detailed format specifications, refer to Appendix C. The values' units are described in detail in Appendix D.

6.2.2 PermAE Dozer Commands

According to the functional system specification in Section 3.4, threshold and posttrigger values must be adaptable from the backend. Furthermore, data acquisition (i.e., the whole processing sensor) may be power cycled manually. To this end, PermAE uses the Dozer beacon mechanism to receive commands. As the maximum number of Dozer commands is limited to 16 per deployment, only one new command type ("AEBOARD_CTL_CMD") has been defined. The command argument then encodes the message that is sent to PermAE as specified in Table 6.2:

Command	**Description**	**Argument to send**
DAQ_OFF	turn data acquisition off	256
DAQ_ON	turn data acquisition on	512
SET_THR	set the threshold value	1024 + threshold [dB]
SET_PTRG	set the posttrigger value	2048 + posttrigger [samples]

Table 6.2: Dozer command messages. To set a threshold of, e.g., 53 dB, the argument must be set to $1024 + 53 = 1077$. To set a posttrigger value of, e.g., 600 samples, the argument must be set to $2048 + 600 = 2648$.

Note that the threshold value is limited to 7 bits and therefore to a maximum value of 127 dB. Likewise, posttrigger values are limited to 11 bits or a maximum value of 2047 samples.

The Dozer commands can easily be sent to a node running PermAE by utilizing the *dozer_command* virtual sensor in the PermaSense webinterface (c.f. Section A.5).

6.3 Onboard Communication Protocol

As already outlined, the two parts of PermAE communicate over a serial UART interface. Communication is achieved by a simple command based protocol. Generally, every communication round is started by the TinyNode sending a command. The STM32 may then respond according to the following protocol. In order to request an exchange of messages (e.g., to forward AE event parameters to the TinyNode), the STM32 raises one of the shared GPIO pins (called the "ARM_COMM_FLAG") and waits for the TinyNode to start communicating. All predefined commands and messages have a length of 1 Byte.

The onboard communication protocol uses a simple acknowledgement scheme. Table 6.3 defines the messages used. If any party sends a NACK message

at any point in a message exchange, the communication flow is aborted and the protocol must be reset on both sides.

Hex	ASCII	Description
0x41	A	acknowledge (ACK)
0x4E	N	not acknlowledge (NACK)

Table 6.3: Onboard ACK and NACK messages

To initiate a communication round, the TinyNode sends one of the commands specified in Table 6.4.

Hex	ASCII	Description
0x53	S	start data acquisition
0x4F	O	stop data acquisition
0x45	E	set threshold value
0x49	I	set posttrigger value
0x4D	M	send a message

Table 6.4: Onboard command messages

When requested to send a message, the STM32 responds with a message indicating the data message type to follow. Table 6.5 specifies the possible types.

Hex	ASCII	Description	Size	Dozer equivalent
0x45	E	AE event data	23 Bytes	aedata
0x53	S	AE statistics	6 Bytes	aestatistics
0x48	H	health data	14 Bytes	n/a

Table 6.5: Onboard data message types

AE event data messages and AE statistics messages are equivalent to the respective Dozer messages and may be directly forwarded by the TinyNode. The health data is first extended by additional values measured by the TinyNode and only then forwarded in an "aedaqhealthdata" message (c.f. Appendix C).

The possible valid communication flows are finally specified in Figure 6.1.

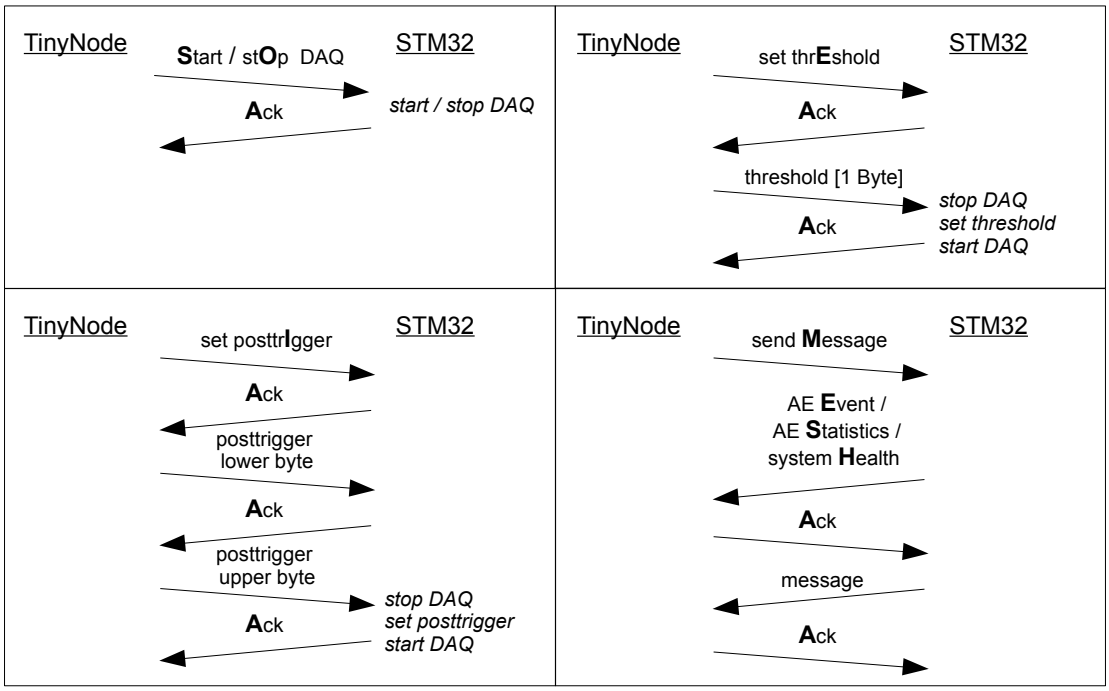


Figure 6.1: Valid onboard message flows

Both parties must send a NACK message if either an invalid command / message was received or if an expected response has not arrived within 50 ms. The protocol state must then be reset on both sides and communication restarts.

6.4 STM32 Implementation

The STM32 slave processor is mainly responsible for acquiring and processing AE sensor data as well as forwarding detected event parameters. Furthermore, as many raw events as possible shall be stored on the SD card attached and the DAQ system's health shall be supervised regularly. As all these tasks must virtually run in parallel, the FreeRTOS real time operating system kernel [55] has been utilized to implement this part of PermAE efficiently.

FreeRTOS features a fully preemptive task model with execution timeslots of a fixed maximum length. Tasks preferably communicate over FreeRTOS' FIFO queues. Reading from an empty queue or writing to a full queue may or may not block the respective task depending on the programmer's decision. When either an execution timeslot has expired or after a read or write

operation on a queue, the scheduler gives control to the highest priority task that is not blocked, explicitly delayed or suspended. For more information about the FreeRTOS execution model, please refer to the FreeRTOS documentation [56, 57] on the enclosed CD.

The STM32 application has been split into the following tasks:

- The **data analysis controller task** enables and disables the data acquisition process. It however only does so when receiving an according command from the TinyNode via the communication task.
- Two **data analysis tasks** analyze the incoming raw data samples, detect AE events and parametrize them. Each data analysis task is responsible for analyzing one input channel's data.
- The **communication task** manages the serial interface to the TinyNode. It therefore implements the communication protocol specified in Section 6.3.
- The **storage manager task** manages the SD card resource and writes incoming raw event sample buffers onto the SD card.
- The **event statistics task** reads and resets the event counter values every two minutes and generates the according aestatistics messages. Likewise, the **voltage supervision task** measures the supervised system voltages with the internal ADCs and generates the system health messages every two minutes.

Tasks are listed *according to their assigned priorities*, i.e., the data analysis controller tasks has the highest priority and suspends all other tasks, whereas the event statistics and voltage supervision tasks have the lowest priorities. These task priorities directly reflect the output data priorization specified in Section 3.4: Data acquisition and event count is done with highest priority, forwarding of parameter is done with medium priority and storage of events on the internal SD card is done with low priority. System supervision tasks are finally carried out with lowest priority as they are not critical for the detection of AE activity. Figure 6.2 depicts the PermAE application structure on the STM32.

For efficiency reasons, all necessary data buffers are allocated statically during initialization. Thus, the tasks only pass pointers to these static buffers via FreeRTOS queues. Consequently, PermAE also requires feedback queues, containing pointers to empty / processed buffers that may be filled with new data. These feedback queues have been omitted in Figure 6.2 for simplicity. Table 6.6 displays PermAE's memory partition.

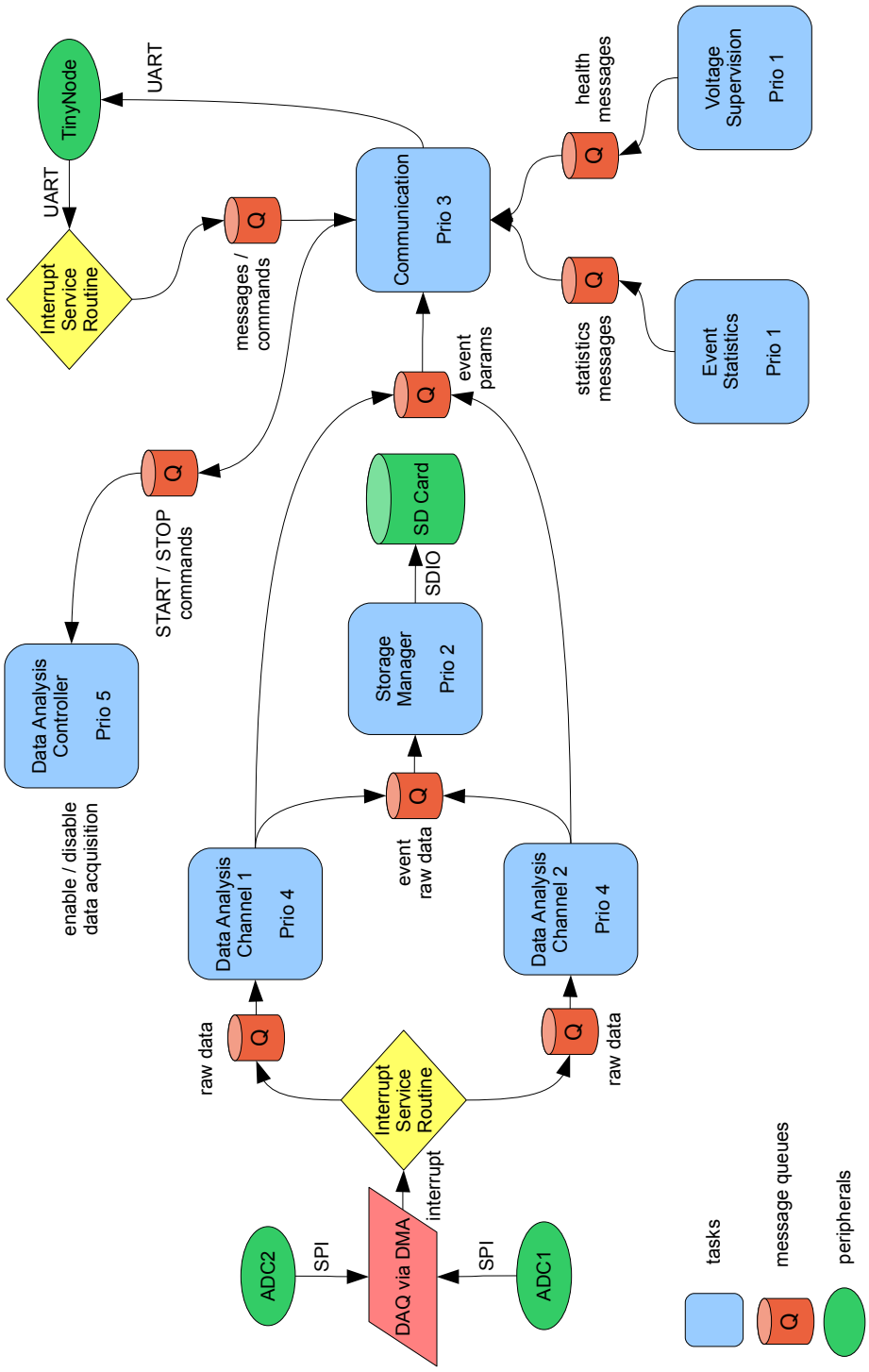


Figure 6.2: PermAE application structure on the STM32

Size [kB]	Description	# of Buffers
5	Task Stacks	n/a
36	Raw Data Input Buffers	18
9	Parameter Buffers	400
4	Storage Buffers	2
7	Queues	n/a

Table 6.6: PermAE memory partition. Total memory size is 64 kB.

6.4.1 PermAE Initialization

Before data acquisition and analysis may start, the PermAE application must be initialized. An initialization task (not included in Figure 6.2) takes care of allocating buffer memory and initializing all required data structures.

In order to ensure correct initialization after power cycling the STM32, PermAE stores some information in an initialization block on the SD card. This block is initially generated by the PermAEInit application (c.f. Section 6.4.8). Table 6.7 specifies its format and contents.

The PermAE initialization task starts by reading this initialization block and checking the magic bytes as well as the device ID in order to ensure that this SD card has been initialized with the current AEBoard. It then uses the rest of the initialization block's information to prepare data acquisition, analysis and storage. Finally, after all other tasks and data structures have been created successfully, the initialization task suspends itself forever.

For debugging purposes, the initialization task features verbose status messages on the AEBoard's debug console connector, which can be checked by connecting the AEBoard to a terminal emulation software such as, e.g., minicom (c.f. Appendix A).

6.4.2 Data Acquisition

The AEBoard connects the AD7980 ADCs to two of the STM32's SPI interfaces. The ADCs' operating mode requires their controller to initiate sampling by pulling their CNV input to "high", waiting for a fixed amount of time and then releasing CNV again. The sample may then be read by generating 16 clock pulses on the SCK input and reading the output line SDO at each raising clock edge [49]. In PermAE, both AE channels are sampled synchronously. The conversion clock, as well as the SPI clock, are generated by two timer peripherals on the STM32. Consequently, the SPI peripherals are configured to operate in slave mode. Figure 6.3 illustrates this wiring.

Offset	Size [Bytes]	Name	Description
0x00	4	magic bytes	The static string “AEP”
0x04	12	device ID	The STM32’s unique device identifier
0x10	4	serial number	The current AE event counter value
0x14	4	blocks stored	The number of raw events stored on the SD card
0x18	2	stored event size	The size of one raw event stored on the SD card
0x1A	2	stored event samples	The number of raw samples per stored event
0x1C	2	channel 1 offset	Offset calibration value for channel 1 (c.f. Section 6.4.8)
0x1E	2	channel 2 offset	Offset calibration value for channel 2 (c.f. Section 6.4.8)
0x20	2	threshold	The event detection threshold
0x22	2	posttrigger	The event detection posttrigger value
0x24	476	reserved	Reserved for future use

Table 6.7: The PermAE initialization block

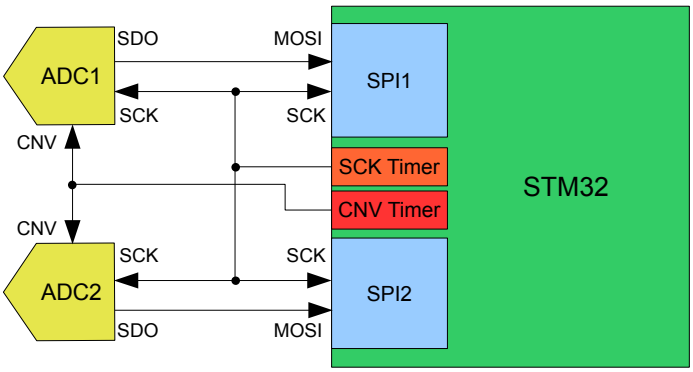


Figure 6.3: AEBoard data acquisition wiring

When data acquisition is enabled by the data analysis controller task, the

CNV timer generates a periodic acquisition pulse with a frequency of 500 kHz. The SCK timer basically generates a continuous clock signal, but it's output signal is masked by a windowing timer such that per sampling period, only 16 well-timed pulses appear at the SCK output.

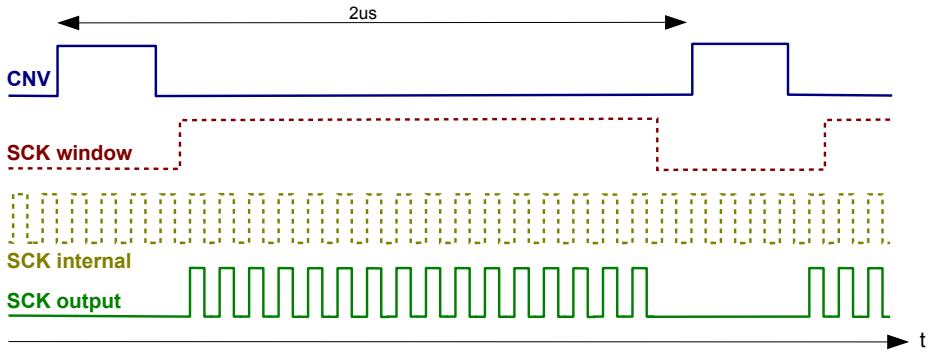


Figure 6.4: Generated data acquisition clocks

The received samples are transferred to a raw data buffer via DMA, only generating an interrupt when the raw sample buffers are full (i.e., after reading 1024 samples) and thereby minimizing unnecessary interrupt overhead. An interrupt service routine re-initializes the DMA controller with new input buffers and passes the full buffers to the data analysis tasks (c.f. Figure 6.2). Unfortunately, this process takes more than two microseconds and consequently, one sample per channel is lost at each buffer switch. To minimize the introduced error, the missing sample is estimated by calculating the mean value of its preceding and its subsequent sample.

6.4.3 Data Analysis Controller Task

Even though the data analysis controller task is the highest priority task in PermAE, it is generally the least active. Its sole purpose is to enable and disable the data acquisition process and to thereby indirectly control the data analysis tasks as outlined in Algorithm 1.

6.4.4 Data Analysis Task

Each AE channel's raw data is analyzed in a separate task. Once a sample's absolute value exceeds the set threshold, an event is detected. Event parametrization runs until the threshold has not been exceeded for the number of samples that is defined by the posttrigger value. Assuming that the first threshold crossing occurred at sample i and the event ends at sample j (i.e., the last threshold crossing occurred at sample $j - posttrigger$), Al-

Algorithm 1 PermAE data analysis control task

```
1: initialize data analysis;
2: while true do
3:     wait for DAQ_START;
4:     initialize data acquisition;
5:     start data acquisition;
6:     wait for DAQ_END;
7:     stop data acquisition;
8: end while
```

Algorithm 2 PermAE event parametrization

```
1: length := j − i;
2: amplitude := 0;
3: risetime := 0;
4: energy := 0;
5: count := 0;
6: for k = [i ... j] do
7:     if abs( samples[k] ) > amplitude then
8:         amplitude := abs( samples[k] );
9:         risetime := k − i;
10:    end if
11:    energy = energy + (samples[k])^2 / 512;
12:    if samples[k − 1] < threshold and samples[k] >= threshold then
13:        count := count +1;
14:    end if
15: end for
```

gorithm 2 is equivalent to PermAE's event parametrization algorithm (c.f. Section 2.2.2 and Figure 2.5):

There are two points noteworthy about this implementation:

- In line 11, the squared samples are first divided by a scaling factor of 512 before being added to the energy parameter. This scaling prevents energy parameter values from overflowing the assigned range of 32 bits.

- Some AE parametrization algorithms use extended criteria to define a threshold crossing in order to suppress high frequency oscillations. PermAE however counts every single exceeding of the positive threshold as depicted in Figure 2.5.

AE Event Metadata

In addition to the actual event parameters, metadata is added to the event parameters: A global event counter is incremented every time an event was parametrized. This counter value is attached to the event data as the event's serial number. The serial number can only be reset by re-initializing the SD card. Thus, it uniquely identifies every event of a sensor's deployment. The event's serial number is also crucial for correlating raw event data from the SD card with parameters stored in the PermaSense database. Furthermore, in order to temporally correlate the AE events precisely (c.f. Section 3.4), a sample counter value indicating the start sample's counter value is added to each parameter set. E.g., if event A's sample counter value is equal to 468194 and event B's sample counter value is equal to 468276, for event A, the first threshold exceedance occurred 82 samples (=164 μs) before event B's first threshold exceedance.

As the sample counter value is only 32 bits wide, it overflows and restarts from zero every 2^{32} samples, which is equivalent to 143 minutes at a sampling rate of 500 kHz. It is furthermore reset when data acquisition is stopped and restarted.

Details about converting the parameter values into physical units are given in Appendix D. Once parameter detection has finished, the detected parameters are passed to the communication task, whereas the event's raw samples and some metadata are passed to the storage manager task (c.f. Figure 6.2).

