NTNU

Norwegian University of
Science and Technology

Adaptive Store and Forward

A look into possible solutions for the
increasing resource consumption in wireless
sensor networks

Gunnar Ranøyen Homb

Master of Science in Electronics
Submission date: June 2016
Supervisor: Kjetil Svarstad, IET
Co-supervisor: Ragnar Ranøyen Homb, Norwegian Creations

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Abstract

Over the recent years the interest for wireless sensor networks and ”Internet of Things” (IoT) has grown significantly. As a consequence of this, there is a need for solutions to conserve restricted resources such as energy and network usage. This thesis presents a look into possible solutions to meet the new demands. A complete test platform is presented and established. A wireless sensor network based on the protocol 6LoWPAN is used as a test bench. In addition a custom energy measurement system is designed, implemented and physically built. Despite some challenges with the physical assembly, several tests was conducted. The tests does indicate that the presented store and forward method could possibly reduce the energy consumption in such systems. This indication was ≈ 10 % in potential energy savings over a duration of 100 Seconds for a single node. A list of tasks for further work is also presented together with possible improvements.

Sammendrag

I løpet av de siste årene har interessen for trådløse sensornettverk og ”Internet of Things” (IoT) har vokst betydelig. Som en konsekvens av dette, er det et behov for løsninger for å spare energi og nettverksbruk i systemer med begrensede ressurser. Denne avhandlingen presenterer en titt på mulige løsninger for å møte de nye kravene. En fullstendig testplattform er presentert og etablert. Et trådløst sensornettverk basert på protokollen 6LoWPAN er brukt som en testbenk. I tillegg har et tilpasset energi-målesystem blitt konstruert, implementert og fysisk bygget. Til tross for noen utfordringer med den fysiske monteringen, ble flere tester utført. Testresulatene indikerer at den presenterte ”store and forward” metoden kan muligens redusere energiforbruket i slike systemer. Denne indikasjonen var på ≈ 10 % i potensielle energibesparelser over en test med varighet på 100 sekunder for en node. En liste over oppgaver for videre arbeid er også presentert sammen med mulige forbedringer.

Table of Contents

List of Tables

List of Figures

Abbreviations

6LoWPAN IPv6 over Low power Wireless Personal Area Networks

ADC Analog to Digital Converter

API Application Programming Interface

ARM Cortex-M Group of 32-bit RISC ARM processor cores

BGA Ball Grid Array (surface-mount packaging technology)

CMS Custom Energy Measurement System

CSV Comma-Separated Values

DMA Direct Memory Access

DUT Device Under Test

EFM32 Family of 32-bit microcontrollers from Silicon Labs

EMA Exponential Moving Average

emlib Silicon Labs low-level peripheral support library

I²C Inter-Integrated Circuit bus

IC Integrated Circuit

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

JTAG Joint Test Action Group (Test and programming interface)

LAN Local Area Network

MCU Microcontroller Unit

MMU Memory Management Unit

MQTT Light weight messaging protocol (formerly MQ Telemetry Transport)

PCB Printed Circuit Board

PRS Peripheral Reflex System (Silicon Labs term)

RAM Random-Access Memory

RTC Real-Time Clock

SMD Surface-Mount Device

SPI Serial Peripheral Interface

SWD Serial Wire Debug

SWO Serial Wire Output

UART Universal Asynchronous Receiver/Transmitter

USART Universal Synchronous/Asynchronous Receiver/Transmitter

WiFi Wireless LAN technology

WSN Wireless Sensor Network

Part I

Overview

Chapter 1

Introduction

1.1 Conserving energy

Lately the interest for wireless sensor networks has grown significantly. The introduction of the term "Internet of Things" does probably have to take some of the blame. In such networks, the energy consumption have a huge impact on several factors such as lifetime, wireless range, sensor opportunities and usage possibilities. In fact, lowering the energy consumption is one of the key requirements needed to successfully deploy wireless sensor networks. If we take a look at the history, the wireless sensor networks and the use of low power techniques in such networks is not anything new and there exists already numerous research in the field. But as the report (Gunnar Ranøyen Homb, 2015) concludes, there is still several issues that could be investigated in an attempt to further minimize the energy consumption.