6.4.5 Communication Task

The communication task basically implements the communication protocol specified in Section 6.3. As soon as PermAE on the STM32 is fully initialized, the ARM_COMM_FLAG line is raised to signal communication readiness. Data acquisition is only enabled or disabled upon the accordant TinyNode commands.

If there are multiple data messages to forward, they are transmitted in strictly prioritized order, depending on their type as shown in Algorithm 3.

6.4.6 Storage Manager Task

The storage manager task controls and manages access to the SD card storage. The SD card is accessed via the STM32's SDIO peripheral, allowing efficient data input / output using bulk DMA transfers.

Raw event data buffers are received from the data analysis tasks and written

Algorithm 3 PermAE data message transmission

```
1: if AE parameters waiting then
2:     transmit AE parameters;
3: else if AE statistics messages waiting then
4:     transmit statistics message;
5: else if Health data message waiting then
6:     transmit health message;
7: end if
```

to the SD card unaltered. For efficiency reasons, the raw data samples of all events are stored in blocks of fixed size, which is defined at device calibration time (c.f. Section 6.4.8). E.g., if this event buffer size is set to 2048 Bytes, one raw event block on the SD card would have the following format:

Offset	Size [Bytes]	Name	Description
0x00	4	serial number	The event's serial number (c.f Section 6.4.4)
0x04	4	start sample	The event's start sample (c.f Section 6.4.4)
0x08	2	length	The event's length parameter
0x0A	2	threshold	The detection threshold used
0x0C	2	posttrigger	The detection posttrigger value used
0x0E	2032	event data	The raw event samples

Table 6.8: Raw event data block on SD card

Thus, maximally 1016 samples per event will be stored. Longer AE events are simply cropped to 1016 samples, shorter AE events do not utilize the whole storage block. Nevertheless, fixing a stored event's block size allows for very efficient write operations, as SD cards are always written in blocks of 512 Bytes. This, however, also limits the possible storage block sizes to multiples this basic block size.

As no PC readable file system is used to store the raw AE events, the data stored on a PermAE SD card must first be preprocessed in order to be used in third party software. For this purpose, a PHP script is provided on the enclosed CD that generates a simple-formatted ASCII file for each event stored on the memory card. These files may then further be processed with third party software tools, e.g., with MATLAB. For details, refer to Section A.4.2.

The storage manager task also periodically updates the PermAE initializa-

tion block stored on the SD card at address 0 (c.f. Sections 6.4.1 and 6.4.8), ensuring initialization block actuality and validity even after power cycles.

6.4.7 Event Statistics and Voltage Supervision Tasks

This rather simple tasks generate information blocks in two minute intervals. The event statistics task reads and resets the event counter values (measured events, parametrized events, stored events) and generates a statistics message thereof. Furthermore, the voltage supervision task samples the following system voltages every two minutes (c.f. Figure 5.5):

- Bias voltages for channel 1 and 2 (4.5 V)
- ADC supply voltage (2.5 V)
- ADC reference voltage (4.5 V)
- Operational amplifier supply voltage (4.5 V)

These voltages, together with the number of free event blocks on the SD card, are forwarded to the communication task as a data acquisition health message.

6.4.8 PermAE Initialization and Calibration

In order to use the STM32 PermAE implementation, some initial system parameters must be set (c.f. Section 6.4.1):

- The initial threshold value
- The initial posttrigger value
- The block size of one raw event data block (c.f. Section 6.4.6)

Furthermore, the AE measurement channels need to be calibrated: For each channel, the input signal theoretically features a static offset of exactly 2.25 V, which would correspond to an ADC output value of 32768 (c.f. Figure 5.3 and Appendix D). However, due to component inaccuracies, this offset voltage varies for every AEBoard input channel, making a preliminary calibration necessary.

For calibration and initialization, a separate application called "PermAEInit" is provided. PermAEInit measures each AE channel's offset value by sampling 1024 samples and calculating the mean value. Therefore, it is important

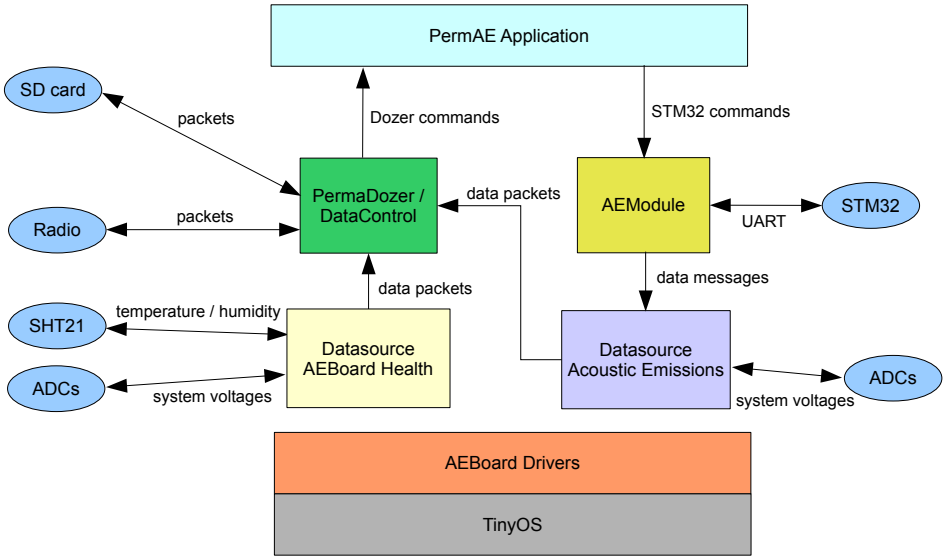


Figure 6.5: PermAE application structure on the TinyNode

to remove the AE sensors before calibrating to make sure that no unwanted sensor stimulation occurs. After measuring the channel offsets, PermAEInit generates the initialization block and writes it onto the SD card (c.f. Section 6.4.1).

The mentioned detection and storage parameters are compiled statically into the application's binary, requiring recompilation of PermAEInit to change them (c.f. Appendix B). For further information about initializing an AEBoard, please refer to Appendix A.

6.5 TinyNode Implementation

In contrast to the implementation on the STM32, the TinyNode part of PermAE is built upon an already existing software stack consisting of the TinyOS operating system [25] and the PermaDozer network stack [4, 2]. Building on this existing infrastructure, the PermAE TinyOS application only has to implement the communication protocol specified in Section 6.3 and provide data source modules as well as a hardware description layer for the AEBoard. Figure 6.5 depicts the application's basic architecture.

All in all, much of the necessary code could be reused from other PermaSense applications, greatly facilitating a seamless integration into the existing PermaSense infrastructure.

6.5.1 AEBoard Drivers

The AEBoard driver layer abstracts the AEBoard hardware resources: SD card and SPI bus, the UART interface to the STM32, the MSP430 ADCs for measuring system voltages and currents as well as the SHT21 temperature and humidity sensor [58]. As most of the components attached to the TinyNode have already been used on other PermaSense platforms (i.e., the sensor interface board (SIB) and the PermaDozer base board), most of the hardware layer code could be reused from the already existing codebase in a slightly adapted form. Comparably lightweight datasource plugins for the PermaDozer DataControl module implement the actual measurement routines.

6.5.2 AEModule

The AEModule basically implements the onboard communication protocol (c.f. Section 6.3). On one hand, it provides a control interface to the PermAE top level application to send command messages to the STM32 (start / stop data acquisition, setting the threshold and posttrigger values). On the other hand, it reads data messages from the STM32 and injects them into the PermaDozer message queue.

6.5.3 PermaDozer and DataControl

The PermaDozer and DataControl libraries already available from other PermaSense applications manage WSN communication and SD storage, therefor utilizing the AEBoard driver hardware abstractions. The existing SD storage engine was slightly adapted for PermAE: In other PermaSense applications, data was written to the SD card only every two minutes. As in these applications, the data rate is well known, queue overflows could easily be avoided. In PermAE however, data packets arrive irregularly and in bursts, sooner or later overflowing the storage module's fixed-size input queue. Thus, the PermAE storage queue adds incoming data packets to the SD card storage buffer immediately, instead of rejecting new packets when the input queue is full. Furthermore, the PermaDozer component generates a few other standard messages (eventlogger, rssi, statecounter) for network management purposes.

6.5.4 PermAE Top Level Application

The PermAE module is the top level TinyOS application. After initializing the hardware, it turns on the STM32 and waits for the ARM_COMM_-

FLAG pin to be raised (c.f. Section 6.3). Then, it sends a START_DAQ command to initiate sampling of AE data. Once up and running, the application behaves as follows:

- When the ARM_COMM_FLAG pin is raised again, a message read operation by the AEModule is initiated.
- Upon reception of a WSN command (c.f. Section 6.2.2), the command is parsed and an according command is sent to the STM32 through the AEModule.
- Every two minutes, PermaDozer initiates periodic measurements. PermAE then starts the system health measurement routines to generate a "nodehealth" as well as an "aedaqhealthdata" packet. The latter packet bases on the last received health message from the STM32 and is extended with some values measured by the TinyNode.

For further information about the generated packets, refer to Appendices C and D.

6.6 GSN Integration

As the PermAE implementation on the TinyNode bases on the same mechanisms as all other PermaSense WSN nodes, AE data integration into GSN is a fairly simple process. The TinyOS `mig` tool converts the message format definitions in the PermAE header files to Java classes, which in turn can be integrated into the PermaSense GSN instances. As the `mig` Makefile has been adapted accordingly, integrating AE messages into another GSN instance is just a matter of running the `mig` tool and copying the generated class files to the respective GSN instance path. For closer details about this process, refer to the PermaSense documentation.

As usual in PermaSense, data calibration/conversion to physical units is done in between the "Private" and the "Public" GSN instances. I.e., when accessing AE data on the "Public" GSN instance, the data displayed has already been converted to physical units (c.f. Appendices C and D).

7 System Evaluation

To complete the presentation of the AEBoard and PermAE, the system's performance characteristics are evaluated in this chapter. A special focus thereby lies on the reliable system operation under harsh environmental conditions.

7.1 Measurements at Room Temperature

7.1.1 Analog Frontend Characterization

In order to ensure a good quality of the measured samples, the analog frontend has been characterized in terms of frequency response and linearity. For both measurements, the input signal has once been applied directly at the AEBoard signal input, and once at the input of the IL-LP-6S external preamplifier by Physical Acoustics.

Input Frequency Response

Figure 7.1 shows the AEBoard's analog frontend frequency response. The input signal has been applied at the AEBoard input test pins and no preamplifier has been attached at the input channel. The output was measured directly at the ADC input.

The system's input is attenuated by 6 dB, due to the bias voltage providing

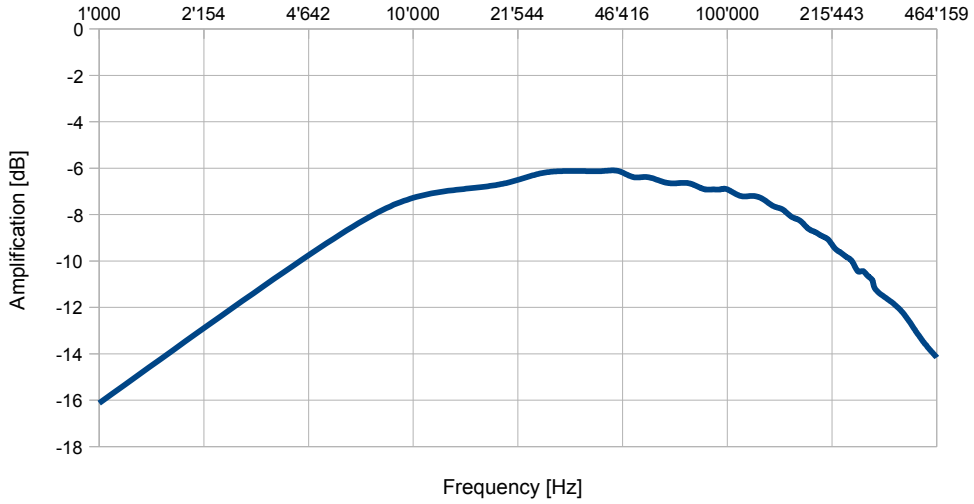


Figure 7.1: Frequency response of the onboard signal conditioning circuitry.

power to the external preamplifier (c.f. Section 5.3). If more input gain is required, the operational amplifier feedback path's resistor value could be increased (c.f. Figure 5.3 and Appendix E).

The frequency response has also been measured with the external preamplifier included in the signal path. The input signal has been applied at the preamplifier input, the output was measured again directly at the ADC input. Figure 7.2 depicts this measurements' results.

For the whole signal conditioning chain, the point of maximal amplification lies at a frequency of 35 kHz with an amplification of 22 dB. In the frequency range of 20 to 100 kHz, the loss compared to the maximal amplification does never exceed 1 dB. Thus, the input filter matches the system specifications in Section 3.4, requiring an input frequency bandwidth of 20 to 100 kHz.

Filter Linearity

Besides the frequency response, the input filter's *linearity* is another factor that heavily affects the sampled signal's quality. A linear filter only changes amplitude and phase of an input signal's frequency parts, without introducing frequencies that have not been part of the original signal. Thus, if a linear filter is fed with a perfectly sinusoidal signal, its output is a phase-shifted, amplified copy of the input signal.

The signal conditioning circuitry's linearity has been analyzed at input frequencies of 30, 50 and 100 kHz. The sinusoidal input signal was attached at the AEBoard's input connector and the filter output has been measured at

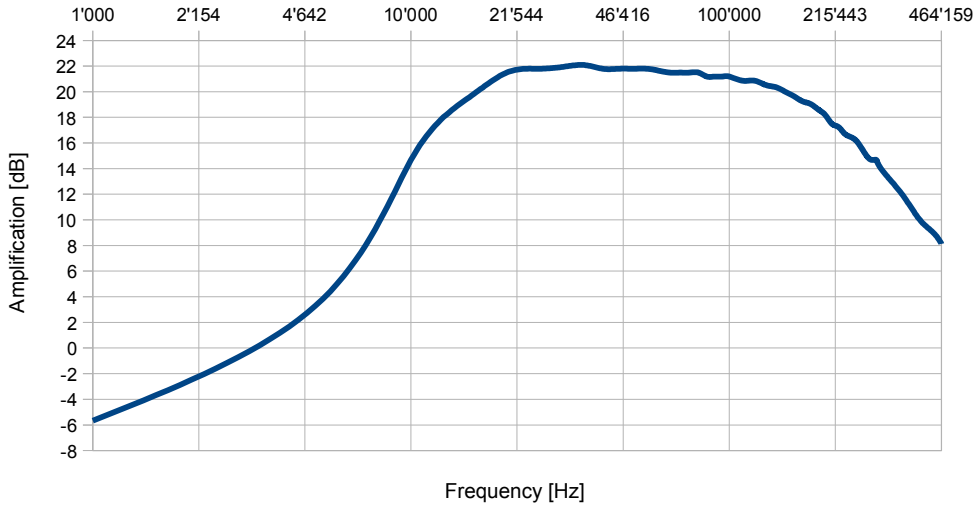


Figure 7.2: Frequency response of the signal conditioning circuitry including the external preamplifier.

the ADC input. As the AEBoard input filter is inverting the input signal, the sum of both signals is equal to the signal change introduced by filtering. Figure 7.3 shows the input and output signal for the 50 kHz measurement, as well as their sum.

Obviously, the error signal is also sinusoidal, indicating a linear filter character. Indeed, after shifting and amplifying the output signal adequately (i.e., compensating the linear filtering), the difference between the signals is smaller than 5% as shown in Figure 7.4.

However, when adding the external preamplifier to the signal conditioning chain, the linearity is lost. Figure 7.5 clearly shows, that even after compensating phase shift and amplification, significant nonlinear distortions of the output signal remain. Because the IL-LP-6S preamplifier also inverts its input signal, the output signal must now be subtracted from the input in order to find the error introduced.

Nevertheless, although this nonlinear behaviour certainly affects the raw event data that is stored on the SD card, it should not notably affect the AE event parametrization results. However, further analysis based on the stored raw samples that includes frequency domain calculations should take these nonlinear effects into account.

For all linearity measurement results, please refer to Section F.1.

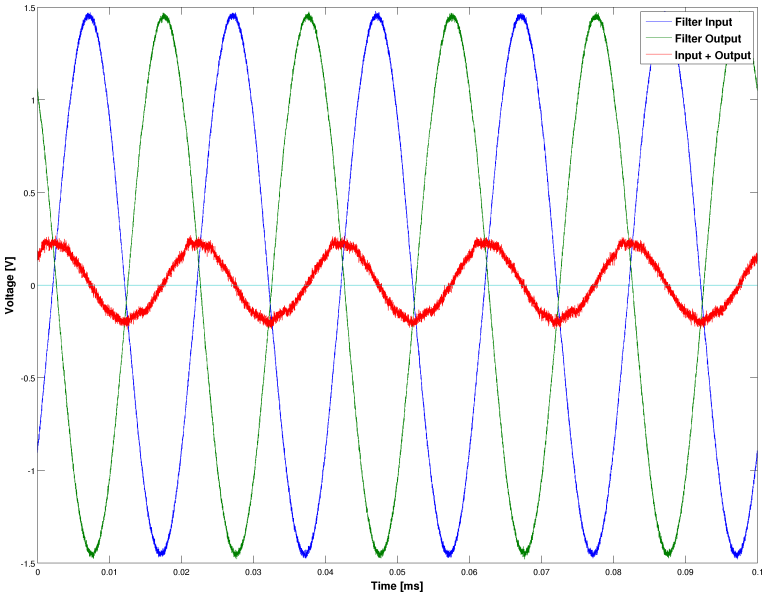


Figure 7.3: Uncompensated error introduced by the AEBoard input filter.

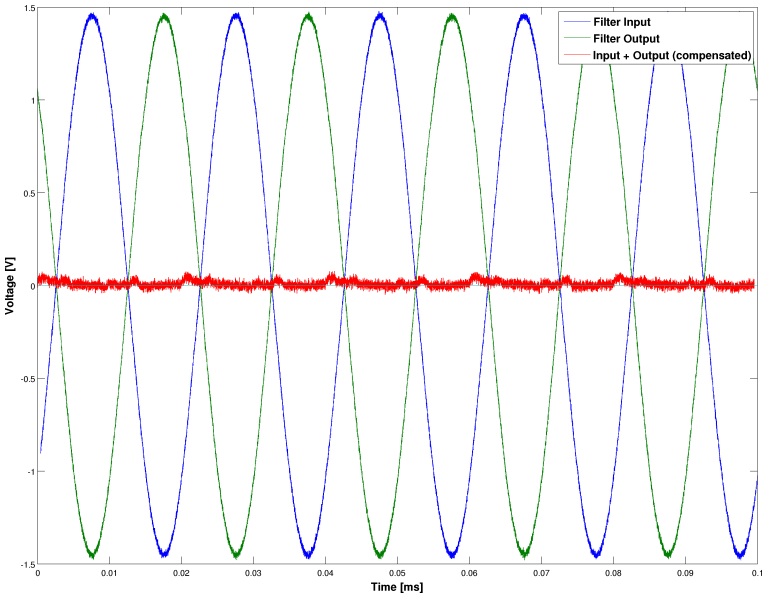


Figure 7.4: Phase and amplitude compensated error, introduced by the AEBoard input filter.

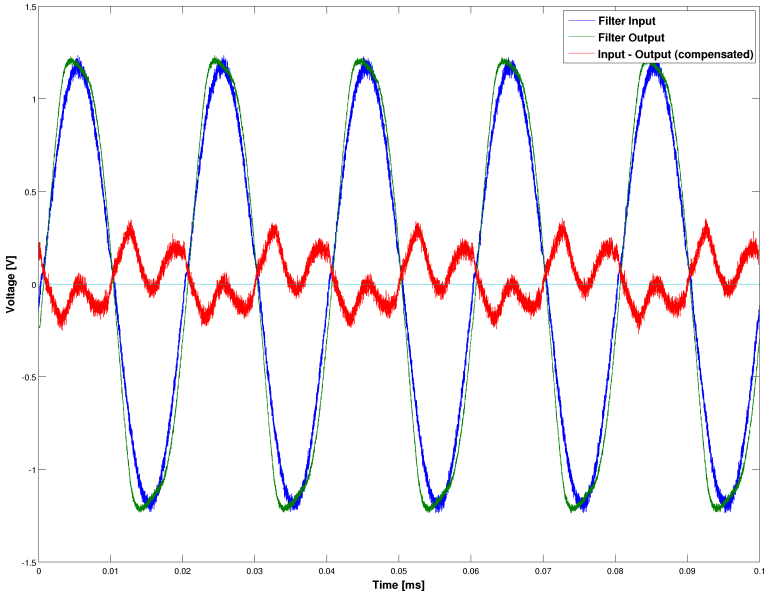


Figure 7.5: Phase and amplitude compensated error, introduced by the external preamplifier and AEBoard input filter.

7.1.2 Power Consumption

The AEBoard's power consumption has been measured at room temperature for various isolated system parts as well as for the whole system in operation. Table 7.1 summarizes these measurements.

Power Domain	**Input Voltage**	**Input Current**	**Input Power**
Digital Master Domain	3.3 V	23 mA	75 mW
Digital Slave Domain	2.8 V	44 mA	122 mW
Sensing / Conversion Domain	5.5 V	13.7 mA	75 mW
System in operation	12 V	48 mA	576 mW

Table 7.1: AEBoard power consumption (for domain names, refer to Figure 5.5)

The difference between the total system power consumption and the accumulated subsystem consumption is due to losses in the power supply system. These losses clearly dominate the overall power consumption with a total of 304 mW, probably leaving much potential for further optimization.

From the beginning, the AEBoard was designed to support a photovoltaic power supply. Nevertheless, the Li-$SOCl_2$ batteries currently used in PermaSense (the "LSH 20" model by Saft batteries [59]) feature a nominal capacity of

13Ah per cell. When combining three of these 3.6 V batteries to a 10.8 V power supply, an AEBoard could continuously acquire data for about 10 days. This may be sufficient for shorter AE sensing campaigns. For long-term measurements however, a photovoltaic power supply is inevitable.

7.2 Temperature Cycle Tests

In order to evaluate the system performance under realistic environmental conditions, the following performance tests have been carried out under varying temperature conditions. To this end, an automatic climate chamber has been utilized to ensure controlled experimental conditions. The following questions have been studied specifically:

- Is the system operating stable under quickly varying temperature conditions?
- Does temperature affect the system's performance in terms of maximal AE activity load that can be processed and stored?
- Does temperature affect the system's power consumption?
- How does the signal conditioning circuitry behave under temperature cycles and how is the parametrization result affected?

To answer these questions, two experiments have been carried out in the temperature chamber, as discussed in the following sections. As the experimental conditions shall be reproducible, no piezoelectric AE sensor has been used for these experiments. Instead, the "STM32Discovery" evaluation kit for the STM32 platform has been utilized to simulate the output of a piezoelectric AE sensor. Thus, the following experiments are strictly electrical and do not consider any mechanical or electrical characteristics of the AE sensor itself. The application used for event simulation is also included on the enclosed CD or in the PermaSense SVN repository (c.f. Appendix B and G).

7.2.1 Load Test

The first temperature test setup was designed to evaluate the overall system stability under changing temperature conditions as well as measuring the system's event processing capabilities over an extended period of time and for different temperatures. The AEBoard, as well as the external preamplifiers, have been placed in the climate chamber, whereas the power supply and the

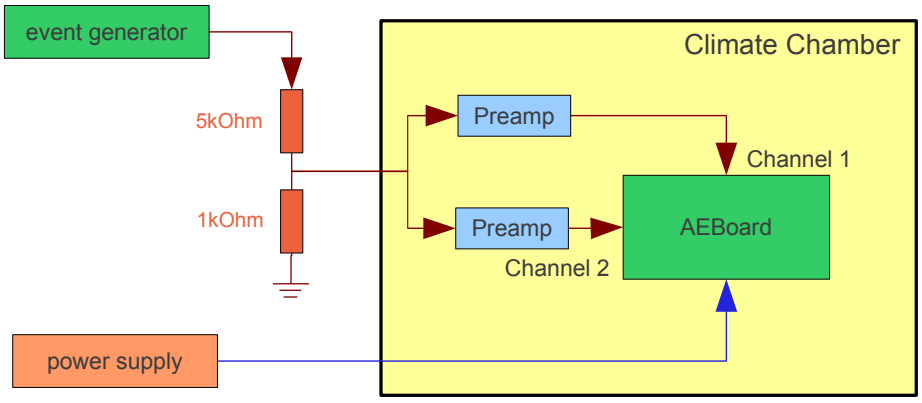


Figure 7.6: Load test setup.

event generator were kept at room temperature in order to ensure stable conditions. Figure 7.6 sketches this setup.

Every five minutes, the event generator produced fifteen AE events per second on both acquisition channels over a period of 30 seconds, resulting in 900 generated AE events per stimulation period. This event rate is high enough to be above all expected event bursts occurring in a real measurement deployment (c.f. Section 3.1.2). All generated AE events were identical, featuring an amplitude of 80 dB (1 V) at the ADC input and an event length of about 2700 samples (1.35 ms). For each stimulation period, the according "aestatistics" messages have been analyzed in order to evaluate the system's parametrization and storage performance. During the experiment duration of totally seven hours, the ambient temperature was varying between -35°C and +50°C. Figure 7.7 shows the results of this load test experiment.

Obviously, the varying temperature conditions did not affect the system's performance in terms of operational stability as well as parametrization and storage capability. Furthermore, according to the preliminary experimental results described in Section 3.1, the applied load is unrealistically high. Additional tests showed that by reducing the stimulation period to only 15 seconds, all occurring events during a stimulation period could be parametrized, and over 90% of the events could be stored on the SD card. The system's performance bottleneck in terms of parametrization capability is the serial port's communication speed to the TinyNode - as more events arrive than can be forwarded, the acquisition system slowly runs out of input buffers. SD storage performance however is not limited by the speed of the SDIO interface. Rather, as the storage task features a lower priority than the communication task, it simply does not have enough processor time to store all events. Raising the storage task's priority would solve this problem but also

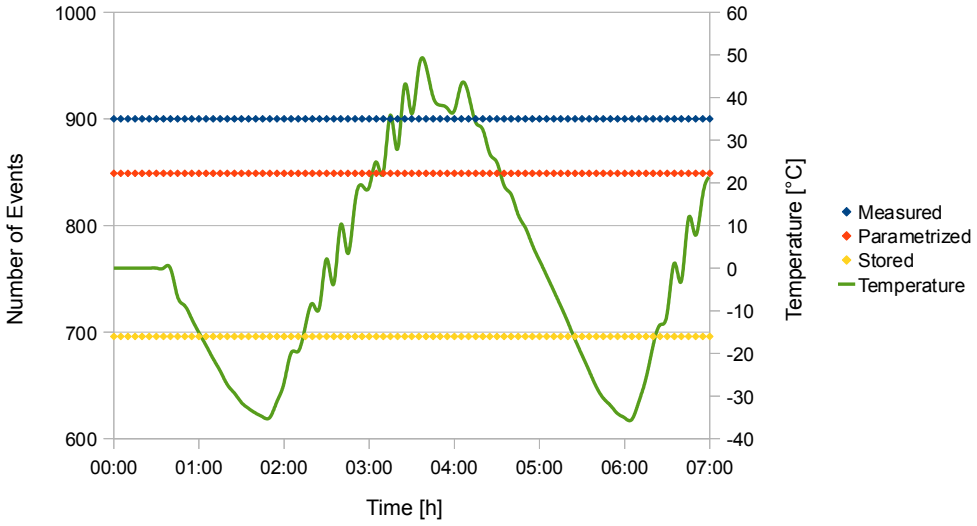


Figure 7.7: System performance under varying temperature conditions.

violate the system specification (c.f. Sections 3.4 and 6.4).

7.2.2 Data Quality Test: AEBoard and USB AE Node

Even tough temperature seems not to influence the system's overall reliability and speed, it certainly affects the data acquisition system's analog components and thus also the parametrization results. In order to investigate on these effects, data acquisition quality has been tested and compared to a commercially available system.

Again, the AEBoard has been placed in the climate chamber, together with the external preamplifier for channel one. The preamplifier for channel two however was placed outside the chamber to allow for analysis of the preamplifier's impact on data quality. Furthermore, an USB AE Node by Physical Acoustics Corporation [60] together with a preamplifier was also placed in the climate chamber in order to get reference measurements of a state of the art AE measurement device. This USB device was connected to a PC running the AEWin software [39]. Figure 7.8 depicts the data quality test setup.

All AE preamplifiers were fed with the same input signal generated by the aforementioned event generator based on the STM32Discovery evaluation kit and the "EventGenerator" software. The test signal, as shown in Figure 7.9, simulates an AE event consisting of three frequency components at 35, 50 and 60 kHz, with the 50 kHz oscillation being the dominant frequency component. This characteristics match well the "average" waveform that

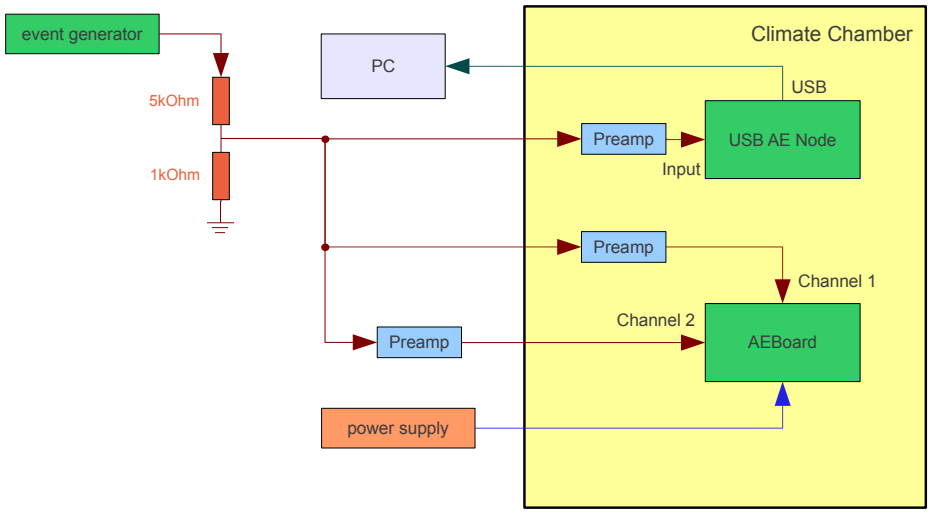


Figure 7.8: Data quality test setup.

has been captured during the preliminary experiments. For detection, a threshold value of 52 dB was used in conjunction with a posttrigger time of 400 samples (800 μs).

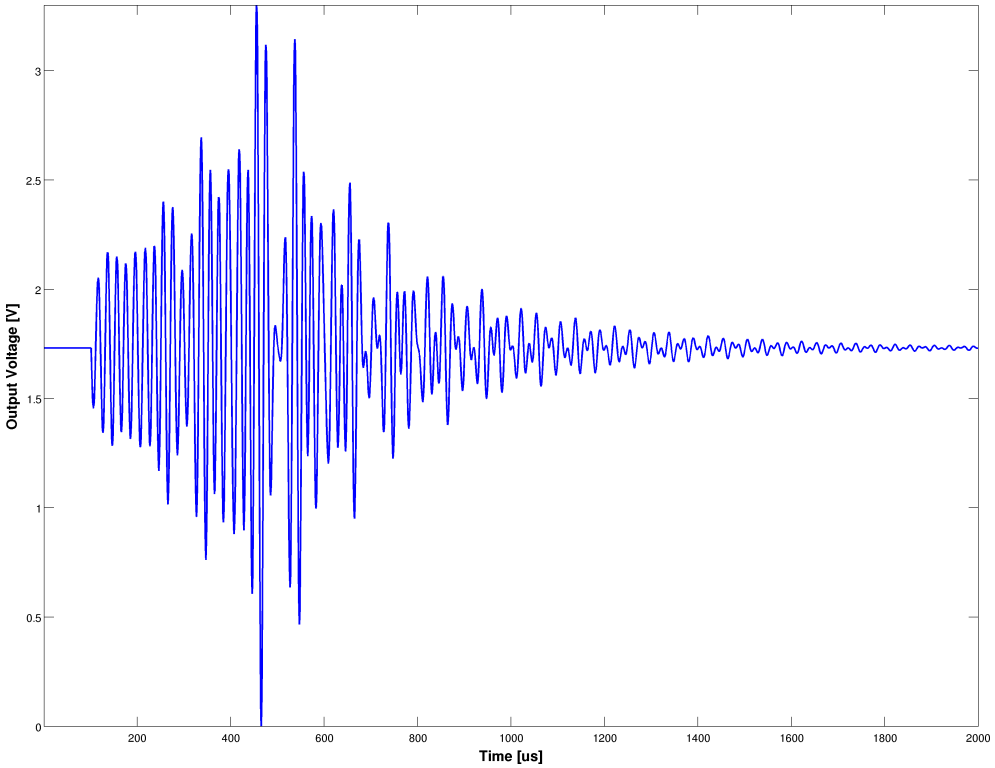


Figure 7.9: Test signal generated by the EventGenerator application.

The test signal was generated exactly once every 60 s over the total experiment duration of 18 hours. Ambient temperature varied in the range of -30°C up to +40°C during that timespan. The measured parameters for all captured test signals have then been compared. In an ideal system, these measured parameters would have remained constant over the whole experiment.

Generally, this data quality test showed that some AE parameters are heavily influenced by temperature variations, while others are not. Section F.2 shows the result plots for all parameters and channels, while the most outstanding effects are discussed in the following.

Temperature Dependent Parameter Variations

Table 7.2 gives an overview of the results for AEBoard channel one, which is the most relevant measurement for the estimation of the AEBoard's data quality in a productive deployment. The mean values at room temperature are compared to the mean values at -30°C and +40°C. The "B" column indicates, whether the parameter exposes pro-cyclic[1] or anti-cyclic[2] behaviour with temperature.

Param.	**Mean +15°C**	**Mean -30°C**	**Mean +40°C**	**B**
Length	1266	1321 (+4.3%)	1229 (-2.9%)	a
Risetime	218	218 (+0.0%)	218 (+0.0%)	n/a
Amplitude	20421	19376 (-5.2%)	20532 (-0.5%)	p
Count	73	77 (+5.5%)	70 (-4.1%)	a
Energy	382e6	418e6 (+9.4%)	317e6 (-17.0%)	a

Table 7.2: Data quality test results for AEBoard channel one. "p" stands for pro-cyclic, "a" for anti-cyclic behaviour. See Section F.2 for the according plots.

The table shows, that the risetime parameter was not influenced by the temperature variations at all and in fact, it almost stayed constant over the whole experiment, thus featuring an absolute precision of less than 5μs. The other parameters however were influenced stronger by temperature changes. Especially the energy parameter exposes a bandwidth of almost 30% of its average room temperature value.

Figure 7.10 depicts the pro-cyclic parameter behaviour of the amplitude parameter versus anti-cyclic behaviour at the example of the energy parameter.

[1] Pro-cyclic: The parameter value rises when temperature rises.
[2] Anti-cyclic: The parameter value falls when temperature rises.

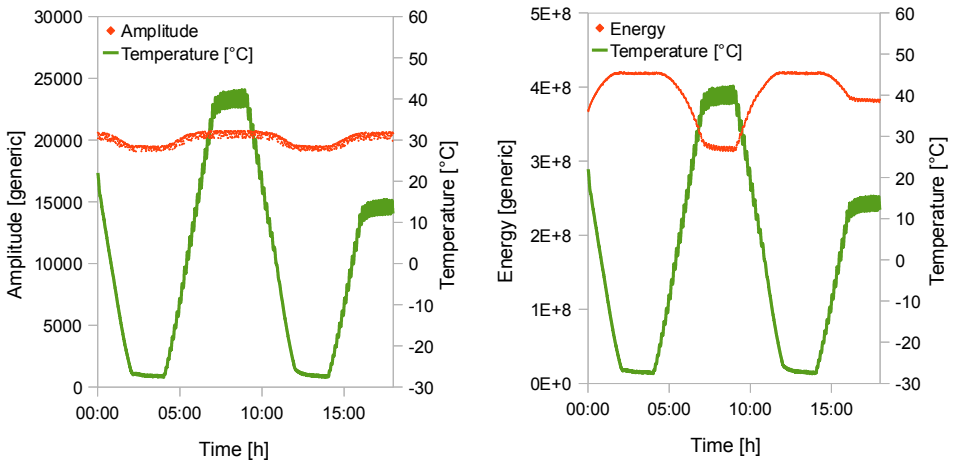


Figure 7.10: Pro-cyclic versus anti-cyclic parameter behaviour.

Another data quality indicator is the parameter variation for constant temperature. As all stimulation events are identical, in an ideal system this variations would be zero for phases with reasonably constant ambient temperature. Table 7.3 lists the constant temperature standard deviations for all AE parameters measured at channel one in absolute units as well as relative to the mean value at the respective temperature.

Param.	**Std Dev +15°C**	**Std Dev -30°C**	**Std Dev +40°C**
Length	14.4 (1.1%)	24.8 (2.0%)	10.6 (0.8%)
Risetime	0.5 (0.2%)	0.4 (0.2%)	0.5 (0.2%)
Amplitude	175.4 (0.9%)	125.2 (0.6%)	183.6 (0.9%)
Count	1.1 (1.5%)	1.3 (1.8%)	0.8 (0.9%)
Energy	106e4 (0.3%)	65e4 (0.2%)	212e4 (0.6%)

Table 7.3: Parameter standard deviations at constant temperatures for AEBoard channel one. See Section F.2 for the according plots.

Obviously, although temperature variations influence the measurements heavily, parameter variations for constant ambient temperatures expose a standard deviation of less than 2% for all parameters.

Thus, for reasonably constant ambient conditions, the parametrization result can reliably be reproduced. For changing temperature conditions however, the parametrization results should be adapted appropriately. To gain reliable understanding of the temperature dependent parameter variations however, further tests under controlled conditions are necessary.

Influence of the External Preamplifier

As already described at the beginning of Section 7.2.2, the external preamplifier for AEBoard channel two was placed outside the temperature chamber in order to analyze its effect on the temperature dependent parameter variations. Like in the previous section, Tables 7.4 and 7.5 list the temperature dependent variations as well as the parameter standard deviation at constant temperatures.

Param.	**Mean +15°C**	**Mean -30°C**	**Mean +40°C**	**B**
Length	1260	1276 (+1.3%)	1238 (-1.7%)	a
Risetime	218	218 (+0.0%)	218 (+0.0%)	n/a
Amplitude	20853	20931 (+0.4%)	20914 (+0.3%)	n/a
Count	72	73 (+1.4%)	71 (-1.4%)	a
Energy	396e6	389e6 (-1.5%)	373e6 (-5.6%)	a

Table 7.4: Data quality test results for AEBoard channel two. "p" stands for procyclic, "a" for anti-cyclic behaviour. See Section F.2 for the according plots.

Param.	**Std Dev +15°C**	**Std Dev -30°C**	**Std Dev +40°C**
Length	15.5 (1.2%)	17.1 (1.4%)	9.6 (0.8%)
Risetime	0.5 (0.2%)	0.5 (0.2%)	0.5 (0.2%)
Amplitude	174.4 (0.8%)	190.9 (0.9%)	185.6 (0.9%)
Count	0.9 (1.3%)	0.9 (1.3%)	0.6 (0.8%)
Energy	424e3 (0.1%)	396e3 (0.1%)	933e3 (0.2%)

Table 7.5: Parameter standard deviations at constant temperatures for AEBoard channel two. See Section F.2 for the according plots.

When comparing these values with the values from Section 7.2.2, we find that keeping the external preamplifier's ambient temperature constant

- significantly reduces the temperature dependent parameter variations for all parameters (except the risetime parameter, which does not show any temperature dependency at all).
- slightly reduces the parameter variation for constant AEBoard ambient temperatures.

Figure 7.11 illustrates the first finding by depicting the mean ranges from Tables 7.2 and 7.4.

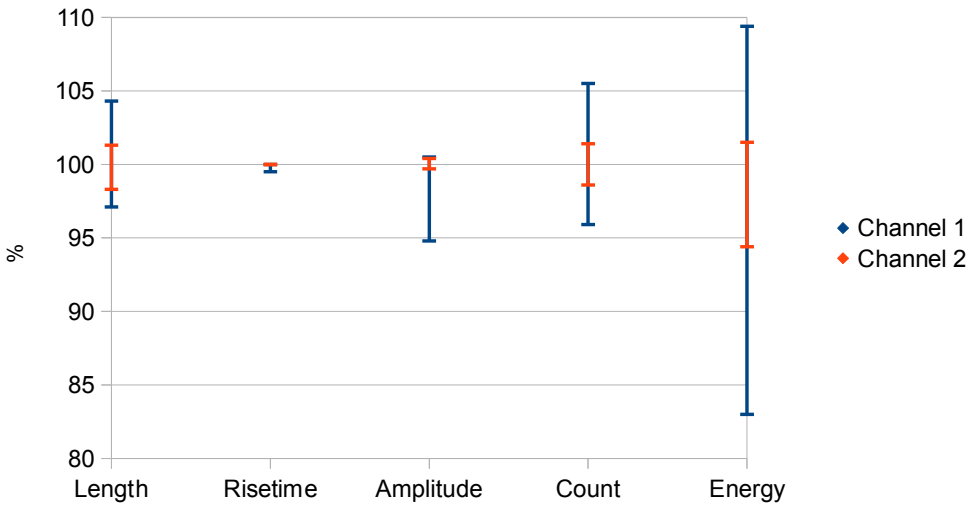


Figure 7.11: Temperature dependent mean parameter ranges for both AEBoard channels. For channel 1, the external preamplifier was placed inside the climate chamber, whereas the preamplifier for channel 2 was held at constant temperature. The mean parameter value at 15°C is set to 100%.