1.2 Report structure

This assignment investigates the potential for energy savings in a wireless sensor network. The contributions developed from work behind this report is summarized in section 1.5. The contributions and the result establish a foundation for further work on the topic.

Report Layout In this chapter, the problem description, methodology and motivation behind this work is presented together with a short description of the issuer of the assignment. Thereafter in chapter 2, background is presented for establishment of fundamental knowledge needed for this report.

The report is divided into six main parts. Part I is the part establishing an overview over the report. This part contain the introduction and the background chapter (2).

In part II the complete test setup is described. Chapter 3 introduces the test setup and presents the test objectives. The three subsequent chapters 4, 5 and 6 goes through the WSN platform, energy consumption sources and the measurement setup.

A custom energy measurement system designed for smaller wireless sensor nodes is presented in part III. The measurement system is presented in chapter 7. This chapter contains the requirements and foundation behind the implementation. In chapter 8, a detailed walkthrough of the implementation is presented. Chapter 9 and 10 presents the hardware, PCB layout and assembly. Chapter 11 present the firmware made for the measurement system. While chapter 12 presents the analyzer software made to analyze the measurement data. Both the firmware and software chapter is organized with a presentation of the design concepts first, then a presentation of the construction and system operation.

In part IV the testing described in part II is executed and analyzed. In chapter 13, 14 and 15 three different tests is covered. Within each chapter, the implementation, execution, results and analysis is presented. Finally, chapter 16 summarize results from the three tests.

Second to last part of the thesis is part V. This part cover a discussion in chapter 17 before a final conclusion is made in chapter 18. The last part of the thesis is part VI which contains all of the appendices.

1.3 Problem Description

Emerging use of sensors on new areas and connectivity to the Internet introduces new challenges. With an increasing amount of sensors together with a rapid development of factors such as sensor speed and resolution, rising resource consumption is an unavoidable side effect. Therefore, solutions to handle this increasing demand is important. This demand is both relevant when looking at power consumption, but can also be a challenge for the total network capacity. One possible solution to solve the resource demands is implementation of new data handling methods. The traditional way is to send all the gathered data upwards to a processing server. This scheme is static and does not reflect the flexible nature of a wireless sensor network. By implementing more adaptive solutions to decide whether data should be sent upwards or stored locally for later use, savings could possibly be achieved, in both power consumption as well as network usage.

The assignment is an examination of whether and how an ”adaptive store and forward” solution could improve a wireless sensor network. To approach the task, several work goals are listed:

- Creation of a test system including appropriate hardware to survey the topics of this thesis
- Data-gathering from a traditional data-harvesting-method to establish a baseline for the challenge being looked at
- Development of ”store and forward” method

- Benchmark the store and forward method with respect to:
  - Energy consumption
  - Network usage
- Comparison and analysis of the test results excluding and including the developed methods
- Investigate opportunities for improvements

The author of this report have emphasized the establishing of a robust test system through testing the fundamentals first and subsequently add more tests one by one. This ensures that a robust test and research platform is established.

1.4 Methodology

This report cover multiple aspects of applied methodology. The reason for this is that the work cover both theoretical and practical issues. Before working with the different topics they have been studied and mapped in advance. It allows the author to gain an overview and knowledge of the area.

Relevant material has been searched for in several locations. In search for already published material the main searching engines used are: NTNU's university library searching engine Oria, Google scholar, Google Books, IEL (IEEE Xplore), ACM Digital Library and SpringerLink. The basic "Google" search tool have been used in search for commercial material such as datasheets and existing solutions. The choice of knowledge sources have been based on purposeful choices done by the author.

When working with the hardware prototyping, the philosophy "Think twice, act once" has been used. The author have positive experiences with this philosophy earlier and this has several times proven to reduce the time needed to complete a given task.

Despite this philosophy, there have been complications regarding the hardware production work during this thesis. When designing and implementing a physical system, unpredicted problems can occur. In this case the external hardware production took much longer time than predicted. In addition the delivered hardware had several problems and necessary debugging and correction was time consuming.