Figure 7.12 shows the energy parameter plots for both channels, further demonstrating the reduction in temperature dependent parameter variations.

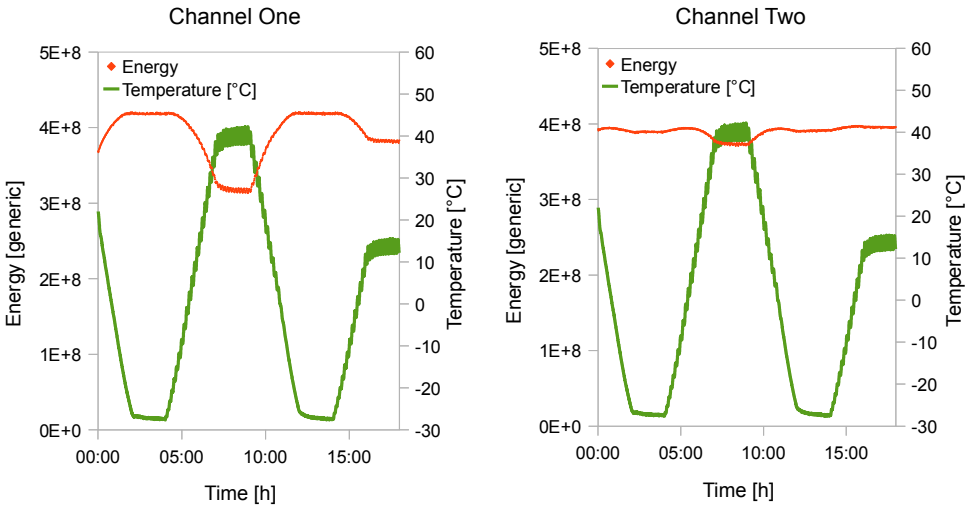


Figure 7.12: The effect of keeping the external preamplifier's ambient temperature constant at the example of the energy parameter.

This findings indicate that keeping the preamplifier's ambient conditions constant only reduces the effects on the parametrization result, but does not change the effects' characteristics. Hence, the AEBoard's signal conditioning circuitry's behaviour under temperature variations is qualitatively as well as

quantitatively well comparable to state of the art industry products such as the IL-LP-6S preamplifier.

USB AE Node Test Results

As for the USB AE Node, no temperature specification is given by the manufacturer, it was also exposed to the full temperature cycle in order to test out its limits. Basically, the results show a behaviour that is comparable to the AEBoard's results. However, the USB AE Node only seems to operate reliably in a temperature range between -10°C and +30°C. Figure 7.13 illustrates this with the USB AE Node's risetime and count plots.

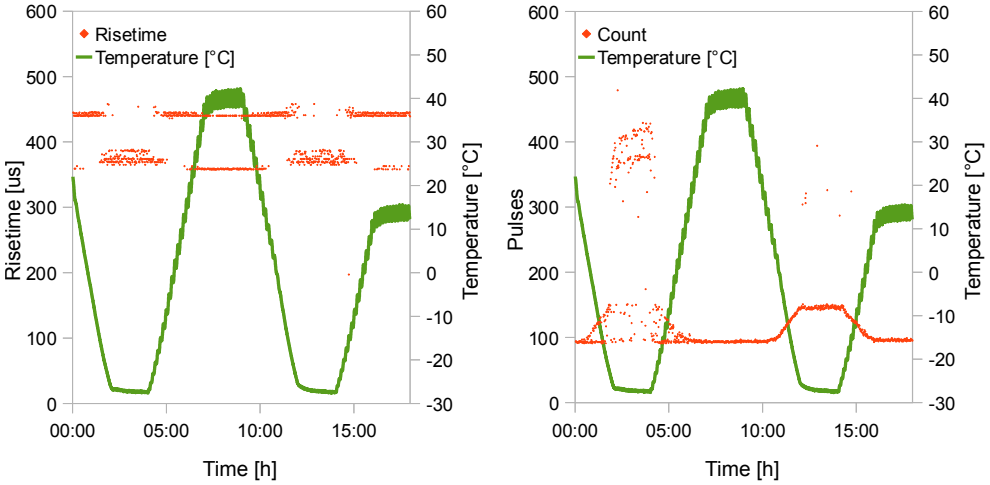


Figure 7.13: USB AE Node temperature dependent variations. At temperatures below -10°C and above +30°C, some parameter values start to expose strong variability. See also Section F.2 for other parameter plots.

Furthermore, the USB AE Node requires a PC in order to acquire data. Most of these devices are also not specified for extreme temperatures. Thus, utilizing the USB AE Node for reference AE tests is only recommended if the environmental conditions are controlled or well known in advance.

Comparing PermAE with AEWin Results

A further goal of the data quality test was to compare results of the PermAE running on the AEBoard with results from a commercial reference system, i.e., the USB AE Node with AEWin. Our findings are that, qualitatively, both devices expose comparable behaviour under the same environmental conditions (c.f. Section F.2). However, a quantitative comparison of the results is difficult, as both the AEBoard and the USB AE Node feature different internal analog signal amplification and filtering. Thus, the signal

measured at both ADCs is never the same, even tough the devices have received the same stimulating signal.

For the three parameters length, rise time and energy, the conversion constants could roughly be estimated based on the data quality test results. Of course, only results within the USB AE Node's reliable operating temperature range have been considered. Table 7.6 lists the respective factors.

Parameter	**PermAE unit**	**AEWin unit**	**c** ($\mathbf{v_{AEWin} = c \cdot v_{PermAE}}$)
Length	samples	μs	2μs
Risetime	samples	μs	2μs
Energy	V^2	V^2	80e3

Table 7.6: Conversion factors between PermAE and AEWin based on the data quality test results.

For the event amplitude and pulse count parameters, the results from PermAE and AEWin differ significantly. Probably, this difference is due to different signal amplification factors in the device's signal conditioning circuitry. The conversion functions between PermAE and AEWin results for these two parameters are most probably not linear and their identification would require further in-depth investigations.

7.2.3 Current Consumption

During the data quality test, also the AEBoard's current consumption at a supply voltage of 12 V has been constantly monitored (c.f. Section 7.1.2). As Figure 7.14 shows, the AEBoard's current consumption does not exhibit a distinct temperature dependency.

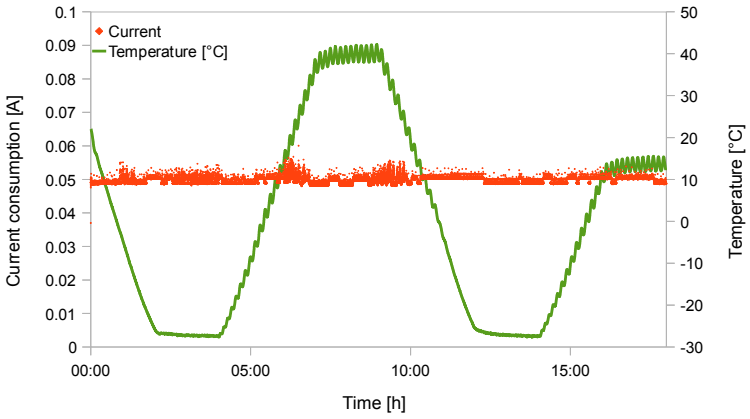


Figure 7.14: AEBoard's power consumption at varying ambient temperature.

8

Conclusion and Outlook

In this thesis, the architectural approach of processing sensors has been presented, allowing to sense data with high sampling rates in a low-power optimized WSN environment (c.f. Chapter 4). With processing sensors, a low data rate sensor is emulated by performing parts of the data processing already on the sensor device and thus reducing the data volume drastically. Although this thesis focuses on the application of AE sensing, processing sensors may also be used for other applications, e.g., on sensor processing of camera images.

Based on the concept of processing sensors, an AE sensing platform for PermaSense has been implemented (c.f. Chapters 5 and 6). The necessary data acquisition hard- and software systems have been designed from scratch, whereas the TinyNode could be used as WSN platform, thereby ensuring maximal compatibility with the existing PermaSense infrastructure. An evaluation of various system characteristics showed that the developed system indeed meets its specifications and is also competitive with commercially available AE sensor systems (c.f. Chapter 7). However, the system's stability and reliability in an outdoor deployment still needs to be proven. According experiments are scheduled for summer 2011.

System tests with heavy load showed that PermAE currently is not always able to parametrize and store all events occurring (c.f. Section 7.2.1). Nevertheless, the performance achieved is sufficient for the specific application, as the applied load was much higher than the expected activity peaks in field experiments. However, it is not clear whether buffer sizes and task pri-

orities chosen in the current PermAE implementation are optimal in order to achieve maximal parametrization and storage throughput. As PermAE's basic structure is essentially not very complex, analytical modelling could help to further optimize PermAE's performance.

As parametrization results vary significantly with changing ambient temperature (c.f. Section 7.2.2), care must be taken when analyzing the results produced in order to not interpret temperature dependent changes as changes in AE characteristics. From the perspective of possible applications in early warning systems, it is even desirable to automate the according error compensation processing and integrate it either in the parametrization software itself or at a later stage in the data backend.

Further steps based on this work may include the following:

- Deploying the developed AE sensors in the field in order to validate their robust operation over longer periods of time.
- Optimizing the AEBoard's power consumption.
- Optimizing PermAE's parametrization performance under heavy load possibly using analytical modelling techniques.
- Quantitatively analyzing the measurement errors introduced by temperature variations in order to eliminate these errors before further data analysis.
- Building other high data rate sensors based on the concept of processing sensors.

Of course, from a non-technical perspective, further steps also include application-focused field deployments in order to gain insight into crack formation processes in rock walls.

We hope that the work presented will eventually contribute to the development of reliable rockfall warning systems, potentially saving many human lives from these natural hazards. Technology must always be about people after all.

> *"Das Gefährlichste an der Technik ist, dass sie ablenkt von dem, was den Menschen wirklich ausmacht, von dem, was er wirklich braucht."* Elias Canetti

A

AEBoard and PermAE User Manual

This chapter gives a short overview of the AEBoard's functionality and explains how to prepare it for deployment. All instructions refer to hardware revision 1.1 and may need to be adapted for further revisions.

A.1 Connectors and LEDs

Figure A.1 schematically depicts the AEBoard's connectors and LEDs.

- **Power supply connector**: Connect to a power source with an output voltage between 7 and 20 V.
- **On / Off switch:** Use to switch the main power supply on or off.
- **Conversion frequency jumper:** Apply a jumper to fix voltage conversion switching frequency. Should be removed for power efficiency reasons.
- **Digital supply voltage jumper:** A jumper must be applied at the *right* position (3.3 V) to ensure correct system operation. This jumper has been removed in rev. 1.2 and digital supply voltage was fixed to 3.3 V.
- **TinyNode reset button:** Press to reset the TinyNode.

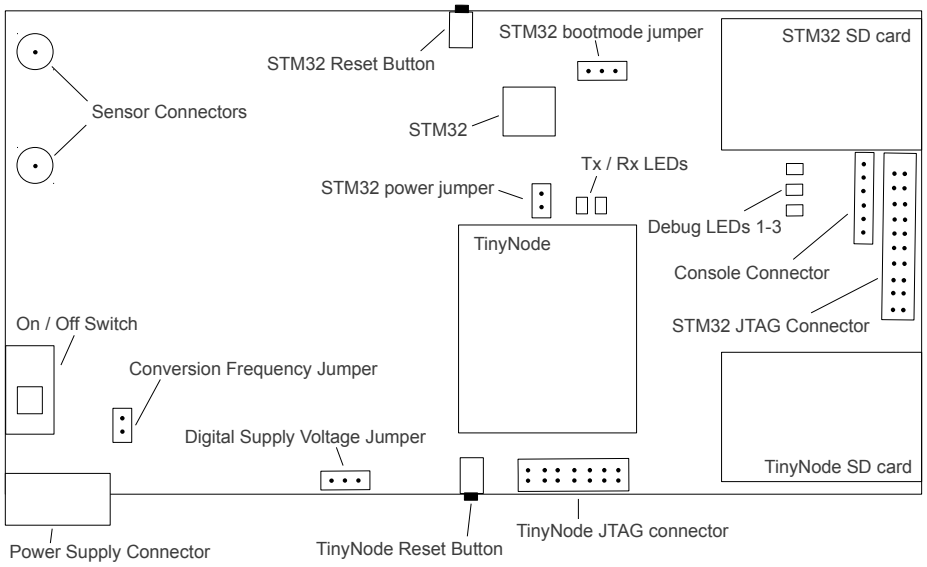


Figure A.1: AEBoard connectors and LEDs.

- **TinyNode JTAG connector:** Use to program or debug the TinyNode's MSP430 microprocessor.
- **TinyNode SD card:** Insert an SD card to enable TinyNode backlogging (c.f. Section A.2).
- **TX / RX LEDs:** UART traffic LEDs to indicate communication between the TinyNode and the STM32.
- **STM32 power jumper:** Apply a jumper in order to turn on the STM32 once the main power supply is enabled. This jumper must be removed for productive operation and is only intended for deployment preparation (c.f. Section A.2).
- **Debug LEDs 1-3:** Display PermAE's status on the STM32:
  - *LED1 (red):* Indicates that the STM32 is turned on.
  - *LED2 (orange):* Indicates a software error if turned on (c.f. Section A.3).
  - *LED3 (green):* Toggles upon detection of an AE event.
- **Console connector:** Attach e.g. a TTL-232R-3V3 USB TTL serial cable [54] to display debug messages on a PC console (c.f. Section A.3).

- **STM32 JTAG connector:** Use to program or debug the STM32 microprocessor (c.f. Section A.2 and Appendix B).
- **STM32 SD card:** Insert an SD card for the STM32 (c.f. Section A.2).
- **STM32 bootmode jumper:** Apply a jumper at the *left* position to ensure correct system operation. Applying a jumper to the right would let the STM32 boot from its factory bootloader (c.f. [47]).
- **STM32 reset button:** Press to reset the STM32.
- **Sensor connectors:** Attach the external peramplifier's output directly to these SMA connectors.

A.2 Deployment Preparation

The following guide assumes the reader to be already familiar with PermaDozer and the TinyOS toolchain. Furthermore, the following prerequisites must be met in order to prepare an AEBoard for deployment:

- A windows PC with the "STM32 ST-LINK Utility" software installed[1].
- An ST-LINK JTAG programmer.
- A working TinyOS 2.x toolchain with all PermaSense specific libraries in order to build and flash PermAE for the TinyNode.
- An SIB to format the TinyNode's SD card.

Once these requirements are met, go through the following steps:

1. Make sure that the main power supply is switched off.
2. Remove the TinyNode, both SD cards and both AE sensor from the AEBoard.
3. Apply the STM32 boot mode jumper at the left position and the digital supply voltage jumper at the right position (c.f. Section A.1).
4. Apply a jumper to the STM32 power jumper connector.
5. Connect the AEBoard to a PC using the ST-LINK JTAG programmer connected to the STM32 JTAG port.

[1] Downloadable at http://www.st.com/internet/evalboard/product/219866.jsp

6. Switch the AEBoard's main power supply on.
7. On your PC, open the STM32 ST-LINK Utility software.
8. For AEBoard revision 1.1., make sure to either use a patched ribbon cable (interchanging pins 5 and 13) or enable SWD instead of JTAG communication under "Target - Settings...".
9. Program the STM32 with the `PermAEInit.hex` binary image that can be found in the `Binaries` folder either on the enclosed CD (c.f. Appendix G) or under `/trunk/soft/AEBoard` in the PermaSense SVN repository. This initializes the signal processing with a threshold of *50 dB*, a posttrigger of *400 samples* (800 μs) and a storage block size for raw events of *2048 Bytes* (1016 samples). To change these initialization values, *PermAEInit* needs to be recompiled as described in Appendix B.
10. Insert the STM32's SD card, reset the STM32 and wait until the LED3 (green) is turned on. The SD card is now initialized.
11. Program the STM32 with the `PermAE.hex` binary image that can be found in the `Binaries` folder either on the enclosed CD (c.f. Appendix G) or in the PermaSense SVN repository under `/trunk/soft/AEBoard`.
12. Switch the main power supply off and remove the STM32 power jumper connector.
13. Format the TinyNode's SD card with an SIB and the *SibFormatSDCard* application.
14. Build the TinyOS `PermAE` application (located in the PermaSense SVN repository at `/trunk/soft/tinyos-2.1/apps`) with the desired node ID and flash it to the AEBoard's TinyNode.
15. Insert the TinyNode's SD card and attach the TinyNode on the AEBoard.
16. Connect the sensors to the sensor connectors and the TinyNode's antenna to the TinyNode antenna connector.

The AEBoard is now ready to be deployed.

A.3 STM32 Debugging

In order to more closely monitor an AEBoard's status, a TTL-232R-3V3 USB TTL serial cable [54] may used to connect the console connector to a PC's USB port. If PermAE and PermAEInit for the STM32 were compiled with

the "CONSOLE_ON" option (c.f. Appendix B), the STM32 then outputs debug information over this console port. A terminal emulator (e.g., *minicom* under Linux) may be used to display the AEBoard's data stream which is output at a baudrate of 2000000 bits/s per default. Console communication is unidirectional and the STM32 does not process any data received over the console port.

If the STM32's LED2 (orange) is illuminated, a software failure occurred. This can be one of the following:

- An initialization error. Use the console debugging port to get more information about the initialization error that has occurred.
- The data acquisition has run out of input buffers, probably because the data analysis tasks were too slow in processing the input data. Even though data is lost, the system continues normal operation as soon as empty input buffers are available.

For PermAE code debugging, refer to Appendix B.

A.4 Data Retrieval

A.4.1 Data Sent Over the WSN

The data packets produced by PermAE may simply be retrieved over the PermaSense data webinterface, i.e., `http://data.permasense.ch`. The relevant virtual sensors are called according to the message types specified in Appendix C. On the "Public" GSN instance, data conversion to physical units (c.f. Appendices C and D) has already been performed by GSN.

For a detailed description of the messages produced by PermAE, refer to Section 6.2 as well as Appendix C. For a description of the exact event parametrization algorithm as well as the timing correlation between events, refer to Section 6.4.4.

A.4.2 Raw Event Samples

The raw event data storage on the STM32's SD card does not use a PC readable filesystem. Thus, in order to use the stored data for further analysis, it must be converted into files readable by third party software.

Raw data can be retrieved with the help of a PHP script, reading the raw SD card data stream and writing the raw events to ASCII files. The script is named `convertStoredEvents.php` and may be found in the `AEBoard/Tools`

folder on the enclosed CD or in the PermaSense SVN repository. On a Unix PC, the `dd` tool may be used to forward the raw data to the PHP script. E.g., if the SD card can be accessed at `/dev/sdb` and the generated ASCII files shall be written to `/home/user/RawData`, enter the following command in a terminal:

```
sudo dd if=/dev/sdb | convertStoredEvents.php /home/user/RawData
```

When the command has finished, `/home/user/RawData` contains files named `event_XX.txt`, where `XX` denotes the event's serial number. The output files have a very simple format, e.g.,

```
serial#: 0
startSample: 4166340
length: 1016
threshold: 580
posttrigger: 400
channel: 1
-745
...
-1056
```

The first six lines refer to the relevant information contained in the initialization block (c.f. Section 6.4.6), the rest of the lines contain the event's sample values in PermAE's internal data format (for converting these values to V or dB, refer to D.1.2).

A.5 Sending Commands to the AEBoard

As already described in Section 6.2.2, commands may be sent to the AEBoard in order to enable and disable data acquisition as well as controlling the threshold and posttrigger values used for event detection.

The simplest way to send a command to a deployed AEBoard is via the *dozer_command* virtual sensor on the PermaSense data webinterface. On the GSN homepage, select the *dozer_command* virtual sensor according to your deployment (e.g., *jungfraujoch_dozer_command*, open the "Upload" tab and fill out the form as follows:

- **destination:** The AEBoards TinyNode node ID.
- **command:** Select "AEBOARD_CTL_CMD".
- **arg:** Fill in the encoded message as described in Section 6.2.2.

- **repetitioncnt:** You may enter a number of 3 repetitions.

To send the Dozer command, finally press the "upload" button. You may check your command's effect in incoming AE messages (the *aedata* virtual sensor) for threshold and posttrigger settings, or in the following *aedaqhealth-data* messages for the STM32's power state (c.f. Appendix C).

B

PermAE Developer's Guide

B.1 STM32

For developing the PermAE software on the STM32 processor, the TrueSTUDIO/STM32 IDE from Atollic has been used. A "Lite Version" of this IDE can be downloaded on the Atollic website[1]. TrueSTUDIO is based on the Eclipse IDE framework and offers an integrated workflow for programming the STM32 microprocessors as well as in-circuit debugging.

For simplicity, the `AEBoard` directory, either on the enclosed CD or in the PermaSense SVN repository, directly contains a complete Atollic TrueSTUDIO/STM32 workspace. For adapting the software, it is therefore sufficient to download the IDE and open the `AEBoard` folder when prompted for the desired workspace location.

The three projects contained in the STM32 workspace are described in greater detail in the following sections.

B.1.1 PermAE

PermAE is the main application, running on the STM32 when the AEBoard is in operation. The `/src` directory contains all application source files, i.e., C source and header files.

[1] `http://www.atollic.com/index.php/targets/stm32`

- `PermAEParams.h`: Parameters that affect the application's behaviour, e.g., the static task priorities and buffer sizes. Furthermore, the data output mode can be changed, i.e., if parameters are forwarded to the TinyNode or if they are output on the console port.
- `PermAE.h`: Header file defining all application data structures, i.e., tasks, queues etc.
- `CommunicationProtocol.h`: Definitions for the onboard communication protocol (c.f. Section 6.3).
- `FreeRTOSConfig.h`: FreeRTOS configuration file. For details about configuring FreeRTOS, please refer to the FreeRTOS documentation [56, 57] on the enclosed CD.
- `main.c`: The main program file. Basically, only the AEBoard is initialized and then FreeRTOS is started immediately, starting with the initialization task.
- `task*.c`: The FreeRTOS tasks as described in Section 6.4.
- `ISR_*.c`: Interrupt service routines for data acquisition and console port communication.

The STM32 workspace is configured to enable console text output when compiling PermAE. If console output is not desired for any reason, the symbol *CONSOLE_ ON* must be removed in *Project - Properties - C/C++ General - Paths and Symbols - Symbols*.

The PermAE application links to a `Lib` directory, containing firmware and drivers for the STM32 in the `STM32` directory, AEBoard specific drivers in the `AEBoard` directory as well as the FreeRTOS kernel in the `FreeRTOS` directory. Note that the firmware source files in `STM32` are slightly adapted versions of the original vendor files. Specifically, the following changes to the original files have been made:

- `startup_stm32f10x_hd.s` has been adapted in order to point to the FreeRTOS system hooks.
- `system_stm32f10x.c` has been adapted to expect a 6 MHz oscillator input instead of 8 MHz.

Furthermore, note that FreeRTOS offers three possible memory allocation implementations, defined in `Lib/FreeRTOS/MemMang`. As in PermAE all memory is allocated statically, the simplest implementation (`heap_1.c`) has been used. Make sure that always only one of the three implementation files

is included in compilation. You may edit the according source filter in *Project - Properties - C/C++ General - Paths and Symbols - Source Location*. For more information about memory allocation in FreeRTOS, refer to [56].

B.1.2 PermAEInit

The PermAEInit application must be used to initialize an SD card to be used on an AEBoard (c.f. Section 6.4.8). As some initialization parameters are directly compiled into the application, you may need to adapt PermAEInit accordingly before initializing the AEBoards. The `/src` directory contains all application source files, i.e., C source and header files.

- `AEDetectionParameters.h`: Change this file in order to adapt the initial threshold, posttrigger and storage buffer size parameters (c.f. Section 6.4.8)).
- `AEInitApplication.h`: Some application specific data structures.
- `FreeRTOSConfig.h`: FreeRTOS configuration file. For details about configuring FreeRTOS, please refer to the FreeRTOS documentation [56, 57] on the enclosed CD.
- `main.c`: The main program file. Basically, only the AEBoard is initialized and then FreeRTOS is started immediately, starting with the initialization task.
- `taskInit.c`: The only FreeRTOS task in this application, calibrating and initializing the AEBoard and writing the calibration data onto the SD card.
- `ISR_DAQ.c`: Interrupt service routine for the calibration data acquisition.

PermAEInit is linked to the same `Lib` directory as PermAE. Remember that every change in `Lib` affects both applications (c.f. Section B.1.1).

For further information about PermAE on the STM32, refer to Section 6.4 as well as the comments in the sourcecode.

B.1.3 EventGenerator

The EventGenerator application was used to generate the stimulation events during the temperature chamber tests. It basically outputs a well-defined analog signal pulse at pin PA4 in configurable intervals. The EventGenerator's directory structure is directly based on TrueSTUDIO's template

projects and runs on the STM32Discovery evaluation kit. The whole application code can be found in the `main.c` file. At the top of `main.c`, the parameter definitions may be changed in order to generate different stress periods as described in Section 7.2.

The samples to be output at the STM32's DAC are stored in the `eventBuffer.h` file. This file has been generated using the `genEventBuffer.php` script. For further documentation of the EventGenerator, please refer to the sourcecode comments.

B.1.4 Programming and Debugging the STM32

For programming or debugging PermAE or any other application on the STM32, the default debugging configurations shipping with TrueSTUDIO may be used together with the ST-LINK programmer device. For the AEBoard rev. 1.1, make sure to either use a patched ribbon cable or enable SWD instead of JTAG access in *Debug Configurations - PROJECTNAME.elf - Debugger* (c.f. Section 5.5.1).

For converting the finished software to a `.hex` file that can be flashed to the STM32 using the ST-LINK Utility Software (c.f. Section A.2), the `objcopy.exe` tool from the Codesourcery toolchain[2] may be used. It can also be found on the enclosed CD or in the PermaSense SVN repository at `AEBoard/Tools`. E.g., the following command makes a newly compiled version of PermAE ready for the ST-LINK Utility Software:

```
objcopy.exe -O ihex PermAE.elf PermAE.hex
```

In order to use the EventGenerator software, simply attach your STM32Discovery device to the host PC and use the standard TrueSTUDIO debug configuration.

B.2 TinyNode

The TinyNode part of PermAE is heavily dependent on the PermaSense specific TinyOS libraries. Therefore, its sourcecode is not included on the enclosed CD. Rather, the whole PermaSense specific TinyOS toolchain should be installed as described in the PermaSense wiki[3].

The PermAE application is located in `apps/PermAE`. This directory contains the following files:

[2] `http://www.codesourcery.com/sgpp/lite/arm`
[3] `http://people.ee.ethz.ch/~nccr/permasense/wiki/TinyOS`

- `AcousticEmissions.h`: The main header file, containing all PermAE specific definitions and datatypes.
- `PermAEC.nc` / `PermAEP.nc`: The top level application files.
- `AEModuleC.nc` / `AEModuleP.nC`: The AEModule, handling the communication with the STM32.
- `AEStorageQueue.h` / `AEStorageQueueC.nc` / `AEStorageQueueP.nc`: A slightly adapted version of the standard PermaDozer storage queue (c.f. Section 6.5.3).
- `AEParamControl.nc` / `ResetSplitControl.nc`: Some interface definitions used in PermAE.
- `Makefile`: The application's makefile.

Furthermore, the PermAE application is dependent on some AEBoard specific driver files. These are located in `tos/sensorboards/AEBoard`.

The AEBoard specific PermaDozer datasources can be found in

`tos/lib/PermasenseData/datasources`:

`DataSourceAcousticEmissionsC.nc` / `DataSourceAcousticEmissionsP.nc` as well as `DataSourceAEBoardHealthC.nc` / `DataSourceAEBoardHealthP.nc`.

For further details about the TinyNode part of PermAE, refer to Section 6.5. For details about compiling and installing TinyOS applications on a TinyNode, please refer to the TinyOS and PermaSense documentation.

C

PermAE Output Messages

In addition to some standard PermaDozer messages (eventlogger, rssi, statecounter), PermAE basically produces four different messages containing AE data as well as system status information (c.f. Table 6.1). These messages are specified in detail in Tables C.1 to C.4. Thereby, the following data type conventions are used:

- **uint8** unsigned 8 bit integer number
- **uint16** unsigned 16 bit integer number
- **uint24** unsigned 24 bit integer number
- **uint32** unsigned 32 bit integer number

In the "Conv" column, the conversion function to physical units is given with x denoting the field's value. The "Unit" column then gives the according unit *after* conversion. For a derivation of units, refer to Appendix D. When retrieving data on the "Public" GSN instance (i.e., `http://data.permasense.ch`), data values have already been converted and are displayed in physical units.

aedata				
Message ID:	0xCO			
Field	**Type**	**Conv**	**Unit**	**Meaning**
serial	uint32	x		The event's serial number.
startSample	uint32	$2 \cdot x$	μs	Time of the event's first threshold exceedance.
length	uint16	$2 \cdot x$	μs	The event's length parameter.
risetime	uint16	$2 \cdot x$	μs	The event's rise time parameter.
amplitude	uint16	$\frac{x \cdot 4.5}{65536}$	V	The event's amplitude parameter.
count	uint16	x		The event's pulse count parameter.
energy	uint32	$\frac{x \cdot 4.5^2}{2^{23}}$	V^2	The event's energy parameter.
posttrigger	uint16	x	samples	The posttrigger value used for parametrization.
thresholdChannel	uint8	c.f. D.1.4		Encoding of the threshold value used for parametrization and the AEBoard channel the event was measured on.

Table C.1: Message definition: aedata

aedaqhelathdata				
Message ID:	0xC2			
Field	**Type**	**Conv**	**Unit**	**Meaning**
VccCh1	uint16	$2 \cdot x$	mV	Bias voltage for AE channel 1
VccCh2	uint16	$2 \cdot x$	mV	Bias voltage for AE channel 2
VccAdc	uint16	$1 \cdot x$	mV	ADC supply voltage
RefAdc	uint16	$2 \cdot x$	mV	ADC reference voltage
VccOpa	uint16	$2 \cdot x$	mV	Operation amplifiers' supply voltage
Vcc55V	uint16	$2.4 \cdot x$	V	5.5 V intermediate voltage
I12V	uint16	n/a	n/a	Not valid on AEBoard rev. 1.1 (c.f. Section 5.5.1)
I55VAnalog	uint16	n/a	n/a	Not valid on AEBoard rev. 1.1 (c.f. Section 5.5.1)
I55VDigital	uint16	n/a	n/a	Not valid on AEBoard rev. 1.1 (c.f. Section 5.5.1)
SDEventsFree	uint32	x		Number of free event blocks on the STM32's SD card
DAQPowerState	uint8	x		1: STM32 powered ON 0: STM32 powered OFF

Table C.2: Message definition: aedaqhelathdata

aestatistics				
Message ID:	0xC4			
Field	**Type**	**Conv**	**Unit**	**Meaning**
measured	uint32	x		The number of events detected during the last two minutes.
parametrized	uint32	x		The number of events parametrized and forwarded to the TinyNode during the last two minutes.
stored	uint16	x		The number of events stored on the SD card during the last two minutes.

Table C.3: Message definition: aestatistics

nodehealth				
Message ID:	0x80			
Field	**Type**	**Conv**	**Unit**	**Meaning**
sample	uint16	x		Current PermaSense sample number
uptime	uint24	x	s	TinyNode uptime in seconds
sysvoltage	uint16	$1 \cdot x$	mV	TinyNode supply voltage
sdivoltage	uint16	$6 \cdot x$	mV	12 V external supply voltage
temperature	uint16	$\frac{x \cdot 175.72}{16384} - 46.85$	°C	Onboard temperature
humidity	uint16	$\frac{x \cdot 125}{4096} - 6$	%	Onboard humidity
sibcurrent	uint16	n/a	n/a	n/a
msptemperature	uint16	$\frac{x \cdot \frac{1.5}{4095} - 0.986}{0.00355}$	°C	MSP430 internal temperature
flashStatus	uint16	x	Bytes	SD card storage used
ququeSize	uint8	x	Bytes	Queue buffer used
parentId	uint16	x		The parent node's ID
hopCount / childCount	uint8	x		Dozer hop and child count

Table C.4: Message definition: nodehealth

D

Data Units and Conversion Functions

The units and conversion functions used in this thesis are specified and derived in the following.

D.1 AE Data Units and Conversion Functions

D.1.1 Start Sample, Length and Risetime

As the AE signal sampling rate is fixed to 500 kHz, the smallest time unit in PermAE is 2 μs. Thus, the start sample, event length and event risetime parameters are all measured in multiples of 2 μs (c.f. Table C.1). Thus, with m denoting the time in samples and t denoting the time in seconds, the following conversion functions apply:

$$\begin{aligned} t &= 2 \cdot 10^{-6} \cdot m \quad \text{[seconds]} \\ m &= 5 \cdot 10^{5} \cdot t \quad \text{[samples]} \end{aligned}$$

D.1.2 Amplitude and Threshold

Both amplitude and threshold refer to the signal level. This level can equivalently be measured in V or dB. The input signals at the AEBoard ADCs range from 0 to 4.5 V. This range is sampled with a resolution of 16 bit,

resulting in a base voltage unit of

$$\frac{4.5\,V}{2^{16}} = 68.66\mu V$$

and thus, the conversion functions from

All values in dB refer to a reference voltage of 0.1mV, i.e.,

$$x_{\mathrm{dB}} = 20 \cdot \log_{10}\left(\frac{x_{\mathrm{V}}}{0.0001\,V}\right)$$

with x_{dB} denoting the signal level in dB and x_{V} denoting the signal level in V.

Thus, with x_{s} additionally denoting the signal level in generic PermAE units, the following conversion functions apply:

$$\begin{array}{rcll} x_{\mathrm{V}} & = & 68.66 \cdot 10^{-6} \cdot x_{\mathrm{s}} & [\mathrm{V}] \\ x_{\mathrm{s}} & = & \frac{x_{\mathrm{V}}}{68.66 \cdot 10^{-6}} & [\mathrm{generic}] \\ \\ x_{\mathrm{dB}} & = & 20 \cdot \log_{10}\left(\frac{x_{\mathrm{s}} \cdot 68.66 \cdot 10^{-6}\,V}{0.0001\,V}\right) & [\mathrm{dB}] \\ x_{\mathrm{s}} & = & 10^{\frac{x_{\mathrm{dB}}}{20}} \cdot \frac{0.0001\,V}{68.66 \cdot 10^{-6}\,V} & [\mathrm{generic}] \\ \\ x_{\mathrm{V}} & = & 10^{\frac{x_{\mathrm{dB}}}{20}} \cdot 0.0001 & [\mathrm{V}] \\ x_{\mathrm{dB}} & = & 20 \cdot \log_{10}\left(\frac{x_{\mathrm{V}}}{0.0001\,V}\right) & [\mathrm{dB}] \end{array}$$

D.1.3 Energy

During parametrization, the energy of a sample is calculated as

$$\epsilon = \sum \frac{x_{\mathrm{s}}^2}{2^9}$$

with ϵ denoting the energy and x_{s} denoting the sample value (c.f. Section 6.4.4). Thus, the energy parameter's physical unit is V^2. Given that the basic sample value unit is equal to $\frac{4.5\,V}{2^{16}}$, the energy parameter's conversion functions are

$$\begin{array}{rcll} e & = & \frac{\epsilon \cdot 4.5^2}{2^{23}} & [V^2] \\ \epsilon & = & \frac{e \cdot 2^{23}}{4.5^2} & [\mathrm{generic}] \end{array}$$

with e denoting the energy in V^2.

D.1.4 Threshold and Channel

In order to fit all AE event information into one PermaDozer packet, the threshold used for event detection as well as the channel the event was measured on were encoded into one 8 Byte value (c.f. Table C.1). The thresholdChannel field is built as follows:

$$\mathrm{TC} = 2 \cdot t_{\mathrm{dB}} + c - 1 \qquad c \in [1, 2]$$

where TC denotes the thresholdChannel field value, t_{dB} the threshold value in dB and c the AE channel number the AE event was measured.

Thus, the thresholdChannel field may be decoded as follows:

$$\begin{aligned} t_{\mathrm{dB}} &= \lfloor \tfrac{\mathrm{TC}}{2} \rfloor && [\mathrm{dB}] \\ c &= \mathrm{TC} \bmod 2 - 1 \end{aligned}$$

D.2 System Health Data Conversion

D.2.1 Onboard Voltages

All onboard voltages are sampled with 12 bit ADCs, either by the STM32 or by the TinyNode (c.f. Figure 5.5). The reference voltages for these ADCs is the corresponding device supply voltage, i.e., 2.8 V for the STM32 and 3.3 V for the TinyNode. In order to scale the measured voltages to this range, voltage dividers have been applied (c.f. Appendix E). Thus, with R_1 and R_2 denoting the divider resistors, v_{ADC} denoting the voltage at the ADC and v_{M} the voltage to measure, it holds that

$$v_{\mathrm{ADC}} = v_{\mathrm{M}} \cdot \frac{R_2}{R_1 + R_2}$$

With x denoting the ADC output value received at the data backend and v_{REF} denoting the ADC reference voltage, the following function may be used to convert it back to the original value:

$$v_{\mathrm{M}} = x \cdot \frac{v_{\mathrm{REF}} \cdot \frac{R_2}{R_1+R_2}}{2^{12}} \quad [\mathrm{V}]$$

Table D.1 lists the voltage divider values and conversion constants $c = \frac{v_{\mathrm{REF}} \cdot \frac{R_2}{R_1+R_2}}{2^{12}}$.

Field	Message	v_{REF} [V]	R_1 [Ω]	R_2 [Ω]	c [mV]
VccCh1	aedaqhealthdata	2.8	100k	50k	2
VccCh2	aedaqhealthdata	2.8	100k	50k	2
VccAdc	aedaqhealthdata	2.8	100k	200k	1
RefAdc	aedaqhealthdata	2.8	100k	50k	2
VccOpa	aedaqhealthdata	2.8	100k	50k	2
Vcc55V	aedaqhealthdata	3.3	100k	50k	2.4
sysvoltage	nodehealth	3.3	100k	200k	1
sdivoltage	nodehealth	3.3	100k	15k	6

Table D.1: Onboard voltage constants

D.2.2 Onboard Currents

For current measurements, the MAX9923H current sense amplifiers by Maxim are utilized on the AEBoard [61]. Their output voltage is then sampled with the TinyNode's 12 bit ADC. As the MAX9923H features a constant gain of 100V/V and input voltages are measured over 0.033Ω resistors, the relation between current I and output voltage v_i is

$$I = \frac{100 \cdot v_i}{0.033\Omega} \quad [A]$$

Thus, as the output voltages are not fed through a voltage divider, the following conversion function from a current value x_i (i.e., the I12V, I55VAnalog and I55VDigital fields in the aedaqhealthdata message) to the actual current I applies:

$$I = \frac{100 \cdot x_i \cdot \frac{3.3\,V}{2^{12}}}{0.033\Omega} \quad [A]$$

Note that due to a design error, current measurements on the AEBoard rev. 1.1 are invalid (c.f. Section 5.5.1)

D.2.3 Temperature and Humidity

SHT21 output values

According to the SHT21 datasheet [58], temperature T and relative humidity H may be derived from the raw values x_t and x_h as follows:

$$\begin{aligned} T &= \frac{x_t \cdot 175.72}{2^{14}} - 46.85 \quad [°C] \\ H &= \frac{x_h \cdot 125}{2^{12}} - 6 \quad [\%] \end{aligned}$$

MSP430 Internal Temperature

The MSP430's internal temperature sensor value may be converted to °C as follows [62]:

$$T_{\mathrm{MSP430}} = \frac{x_{\mathrm{msp430}} \cdot \frac{1.5}{4095} - 0.986}{0.00355} \quad [°C]$$

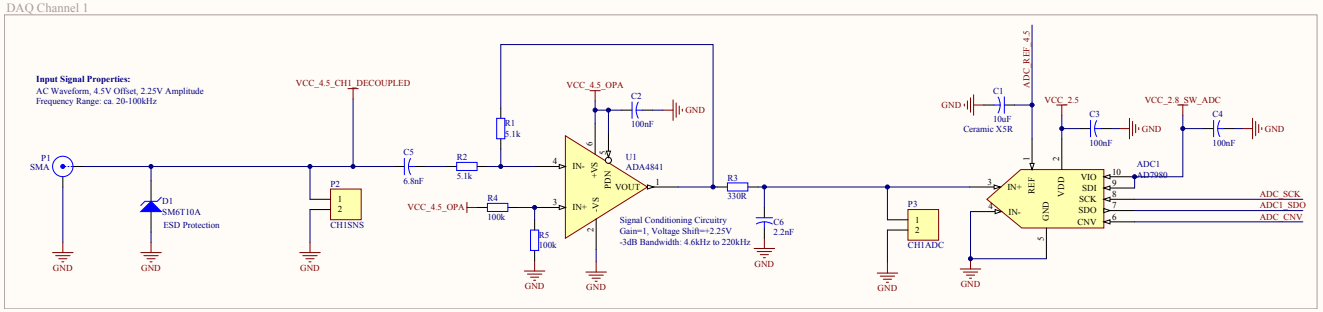
E

AEBoard PCB Schematics and Layout

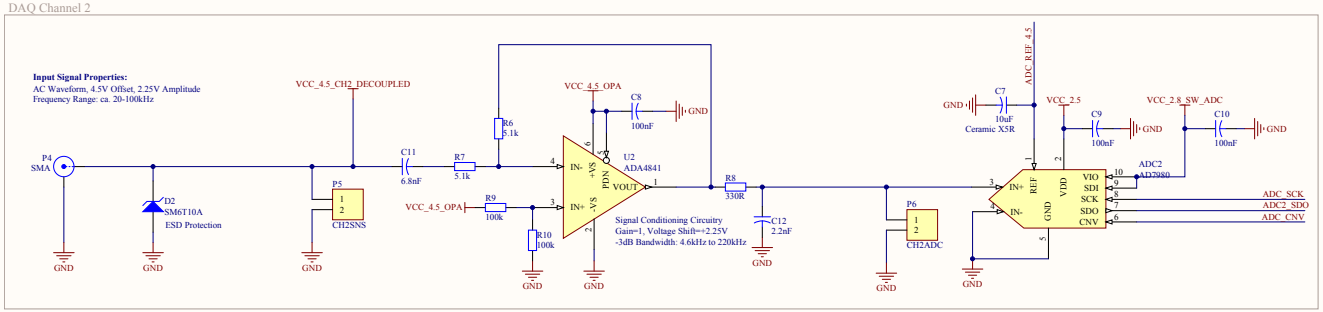
The AEBoard's schematics and PCB layer prints are shown on the following pages. Note that these schematics refer to AEBoard revision 1.1. Some changes have been made for revision 1.2, c.f. Section 5.5.1. The according schematics for revision 1.2 may be found on the enclosed CD (c.f. Appendix G) or in the PermaSense SVN repository under