1.5 Contributions

The following bullet points summarize the contributions from the presented work and report:

- Design of a customized measurement system:
  - Circuit design
  - PCB design

  - Documented implementation
  - Assembly and debugging
  - Fully functioning firmware
- Written analysis software tool for use together with the custom measurement system
- Creation of a test platform:
  - Selection of hardware and software
  - Implementation of a wireless sensor network based on 6LoWPAN and Contiki OS
  - Identify energy consumption sources
  - Planning of the test program
- Execution of tests based on the test platform and analysis tool
- Result analysis
- Discussion and evaluation of the results
- Investigation of possible improvements

1.6 About Norwegian Creations

At Norwegian Creations, the vision "Bridging the Gap Between Idea And Creation" is central. Norwegian Creations gives tools and resources to good ideas and creative people. In this way there can arise a very motivating value generation.

To satisfy people's embedded motivation to create something tangible, Norwegian Creations have developed a system to easily take ideas through phases such as development, market validation and production. When the creation finally reaches a customer, all involved human resources on the project will get ownership in the new companies generated by this creation.

You can read more about Norwegian Creations and their projects here: http://www.NorwegianCreations.com

1.7 Motivation

1.7.1 The Industry

The industry has applied wireless sensor networks for about a decade already. Now, the industry have some major driving forces utilizing wireless sensor networks together with the Internet of Things. This allows ubiquitous sensors with the possibility for big data analytics, supply chain analytics, etc. With the customers nodes connected to a centralized powerful platform the industry may deliver services they couldn't done without the connectivity.

1.7.2 Norwegian Creations

Norwegian Creations has initiated this work on wireless sensor networks based on the need for sensors that could be easily deployed in certain applications. These applications could cover a broad range of challenges from small scale data acquisition tasks to more demanding industrial applications.

1.7.3 My personal

My motivation for this work are mainly based on four different aspects. Before last semester I had no detailed knowledge about wireless sensor networks. As I researched the field throughout the semester, my knowledge and motivation increased significantly. The fact that wireless sensor networks is currently gaining interest because of the Internet of Things is also a motivating factor. With this increased interest, the market is also innovating and changing rapidly, this is also motivating reason. The last primary motivation is my desire to try to produce something useful for both myself and others around me.

Chapter 2

Background

This chapter will establish fundamental knowledge and define different main subject areas in context of this thesis. At the end of the chapter the wireless sensor network platform is reviewed and selected.

2.1 Energy Measurement

Energy consumption could be summarized with the formula in equation 2.1. For a system operating on battery power the energy consumption is a key figure of measure in regards to system lifetime. This is because a battery only have a finite amount of energy stored. In order to measure the energy consumption, the power consumption must be measured together with the time duration.

$$E_{(J)} = P_{(W)} * t_{(s)} \tag{2.1}$$

In a embedded system such as a WSN node, there is multiple sections of the system that contributes to the total energy consumption. Figure 2.1 shows an example of the distribution within a such system. All of the numbers in figure 2.1 is fictive numbers for presentation purpose.

In the work conducted in this thesis one of the main purposes is to test how a described method could possibly conserve energy. Therefore it's a key requirement to be able to measure the energy consumption of a such system in order to solve the task.

To measure energy in the work conducted, the equation 2.1 is used to transfer measured power consumption and time to energy. There exists several methods for measuring the electrical power consumed by a system at a given time. The used method is a current shunt resistor. This is consisting of an accurately known resistance which is placed in series with the load. When current flows through the shunt, a voltage drop will occur over the shunt resistor in accordance to Ohm's law. In addition the supplied voltage is measured and hence the power consumed by the circuit could be calculated.