`/trunk/pcb/projects/0011_acoustic_emission`

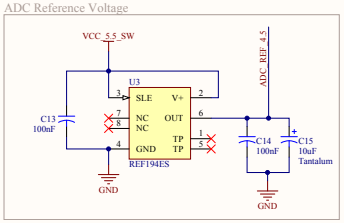
DAQ Channel 1



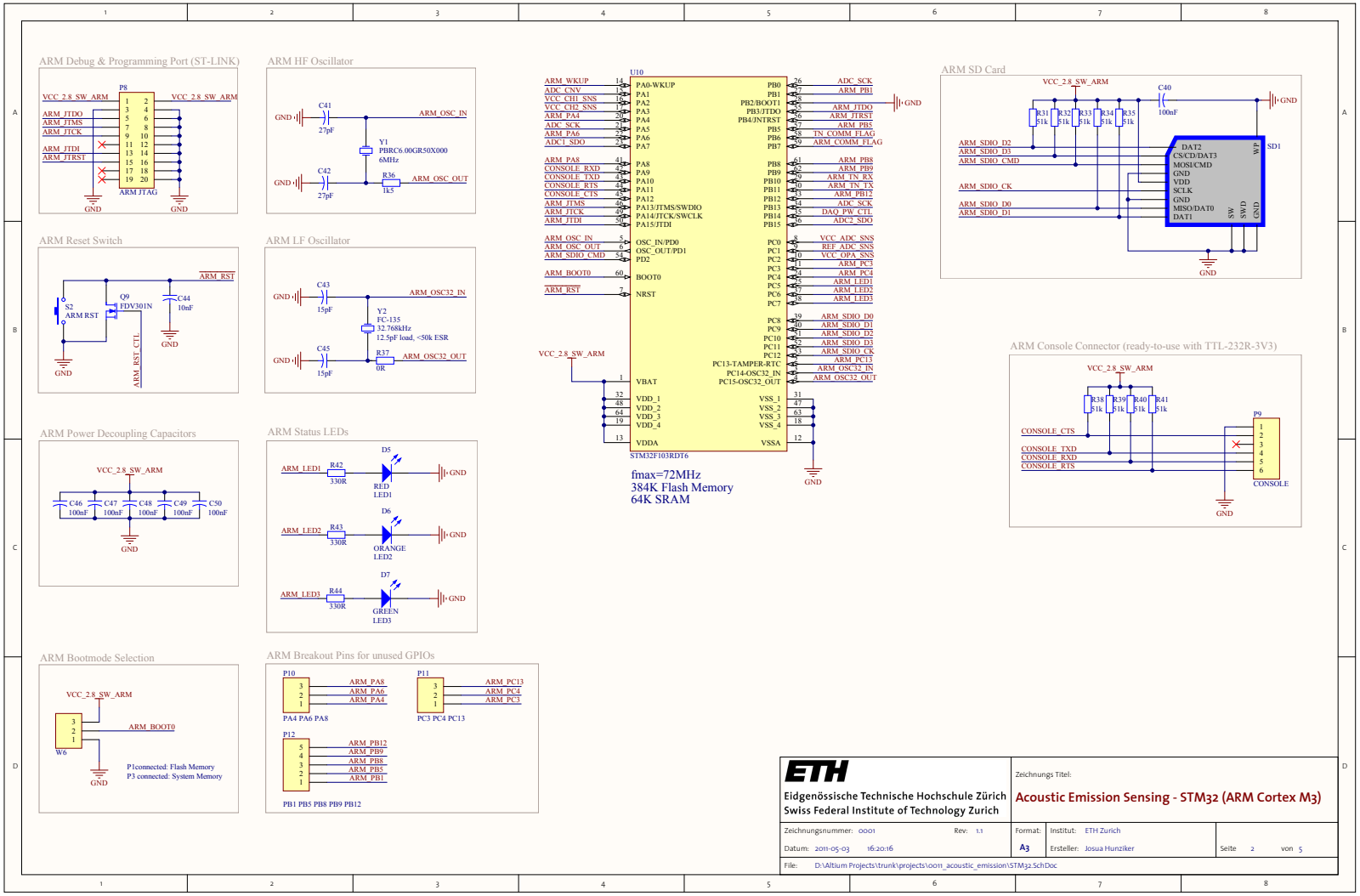
DAQ Channel 2



ADC Reference Voltage



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Zeichnungs Titel: **Acoustic Emission Sensing - Analog Frontend**
Zeichnungsnummer: 0001 Rev: 1.1 Datum: 2011-05-03 16:20:16	Format: A3 Institut: ETH Zurich Ersteller: Josua Hunziker Seite 1 von 5
File: D:\Altium Projects\trunk\projects\0011_acoustic_emission\AnalogFrontend.SchDoc	



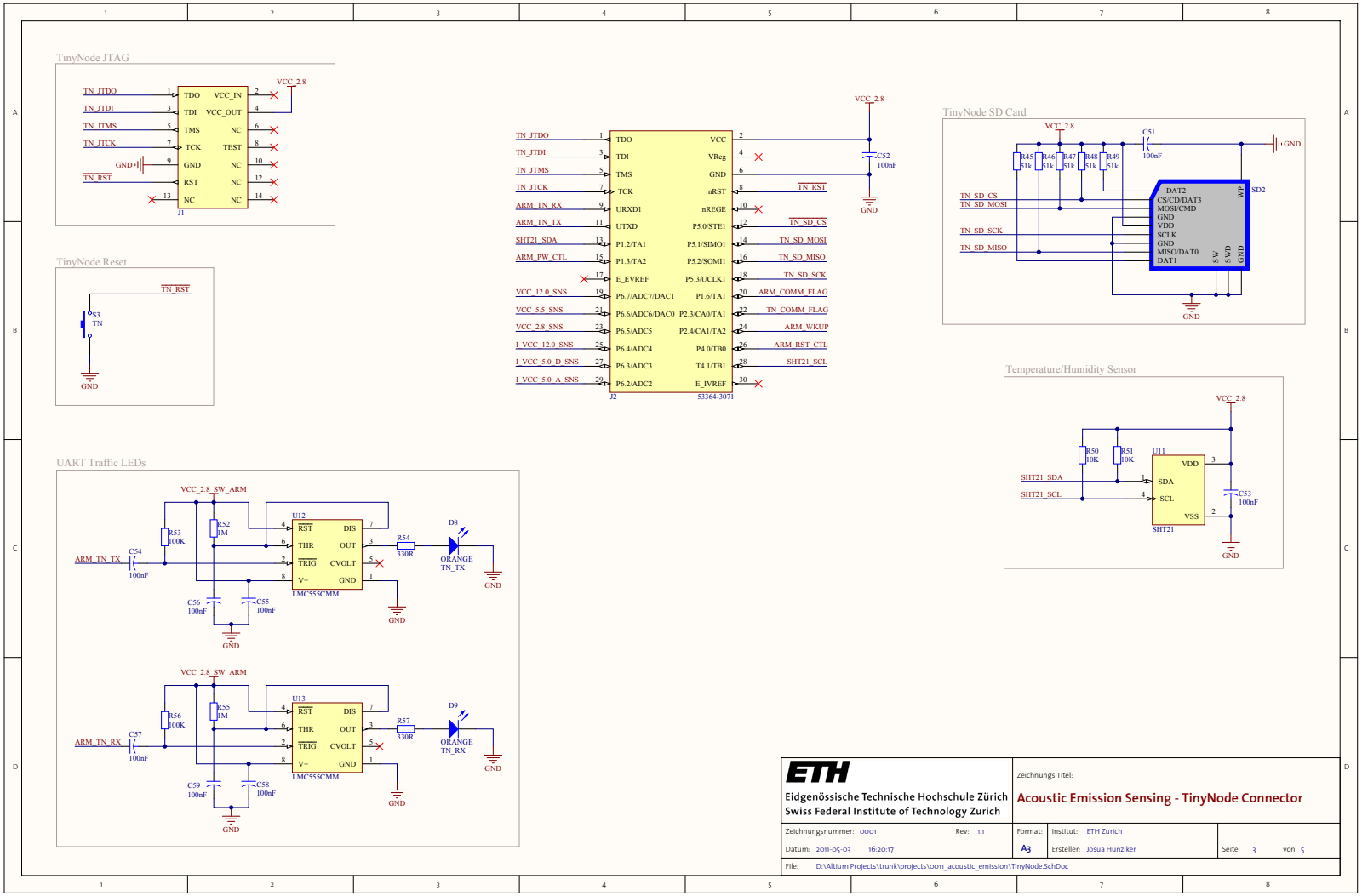
ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Zeichnungs Titel:

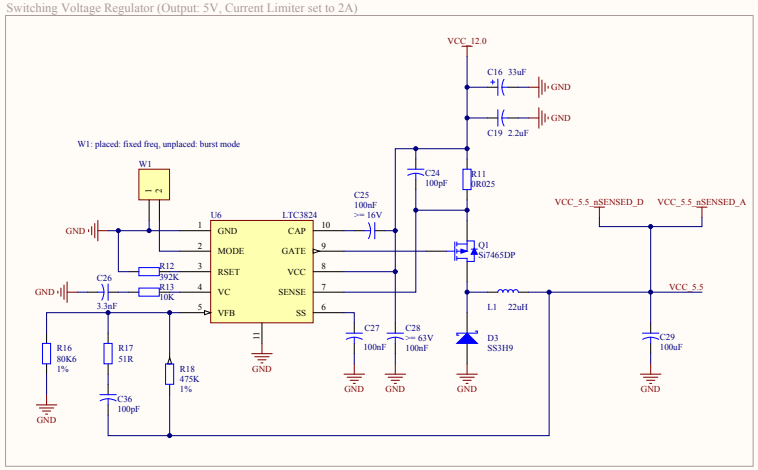
Acoustic Emission Sensing - STM32 (ARM Cortex M3)

Zeichnungsnummer: 0001	Rev: 1.1	Format: A3	Institut: ETH Zurich
Datum: 2011-05-03 16:20:16			Ersteller: Josua Hunziker
File: D:\Altium Projects\trunk\projects\0011_acoustic_emission\STM32.SchDoc			Seite 2 von 5

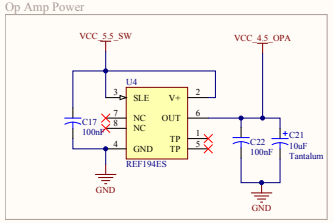


ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Zeichnungs Titel: **Acoustic Emission Sensing - TinyNode Connector**	
Zeichnungsnummer: 0001 Rev: 1.1 Datum: 2011-05-03 16:20:17	Format: A3 Institut: ETH Zurich Ersteller: Josua Hunziker	Seite 3 von 5
File: D:\Altium Projects\trunk\projects\0011_acoustic_emission\TinyNode.SchDoc		

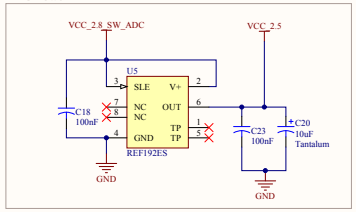
Switching Voltage Regulator (Output: 5V, Current Limiter set to 2A)



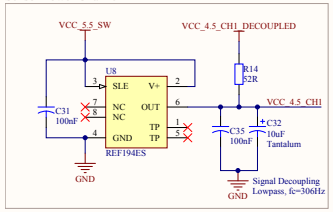
Op Amp Power



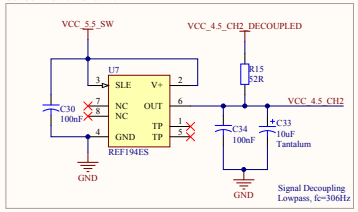
ADC Power



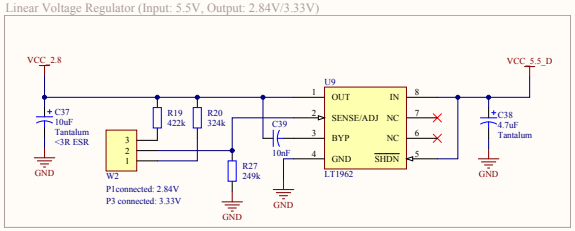
Sensor Power Channel 1



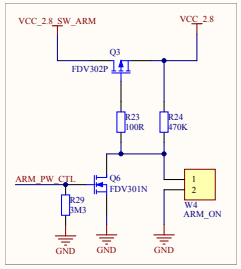
Sensor Power Channel 2



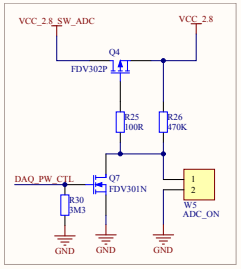
Linear Voltage Regulator (Input: 5.5V, Output: 2.84V/3.33V)



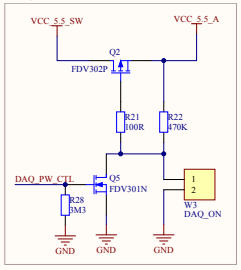
ARM Power Switch



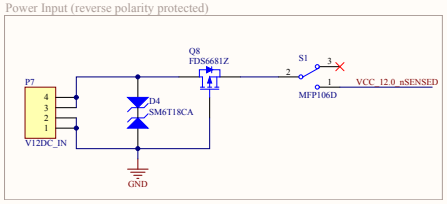
ADC Power Switch



DAQ Power Switch

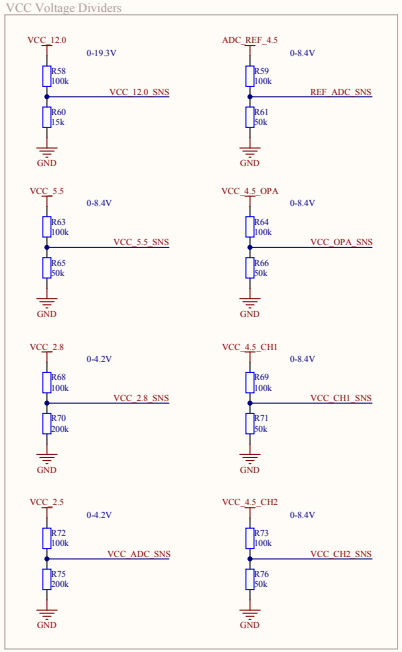


Power Input (reverse polarity protected)

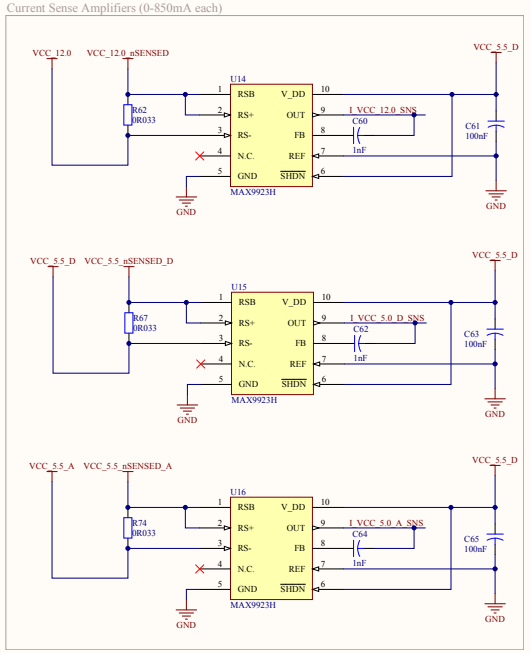


ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Zeichnungs Titel: **Acoustic Emission Sensing - Power Supply**	
Zeichnungsnummer: 0001 Rev: 1.1 Datum: 2011-05-03 16:20:17	Format: A3 Institut: ETH Zurich Ersteller: Josua Hunziker	Seite 4 von 5
File: D:\Altium Projects\trunk\projects\0011_acoustic_emission\Power.SchDoc		

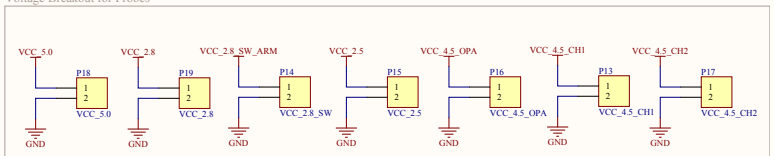
VCC Voltage Dividers



Current Sense Amplifiers (0-850mA each)



Voltage Breakout for Probes



ETH

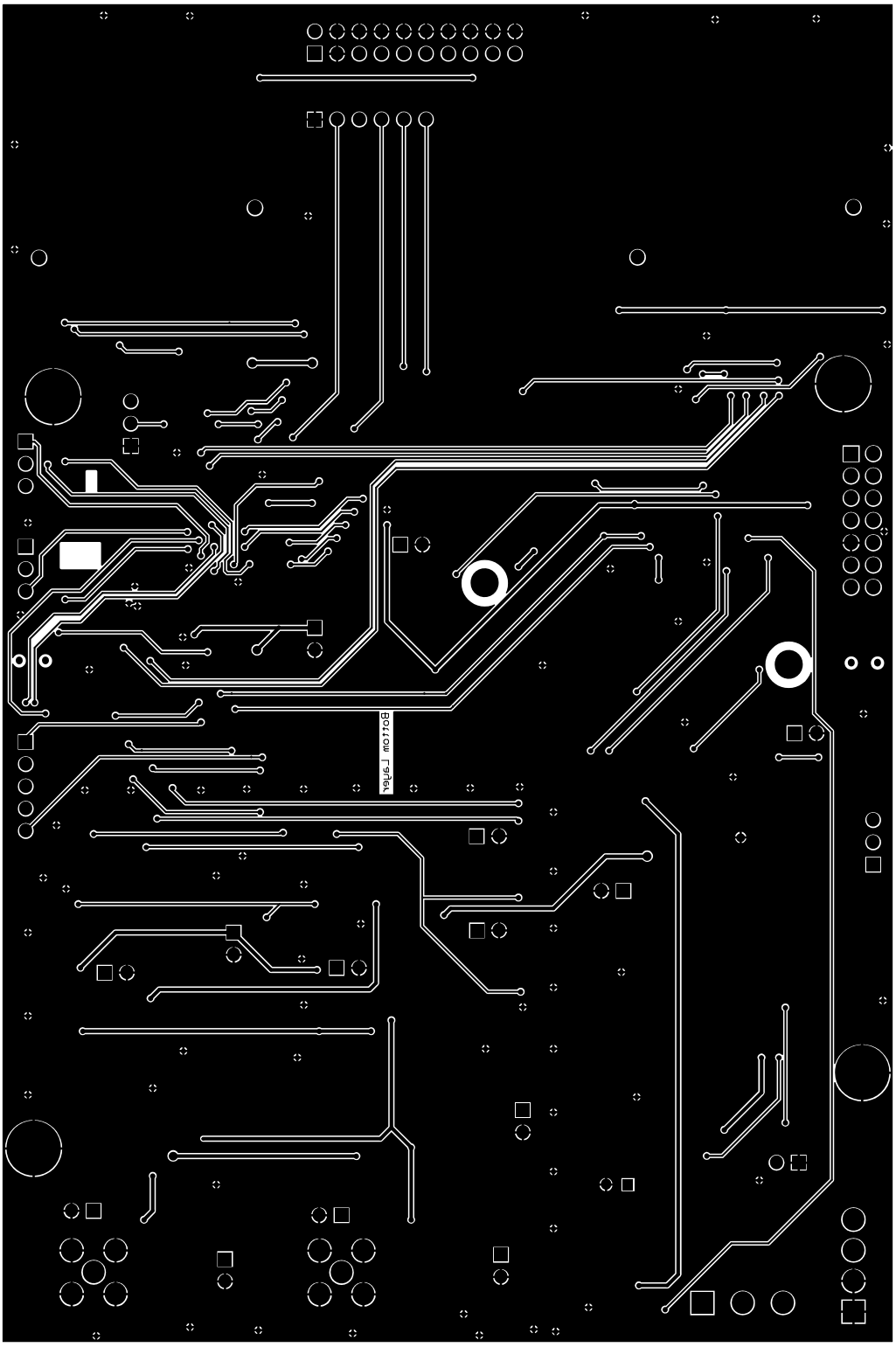
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Zeichnungs Titel:

Acoustic Emission Sensing - Board Health Sensing

Zeichnungsnummer: 0001	Rev: 1.1	Format: A3	Institut: ETH Zurich
Datum: 2011-05-03 16:20:17			Ersteller: Josua Hunziker
File: D:\Altium Projects\trunk\projects\0011_acoustic_emission\HealthSens.SchDoc			Seite 5 von 5

Channel 1
CH1 INPUT
CH1 ADC
ADC VCC
CH1 VCC
Channel 2
CH2 INPUT
CH2 ADC
CH2 VCC
VCC 4.5
DAQ_ON
ADC_ON
Top Layer
B12 B9 B8 B6 B1
STM32 RST
C13 C4 C3
A8 A6 A4
BOOT MODE
STM32
VCC STM32
ARM ON
TN RX
TN TX
LED1
LED2
LED3
STM32 Console
STM32 JTAG
STM32 SD Card
PermaSense
Acoustic Emission Sensing
rev 1.1 (c) ETH 12/2010
VCC 5.0
On
Off
fix frq
12V DC
L: 2.8V
R: 3.3V
VCC 2.8
TN RST
TinyNode JTAG
TinyNode Connector
TinyNode SD Card


Bottom Layer

F

Performance Measurements

F.1 Input Filter Linearity

The following plots show the phase and amplitude compensated error plots as presented in Section 7.1.1.