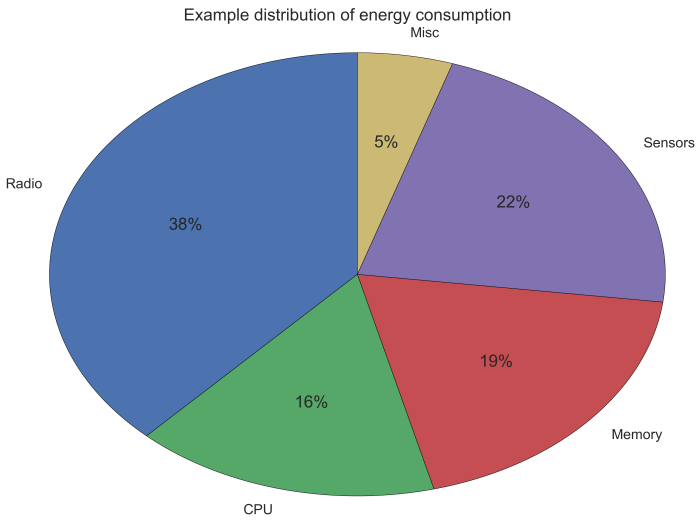


Figure 2.1: Example distribution of energy consumption. Note that the numbers is fictive for presentation purpose

2.2 Store and Forward

Store and forward is a technique where data is sent to a intermediate station where it's stored, before at a later point in time is sent to the final destination. This technique is widely adapted in telecommunication networks where the network connectivity requires high mobility. It's also used as a method to process packets inside certain network switches (Intel Corporation, 2014).

In the work conducted in this thesis the method has been used in a different context. Therefore the behavior of the method is slightly adapted. The following behaviour is used in this work: When a node in the sensor network gathers data (eg. sensor) the information isn't necessarily sent directly to a central server. The information could be stored locally or on a nearby node to conserve energy. The stored data is then sent at a later point in time, such as when a subscriber requires the data or that a memory limit has been reached. It's a requirement in this work that the same amount of data should be transferred between a node and the server, but not necessarily at the same time.

Hereafter when the "store and forward" method is mentioned, it's this adapted behaviour that's meant.

2.3 Adaptive Mechanism

In this thesis the term adaptive mechanism is used for a software mechanism that could adopt to external changes. With the implementation of store and forward

method, data is stored locally on the nodes. This does break the connectivity and in order to fix this an adaptive mechanism is added. This mechanism could ensure that the data is present at the destinations when it's needed. In order to optimize the network and energy usage this mechanism would need to adopt to environmental changes as well as constantly changing usage patterns. Therefore the mechanism must be adaptive in order to release the potential of the flexible nature of such systems.

2.4 Law of Large Numbers

The law of large numbers is a probability theorem. According to the theorem, when a large numbers of trials in a experimented is preformed, the average of the results is close to the expected value (Shiryayev, 1992). For this thesis the theorem is used when conducting energy measurements. Due to random events when measuring the energy consumption, a single experiment doesn't necessarily obtain results close to the expected value. To obtain a result closer to the expected value this theorem is used and a series of experiments is conducted. In this way the results obtained from the experiments could be closer to the reality.

2.5 WSN Platform

In order to implement and test the hypothesis from section 1.3 a wireless sensor network platform must be established. There exists a lot of solutions and vendors that could provide a such platform. Both the hardware and the software toolchains does vary a lot. They differ in openness (or closeness), complexity and opportunities. Some solutions does also have a expensive investment cost in order to acquire correct license for usage.

With the amount of solutions available this section can't cover them all. Therefore only the selected solution is presented here. In order to select this solution a small study was conducted.

The selected solution is based on the Texas Instruments CC2650 MCU on a development platform called "SensorTag" (Texas Instruments Incorporated, 2016c). More specifically, the SensorTag 2.0 was selected. Figure 4.1 shows the physical SensorTag. The CC2650 MCU is targeted applications utilizing one of the protocols Zigbee, 6LoWPAN or Bluetooth Smart. The device support IEEE 802.15.4 MAC layer, which is used in many wireless sensor networks (Texas Instruments Incorporated, 2015a).

To establish a wireless sensor network the protocol 6LoWPAN was selected as a foundation. Amongst others, one reason is that this protocol has IPv6 connectivity directly at each node. In this way there is established a very transparent communication scheme, suited for future implementations. The specific implementation details and set up is covered in chapter 4.

Hereafter the SensorTag 2.0 is referred to as SensorTag or DUT.

Part II

Test Setup

Chapter 3

Introduction & Test objectives

3.1 Introduction

In this part (II) of the thesis the complete test setup is described. In section 3.2 the objectives for the testing is established before chapter 4, 5 and 6 are going through central aspects of implementation and how the work have been done in order to try to fulfil the established objectives.

In this thesis the store and forward method is applied in a different context than in the traditional telecommunication networks. The technique is applied in an attempt to conserve energy. In a wireless sensor network there is no doubt that the radio transmissions does consume a great amount of energy. Therefore the method is used to try to reduce the amount of radio communication and temporarily store the data at the node. In this way the information could be sent in bigger packets with a lower frequency. With bigger packets, the overhead is possibly reduced and energy is also possibly conserved. Another factor is that the radio consume energy to wake up and go to sleep again. With lowered frequency and fewer wake/sleep transitions, even more energy savings could be achieved.

But this hasn't been investigated or tested, therefore the actual degree of improvement is unknown. In order to find out, tests of the store and forward method should be conducted.

In a wireless sensor network the connectivity is often realized as a dense mesh network. In other words, data packets is often relayed through several nodes before reaching it's destination. This does also mean that the store and forward method could possibly conserve energy on both the node generating data and on all of the relaying nodes as well. Therefore the tests done with respect to the store and forward method should be conducted on several nodes. In this way the possible savings throughout a wireless sensor network could be surveyed.

When the improvement grade of the store and forward method is found the adaptive mechanisms could be added. The ideal store and forward condition with respect to energy usage, is probably a condition where the usability of the system

is low. The method will probably store the data packets for as long as possible and thus makes the data unavailable for the user. It's here the adaptive mechanism comes to the rescue. It could then control the balance between energy consumption and system usability. But before the adaptive mechanism is considered, the possibilities of energy savings with the store and forward method must be thoroughly mapped. It's this method that actually conserves energy, the adaptive part only makes the system usable again after the store and forward method is applied.

3.2 Test objectives

The main objectives for the testing are:

- Establish a wireless sensor network platform
- Identify energy consumption sources
- Measure and analyze energy contributions

3.2.1 WSN Platform

To test the hypothesis from section 1.3 a WSN platform must be implemented and established. This platform establish the test bench for testing actual energy consumption in a real world scenario. There are differences in energy consumption between the various available WSN platforms. But at this stage the relative improvements is the most interesting. This means that when all tests is done on the same platform, the choice of platform does not have great significance for the relative result. Chapter 4 will describe further how the selected platform is implemented and built.

3.2.2 Identification of energy consumption sources

Before any tests are conducted the relevant sources of energy consumption must be identified. This is done to determine which tests to conduct in order to answer the hypothesis from section 1.3. Chapter 5 will continue further on this.

3.2.3 Measurement and analysis

The measurements must be able to quantify the energy consumption of one or multiple sensor nodes simultaneously. The reason for measuring multiple nodes is to have the possibility of mapping the energy usage throughout the network when packets are sent through. The energy measurements must be implemented in a such way that the desired tests described in chapter 5 can be carried out. It may also be an advantage to have some kind of automation to the measurement system. Long tests could then be done in an automated fashion without manual interaction during the test. Network usage is also a key factor to measure. Parameters such as packet size and number of packets is relevant to measure. Chapter 6 will describe how the measurement is done in detail.

With respect to analysis, the key objective is to obtain the data from measurement, organize them in a flexible manner, and then the results can be used for further work and of course analysis.

Chapter 4

WSN Platform - SensorTag

This chapter will go through how the SensorTag ecosystem is set up as a WSN platform. Figure 4.1 shows the physical SensorTag. As described in section 2.5 the SensorTag radio support 6LoWPAN. The implementation is based on the SensorTags as 6LoWPAN nodes and one central border router. An illustration of a typical 6LoWPAN topology can be seen in figure 4.2. The topology supports multiple border routers but for simplicity this is not implemented in this set up. Also the number of nodes in this implementation is low, which makes multiple border routers superfluous.

To implement the 6LoWPAN protocol on the SensorTags an already existing solution is used. This solution is based on an operating system called Contiki and the basic software implementation works ”out of the box”.