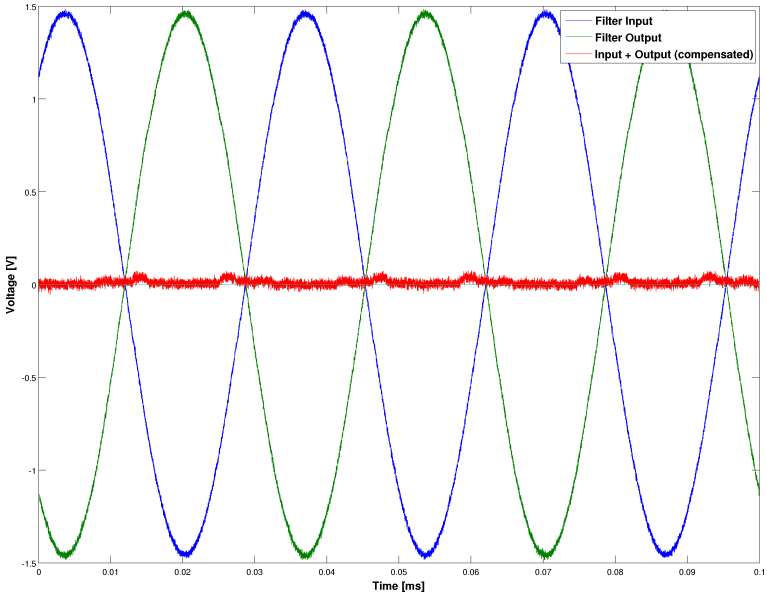


Figure F.1: Phase and amplitude compensated error, no preamplifier, 30 kHz input frequency.

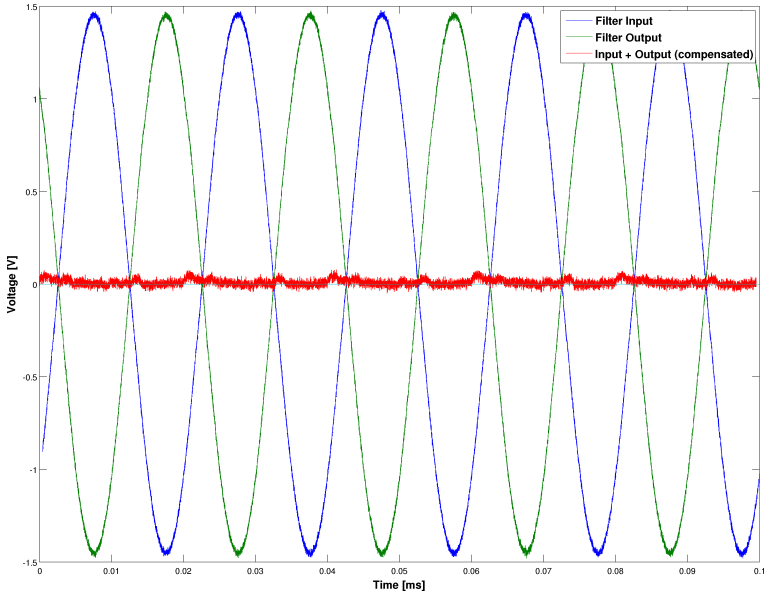


Figure F.2: Phase and amplitude compensated error, no preamplifier, 50 kHz input frequency.

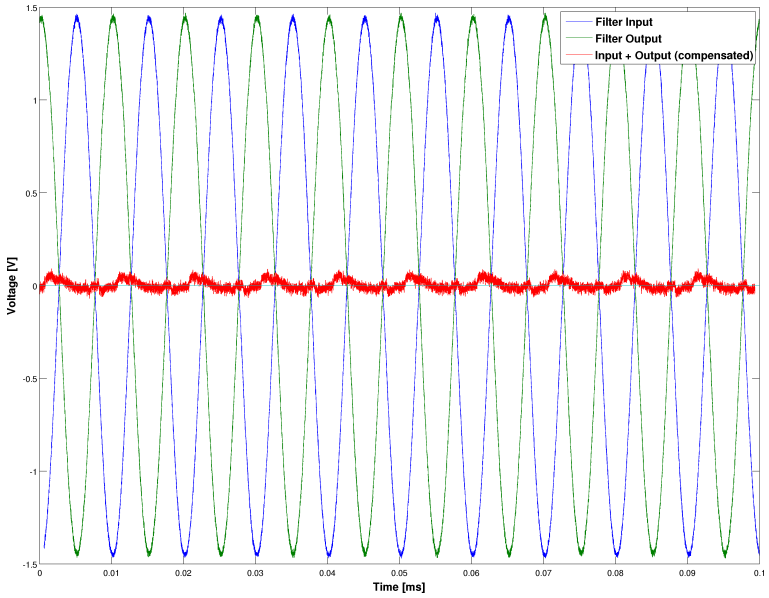


Figure F.3: Phase and amplitude compensated error, no preamplifier, 100 kHz input frequency.

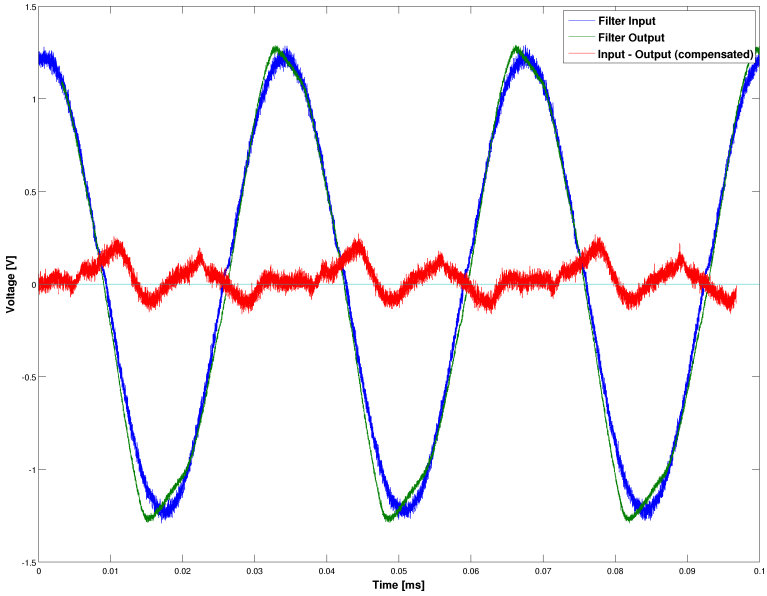


Figure F.4: Phase and amplitude compensated error, with external preamplifier, 30 kHz input frequency.

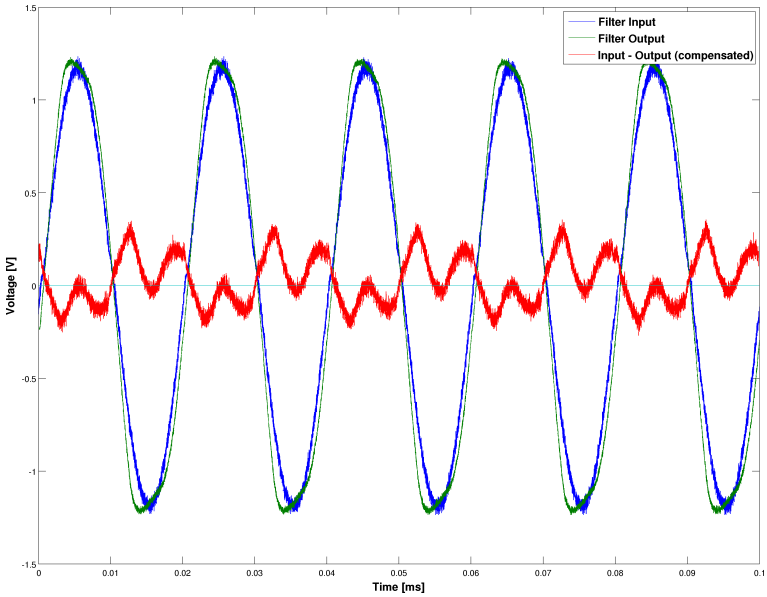


Figure F.5: Phase and amplitude compensated error, with external preamplifier, 50 kHz input frequency.

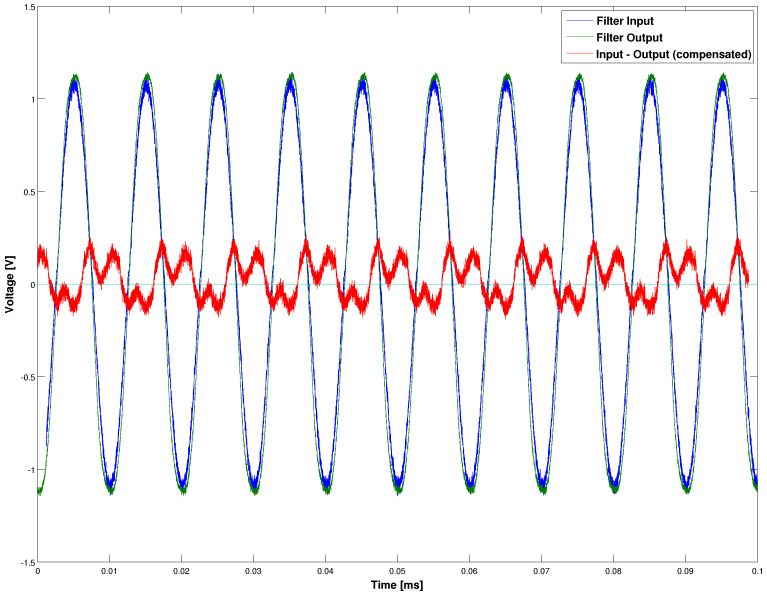


Figure F.6: Phase and amplitude compensated error, with external preamplifier, 100 kHz input frequency.

F.2 Temperature Dependency of AE Parameters

The following figures show the data quality test results for both AEBoard channels as well as for the USB AE Node (c.f. Section 7.2.2).

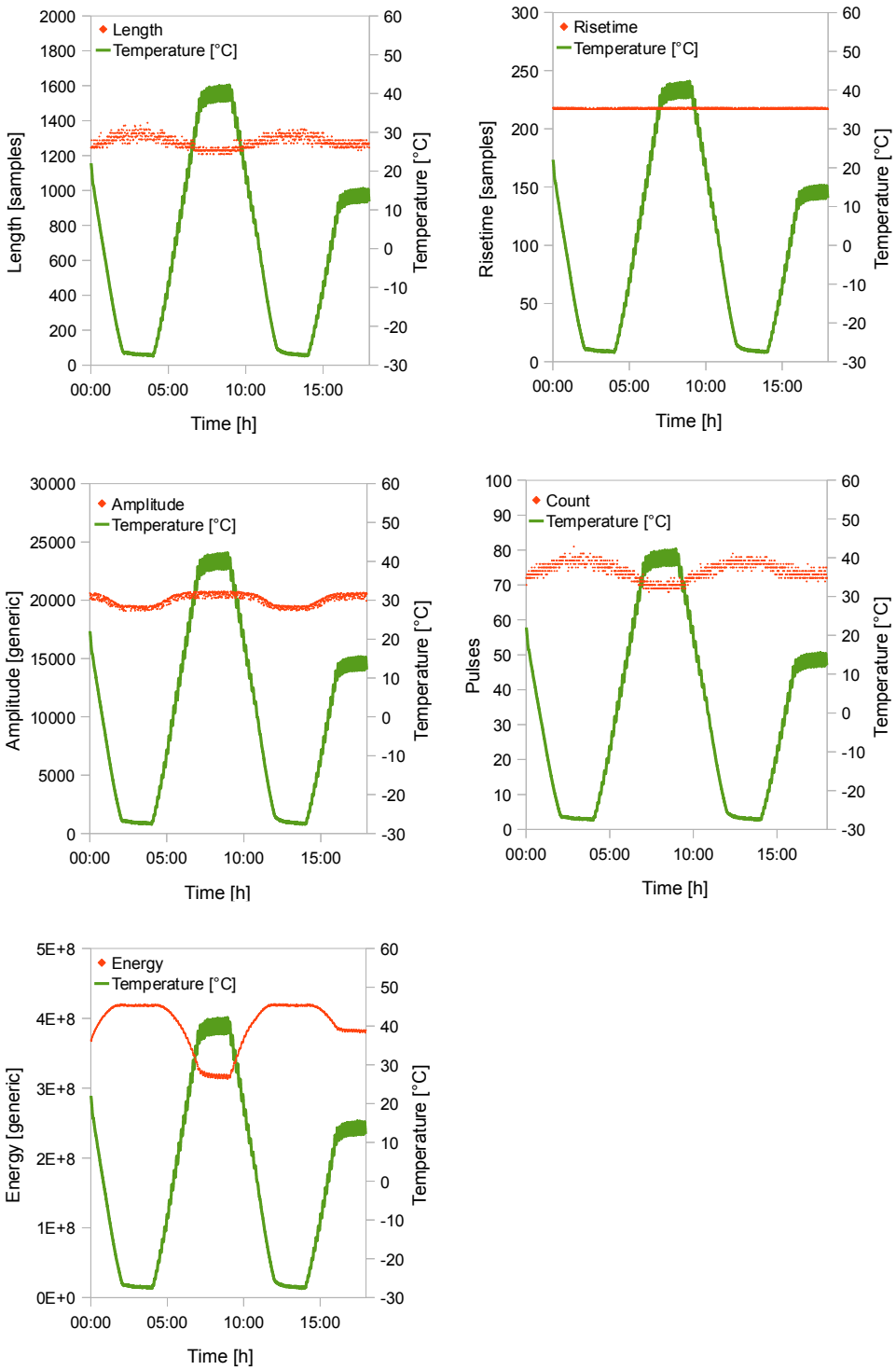


Figure F.7: Data quality test parameter plots for AEBoard channel 1.

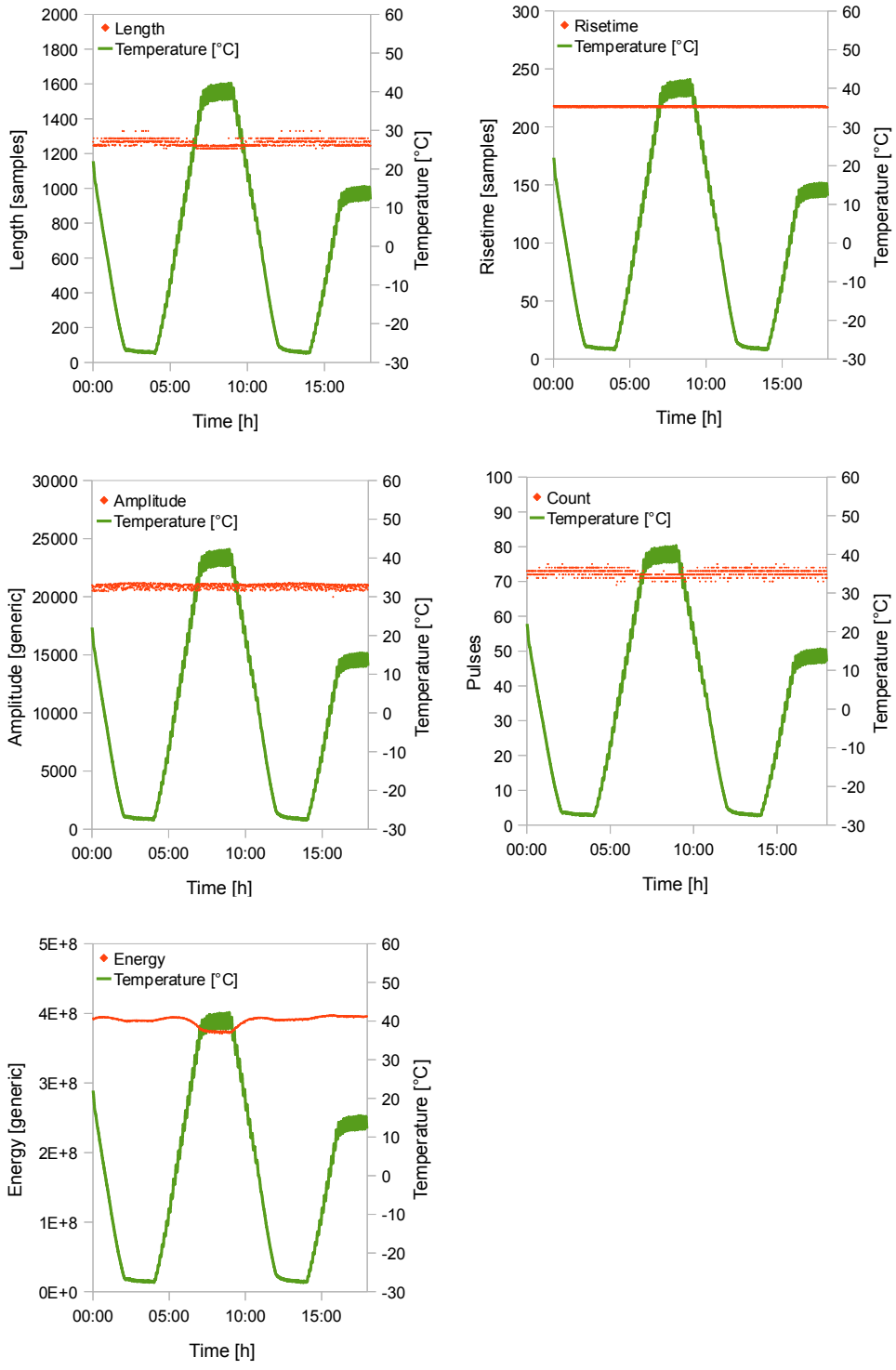


Figure F.8: Data quality test parameter plots for AEBoard channel 2.