4.1 Contiki OS

The SensorTags have been implemented with use of Contiki OS. This operating system is targeted networked, memory-constrained systems with low-power operation in mind (Contiki Open Source Project, 2016). The used version of Contiki OS is: 3.0 Release. The implemented version with all source code is available in appendix E. The SensorTag is one of the supported platforms in Contiki. This means low level drivers, board configurations and some examples is already made. Texas Instruments have also made multiple Wiki pages available where central knowledge of running Contiki on SensorTag is shared (Texas Instruments Incorporated, 2016b). These Wiki pages has been helpful during establishment of this platform.

For development of the application code, the example ”cc26xx-web-demo” has been used as a foundation. All the code that is presented in appendix E has been developed from this example. The path to the example code in the contiki repository is: ”examples/cc26xx/cc26xx-web-demo/”. Unnecessary code from the example have been removed and / or been disabled.

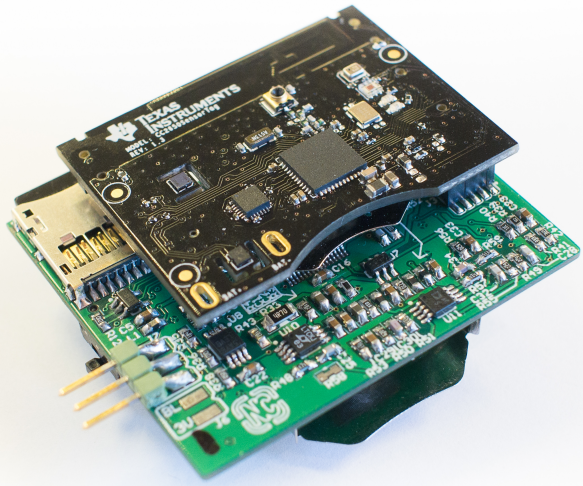


Figure 4.1: SensorTag on the top (black), Measurement system on the bottom (green)

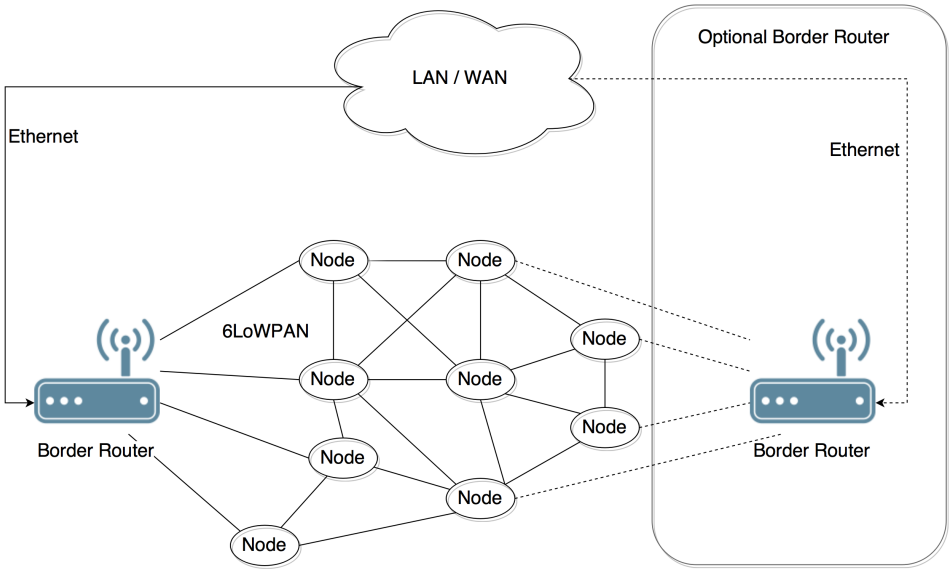


Figure 4.2: Typical 6LoWPAN topology

Figure 4.3: Debugger DevPack connected to SensorTag

4.2 Toolchain & Programming

Contiki OS have a toolchain primarily made of a Make build system with GCC compiler. In addition, several other packages must be installed. Those are mainly platform and architecture dependent packages. Therefore when installing the toolchain, guidance for the target architecture is followed.