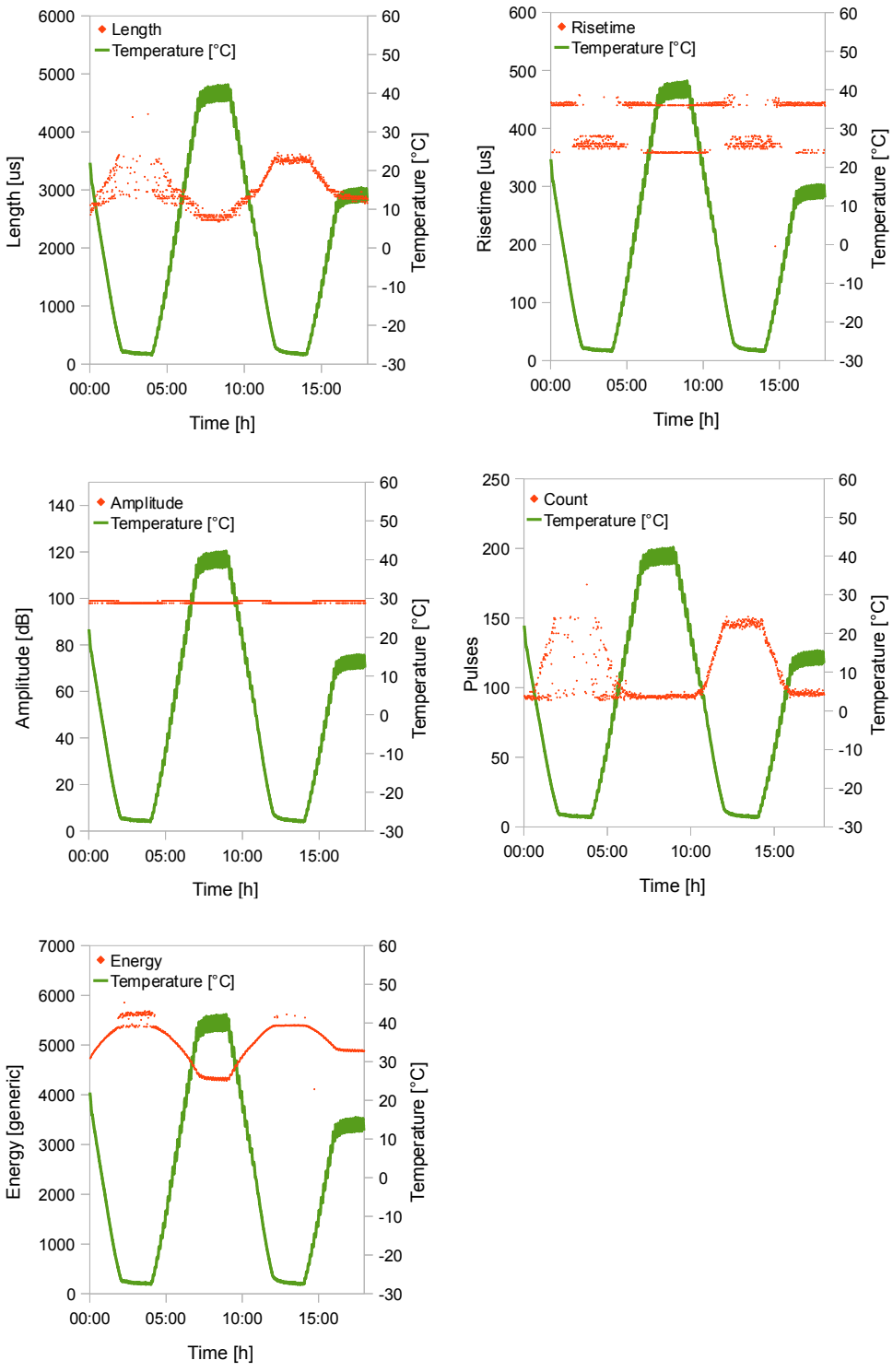


Figure F.9: Data quality test parameter plots for the USB AE Node.

G
CD Contents

The following directory structure is contained on the enclosed CD:

- `AEBoard`
  - `Binaries`: PermAE and PermAEInit binaries (c.f. Section A.2).
  - `PCB Design`: Design files including a detailed bill of material, as well as schematic and PCB prints of AEBoard rev. 1.1 and 1.2.
  - `STM32 Workspace`: Atollic IDE workspace (c.f. Section B.1.1).
  - `Tools`: `convertStoredEvent.php` (c.f. Section A.4.2) and `objcopy.exe` (c.f. Section B.1.4)
- `Datasheets and Manuals`: Datasheets of PCB parts and Physical Acoustics Equipment as well as the FreeRTOS documentation and manuals concerning the STM32 processor.
- `Image Originals`: The source files of some figures contained in this thesis. All figures are in the OpenOffice drawing format.
- `LTSpice`: The LTSpice simulation files for the input circuitry simulation (c.f. Section 5.3).
- `MATLAB Scripts`: MATLAB Scripts for data analysis of AEWin exported data. Refer to the contained README.txt for details.
- `Presentations`: Slides of the initial and of the final presentation.
- `Thesis Sources`: The LaTeX sources of this thesis.

H

Task Definition

The task definition for this master thesis is included in the following two pages.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Lab

ETZ G 75
CH-8092 Zurich

Dr. Jan Beutel
Gloriastr. 35
+41-44-6327032
+41-44-6321035
beutel@tik.ee.ethz.ch
www.tik.ee.ethz.ch/~beutel

MASTERARBEIT
Für
Josua Hunziker

Betreuer: Jan Beutel
Stellvertreter: Stephan Gruber

Ausgabe: 22. September 2010
Abgabe: 06. Mai 2011

Acoustic Emission Sensing for Wireless Sensor Networks

Das PermaSense Projekt befasst sich mit der Beobachtung physikalischer Parameter des Permafrostes in den Schweizer Alpen. Zu diesem Zweck entwickelt und betreibt das Projekt verschiedene Messsysteme. Zu den existierenden Messungen von Temperaturen, Spaltausdehnung und Leitfähigkeitsprofilen sollen nun neue Sensoren entwickelt werden, die im Gegensatz zu den bisher eingesetzten Systemen schneller ablaufende Prozesse erfassen können. Die akustischen Emissionen während der Riss- und Eisbildung im Fels eignen sich hierfür in mehrfacher Hinsicht. Aus geowissenschaftlicher Sicht handelt es sich um relevante, jedoch wenig verstandene Prozesse im Permafrost, aus technischer Sicht stellt sich eine komplexe jedoch überschaubare Aufgabe in der Signalverarbeitung. In dieser Arbeit soll auf Basis der Feldversuche mit Akustik Sensoren am Jungfraujoch im April 2009 ein dediziertes System entwickelt werden.

Das Auftreten relevanter Signaturen in Akustiksignalen ist sporadisch und bedingt durch die stark gedämpfte Ausbreitung im Fels als lokal zu betrachten. Dadurch ergibt sich ein Ansatz mit einer lokalen Vorverabeitung der Signale nahe am Sensor, einer Reduzierung der Datenmenge auf wesentliche Merkmale der zu detektierenden Signaturen und eine anschließende Übertragung mit einem drahtlosen Netzwerk. Der Sensor soll sich, falls möglich, in das stromsparende PermaDozer Netzwerk einbinden lassen und kann ggf. auch eine weitere Netzwerkressource zur Datenübertragung (WLAN) benutzen. Der Sensor soll für den Betrieb im Hochgebirge ausgelegt sein, keine manuellen Interventionen benötigen, und kann bei Bedarf von einer Solarzelle gespiesen werden.

Aufgabenstellung

- Erstellen Sie einen Projektplan und legen Sie Meilensteine sowohl zeitlich wie auch thematisch fest. Erarbeiten Sie in Absprache mit dem Betreuer ein Pflichtenheft.
- Machen Sie sich mit den relevanten Arbeiten im Bereich Sensornetze, piezoakustischen Sensoren, Digitaler Signalverarbeitung vertraut. Führen Sie eine Literaturrecherche durch. Suchen Sie gezielt nach relevanten Publikationen. Prüfen Sie welche Ideen/Konzepte Sie aus diesen Lösungen verwenden können.
- Erstellen Sie eine Übersicht der Anforderungen an einen Akustik Sensor für das PermaSense Projekt. Beziehen Sie hierzu auch die bestehenden Erfahrungen mit ein. In Bezug auf Know-how soll besonders auf das Wissen der Gruppe von D. Amitrano (U.

Grenoble) zurückgegriffen werden. Gegebenenfalls könnte auch ein Besuch im Labor in Grenoble zielführend sein.
- Erstellen Sie eine Funktionsspezifikation auf Basis einer eingehenden Analyse der vorhandenen Rohdaten der Feldversuche.
- Entwerfen Sie auf Basis der Funktionsspezifikation einen lauffähigen Prototyp. Implementieren Sie diesen.
- Validieren Sie den Prototyp mittels eines Referenzsystems.
- Achten Sie darauf, dass die angestrebte Lösung auch in der Praxis anwendbar ist. Dies gilt insbesondere für die Konzepte und Implementierung der Software für die Steuerung.

- Dokumentieren Sie Ihre Arbeit sorgfältig mit einem Vortrag, einer kleinen Demonstration, sowie mit einem Schlussbericht.

Durchführung der Masterarbeit

Allgemeines
- Der Verlauf des Projektes soll laufend anhand des Projektplanes und der Meilensteine evaluiert werden. Unvorhergesehene Probleme beim eingeschlagenen Lösungsweg können Änderungen am Projektplan erforderlich machen. Diese sollen dokumentiert werden.
- Sie verfügen über PCs mit Linux/Windows für Softwareentwicklung und Test. Für die Einhaltung der geltenden Sicherheitsrichtlinien der ETH Zürich sind Sie selbst verantwortlich. Falls damit Probleme auftauchen wenden Sie sich an Ihren Betreuer.
- Stellen Sie Ihr Projekt zu Beginn der Semesterarbeit in einem Kurzvortrag vor und präsentieren Sie die erarbeiteten Resultate am Schluss im Rahmen des Institutskolloquiums Ende Semester.
- Besprechen Sie Ihr Vorgehen regelmäßig mit Ihren Betreuern. Verfassen Sie dazu auch einen kurzen wöchentlichen Statusbericht (email).

Abgabe
- Geben Sie zwei Exemplare des Berichtes spätestens am 06. Mai 2011 dem betreuenden Assistenten oder seinem Stellvertreter ab. Diese Aufgabenstellung soll im Bericht eingefügt werden genauso wie das unterschriebene Unterschriftenblatt „Plagiat“ des Rektorats. Die Entsprechenden Richtlinien des Rektorats sind einzuhalten.
- Räumen Sie Ihre Rechnerkonten soweit auf, dass nur noch die relevanten Quellcode- und Binärdateien, Konfigurationsdateien, benötigte Verzeichnisstrukturen usw. bestehen bleiben. Der Programmcode sowie die Dateistruktur sollen ausreichend dokumentiert sein. Eine spätere Anschlussarbeit soll auf dem hinterlassenen Stand aufbauen können.

Bibliography

[1] "The PermaSense Project," April 2011. [Online]. Available: http://www.permasense.ch 1, 2

[2] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle *et al.*, "PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 2009, pp. 265–276. 2, 8, 57

[3] J. Beutel, S. Gruber, S. Gubler, A. Hasler, M. Keller, R. Lim, I. Talzi, L. Thiele, C. Tschudin, and M. Yücel, "The PermaSense Remote Monitoring Infrastructure," in *International Snow Science Workshop Davos*, 2009. 2, 8

[4] N. Burri, P. Von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*. ACM, 2007, pp. 450–459. 2, 6, 7, 57

[5] J. Hunziker, J. Beutel, and M. Keller, "Online GPS für Permafrost Monitoring," Semester Thesis, Computer Engineering and Networks Lab, ETH Zürich, 2009. 2

[6] P. Guillet, B. Buchli, and J. Beutel, "Online GPS for PermaSense WSN," Semester Thesis, Computer Engineering and Networks Lab, ETH Zürich, 2010. 2, 26

[7] F. Sutton, B. Buchli, and J. Beutel, "GPS-equipped wireless sensor node for permasense," Semester Thesis, Computer Engineering and Networks Lab, ETH Zürich, 2010. 2, 26

[8] M. Allen, L. Girod, R. Newton, S. Madden, D. Blumstein, and D. Estrin, "Voxnet: An interactive, rapidly-deployable acoustic monitoring platform," in *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*. IEEE, 2008, pp. 371–382. 3

[9] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *Proceedings of the 2nd international conference on Embedded networked sensor systems.* ACM, 2004, pp. 1–12. 3

[10] G. Werner-Allen, S. Dawson-Haggerty, and M. Welsh, "Lance: optimizing high-resolution signal collection in wireless sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems.* ACM, 2008, pp. 169–182. 3

[11] C. Grosse, M. Krüger, and S. D. Glaser, "Wireless acoustic emission sensor networks for structural health monitoring in civil engineering," in *European NDT Conference (ECNDT)*, 2006. 3

[12] C. Grosse, G. McLaskey, S. Bachmaier, S. D. Glaser, and M. Krüger, "A hybrid wireless sensor network for acoustic emission testing in shm," in *Sensors and Smart Structures Technologies for Civil, Mechanical and Aerospace Systems 2008*, M. Tomizuka, Ed., vol. 6932, 2008. 3

[13] S. A. Bachmaier, "Event-based acoustic emission technique for structural health monitoring using wireless sensor networks," in *7th fib International PhD Symposium in Civil Engineering*, 2008. 3

[14] Ákos Lédeczi, T. Hay, P. Völgyesi, D. R. Hay, A. Nádas, and S. Jayaraman, "Wireless acoustic emission sensor network for structural monitoring," *IEEE Sensors Journal*, vol. 9, no. 11, pp. 1370–1377, November 2009. 3

[15] N. Gershenfeld, R. Krikorian, and D. Cohen, "The internet of things." *Scientific American*, vol. 291, no. 4, pp. 76–81, 2004. 5

[16] M. Weiser, "Ubiquitous computing," *COMPUTER,*, pp. 71–72, 1993. 5

[17] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002. 6

[18] J. Beutel, O. Kasten, F. Mattern, K. Römer, F. Siegemund, and L. Thiele, "Prototyping wireless sensor network applications with BTnodes," *Wireless Sensor Networks*, pp. 323–338, 2004. 6

[19] I. Akyildiz and M. Vuran, *Wireless Sensor Networks.* LibreDigital, 2010. 6

[20] G. Park, T. Rosing, M. Todd, C. Farrar, and W. Hodgkiss, "Energy harvesting for structural health monitoring sensor networks," *Journal of Infrastructure Systems*, vol. 14, p. 64, 2008. 6

[21] S. Lindsey and C. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *Aerospace Conference Proceedings, 2002. IEEE*, vol. 3. IEEE, 2002, pp. 3–1125. 6

[22] A. Manjeshwar and D. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *ipdps*. Published by the IEEE Computer Society, 2002, p. 0195b. 6

[23] H. Dubois-Ferrière, L. Fabre, R. Meier, and P. Metrailler, "TinyNode: a comprehensive platform for wireless sensor network applications," in *Proceedings of the 5th international conference on Information processing in sensor networks*. ACM, 2006, pp. 358–365. 7

[24] "TinyNode." [Online]. Available: http://www.tinynode.com 7

[25] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer *et al.*, "Tinyos: An operating system for sensor networks," *Ambient Intelligence*, vol. 35, 2005. 7, 57

[26] K. Aberer, M. Hauswirth, and A. Salehi, "A middleware for fast and flexible sensor network deployment," in *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 2006, pp. 1199–1202. 7

[27] M. Keller, M. Yücel, and J. Beutel, "High-Resolution Imaging for Environmental Monitoring Applications," *Relation*, vol. 10, no. 1.149, p. 3558, 2009. 7

[28] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle *et al.*, "Operating a sensor network at 3500 m above sea level," in *Proc. of the 8th Int. Conf. on Information Processing in Sensor Networks (IPSN)*, 2009. 8

[29] NDT Resource Center, "Introduction to Acoustic Emission Testing," April 2011. [Online]. Available: http://www.ndt-ed.org/EducationResources/CommunityCollege/Other%20Methods/AE/AE_Index.htm 8, 9

[30] C. Grosse and M. Ohtsu, Eds., *Acoustic Emission Testing: Basics for Research, Applications in Civil Engineering*. Springer Berlin Heidelberg, 2008. 8, 10, 11, 16

[31] D. Grgic and D. Amitrano, "Creep of a porous rock and associated acoustic emission under different hydrous conditions," *J. geophys. Res*, vol. 114, 2009. 12

[32] T. Spies, J. Hesser, J. Eisenbl
"atter, and G. Eilers, "Measurements of acoustic emission during backfilling of large excavations," in *Proceedings of the Sixth International Conference on Rock Bursts and Seismicity in Mines (Perth)*, 2005, pp. 379–383. 12

[33] G. Manthei, "Characterization of acoustic emission sources in a rock salt specimen under triaxial compression," *Bulletin of the Seismological Society of America*, vol. 95, no. 5, p. 1674, 2005. 12

[34] B. Hallet, J. Walder, and C. Stubbs, "Weathering by segregation ice growth in microcracks at sustained subzero temperatures: verification from an experimental study using acoustic emissions," *Permafrost and Periglacial Processes*, vol. 2, no. 4, pp. 283–300, 1991. 12

[35] J. Murton, R. Peterson, and J. Ozouf, "Bedrock fracture by ice segregation in cold regions," *Science*, vol. 314, no. 5802, p. 1127, 2006. 12

[36] S. Gruber and W. Haeberli, "Permafrost in steep bedrock slopes and its temperature-related destabilization following climate change," *Journal of Geophysical Research*, vol. 112, no. F2, p. F02S18, 2007. 12

[37] A. Hasler, S. Gruber, and W. Haeberli, "Temperature variability and thermal offset in steep alpine rock and ice faces," *The Cryosphere Discussions*, vol. 5, pp. 721–753, 2011. 12

[38] A. Halser, "Thermal conditions and kinematics of steep bedrock permafrost," Ph.D. dissertation, University of Zurich, 2011, draft. [Online]. Available: http://www.geo.uzh.ch/~ahasy/diss_hasler.pdf 12

[39] Physical Acoustics Corporation, "AEWin Software," April 2011. [Online]. Available: http://www.acousticemission.com/index.aspx?go=products&focus=/Software/aewin.htm 13, 68

[40] S. Gruber, D. Amitrano, A. Hasler, J. Hunziker, and J. Beutel, "Taking the lab to the mountains: first experiences with acoustic emission sensing at Jungfraujoch," 2010. [Online]. Available: http://www.mics.org/getDoc.php?docid=3426&docnum=1 14, 20

[41] Physical Acoustics Corporation, "R6α Sensor Datasheet," 2010. [Online]. Available: http://www.pacndt.com/downloads/Sensors/Alpha/R6-Alpha.pdf 16, 36

[42] Maxim Integrated Products, "ADC and DAC Glossary," April 2011. [Online]. Available: http://www.maxim-ic.com/app-notes/index.mvp/id/641 16

[43] Wikipedia, the free encyclopedia, "Nyquist-Shannon sampling theorem," April 2011. [Online]. Available: http://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem 17

[44] D. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952. 26

[45] Analog Devices Inc., "ADSP-2186M DSP Microcomputer Datasheet," 2000. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADSP-2186M.pdf 34

[46] Actel Corporation, "IGLOO Low-Power Flash FPGAs Datasheet," 2010. [Online]. Available: http://www.actel.com/documents/IGLOO_DS.pdf 34

[47] STMicroelectronics, "STM32Fxx Datasheet," September 2009. [Online]. Available: http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00191185.pdf 34, 81

[48] Texas Instruments Inc., "ADS8327 ADC Datasheet," January 2011. [Online]. Available: http://www.ti.com/lit/gpn/ads8327 35

[49] Analog Devices Inc., "AD7908 ADC Datasheet," June 2009. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/AD7980.pdf 35, 36, 50

[50] Physical Acoustics Corporation, "PK6I Sensor Datasheet," 2010. [Online]. Available: http://www.pacndt.com/downloads/Sensors/PK/PK6I%20sensor%2064-08.pdf 36

[51] Analog Devices Inc, "ADA4841 Operational Amplifier Datasheet," December 2010. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADA4841-1_4841-2.pdf 36

[52] Linear Technology, "LTSPice SPICE simulator," April 2011. [Online]. Available: http://www.linear.com/designtools/software/ 36

[53] SD Group, "SD Specifications, Part 1, Physical Layer, Simplified Specification," September 2006. [Online]. Available: http://www.sdcard.org/developers/tech/sdcard/pls/Simplified_Physical_Layer_Spec.pdf 38

[54] FTDI Chip, "TTL to USB Serial Converter Range of Cables Datasheet," September 2010. [Online]. Available: http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232R_CABLES.pdf 39, 80, 82

[55] "The FreeRTOS Project," April 2011. [Online]. Available: http://www.freertos.org 47

[56] R. Barry, "Using the FreeRTOS Real Time Kernel - ARM Cortex-M3 Edition," Real Time Engineers Ltd., 2010. 48, 88, 89

[57] R. Barry, "The FreeRTOS Reference Manual," Real Time Engineers Ltd., 2011. 48, 88, 89

[58] Sensirion AG, "Datasheet SHT21P," May 2010. [Online]. Available: http://www.sensirion.com/en/pdf/product_information/Datasheet_SHT21P_PWM.pdf 58, 100

[59] Saft S.A., "LSH cell range," April 2011. [Online]. Available: http://www.saftbatteries.com/Produit_LSH_cell_range_303_8/Language/en-US/Default.aspx 65

[60] Physical Acoustics Corporation, "USB AE Node Product Bulletin," April 2011. [Online]. Available: http://www.pacndt.com/downloads/LitDirectory/AE/USB_AE_Node.pdf 68

[61] Maxim Integrated Products, "MAX9923H Datasheet," 2010. [Online]. Available: http://datasheets.maxim-ic.com/en/ds/MAX9922-MAX9923.pdf 100

[62] Texas Instruments Inc., "MSP430x1xx Family," 2006. [Online]. Available: http://focus.ti.com/lit/ug/slau049f/slau049f.pdf 101