The toolchain was first installed from scratch on a Linux Debian 8 system. The toolchain did apparently work, compilation without any errors or warnings. But the compiled binaries had a strange error. Some of the 6LoWPAN traffic dropped completely without error messages, just like a black hole appearing and catches some of the packets. After countless of hours debugging, the fault were narrowed down to some error in the toolchain. Because of time limitation the exact error was not found, but either wrong version of one dependency or wrong GCC version (version 4.9.2 on Debian system, version 4.8.4 on the working Ubuntu system) is likely the reason. To solve this problem a separate system called ”Instant Contiki” was used. This is a virtual machine, preinstalled with Ubuntu and a complete toolchain for Contiki development. With this installation the strange bug disappeared and therefore the rest of the Contiki software development have been done on this platform. The used version is ”Instant Contiki 3.0” (SourceForge, 2015).

Figure 4.4: Border Router made of Beagle Bone Black and CC2531 radio module

To upload a program to the SensorTag the Debugger DevPack have been used (Texas Instruments Incorporated, 2016a). This is a tool with an onboard XDS110 debugger from Texas Instruments. Figure 4.3 show both the SensorTag and the Debugger DevPack. Since the development toolchain is on Linux, a Linux approach was tested first. The Texas Instruments software "CCS Uniflash" was installed. This is an universal flash programming software supposedly supporting Linux and the CC2650 MCU. But this didn't work correctly. Therefore the SmartRF Flash Programmer 2 software was installed. Unfortunately this software is only available on Windows. Therefore a shared folder between the Linux and Windows was set up.

4.3 Border Router

The border router in figure 4.2 is also often called "sink" or "edge router". Simplified this is a device that translates the packets sent in air to cable, similar to what an access point are in the WiFi world. The border router have been implemented with an Beagle Bone Black and a CC2531 USB radio dongle. Figure 4.4 show the hardware.

The Beagle Bone runs Linux Debian 8.3 (Jessie) installed from a customized

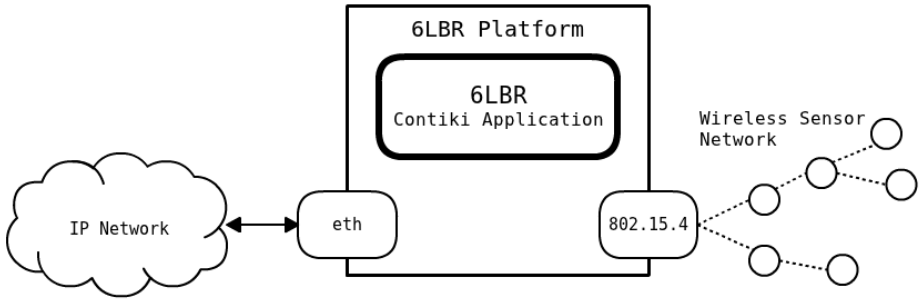


Figure 4.5: Simplified 6LBR diagram. (Source: CETIC asbl (2016b))

image provided by beagleboard.org (BeagleBoard.org Foundation, 2016). The software used to interconnect 6LoWPAN to an IPv6 Ethernet based network is called CETIC 6LBR (CETIC asbl, 2016a). Figure 4.5 illustrates how the 6LBR border router works. The software have been installed by following the installation manual for 6LBR (CETIC asbl, 2016b). The system is configured with a configuration file, and the implemented configuration can be found in appendix F.

A special firmware must be installed on the USB Dongle such that 6LBR could communicate with the radio hardware. This firmware is often called ”slip-radio” which in short makes the radio module available over an serial terminal for the 6LBR software. SLIP is an abbreviation for serial line IP. Texas instruments have already made a customized slip-radio firmware for the CC2531 USB dongle. This binary can be found in appendix F (Digital attachment). The firmware have been programmed with Texas Instruments CC-Debugger and the software SmartRF Flash Programmer on a Windows machine (Texas Instruments Incorporated, 2014).

4.4 IPv6 network & Server

6LoWPAN is an acronym of IPv6 over Low Power Wireless Personal Area Networks. This means IPv6 in the network is essential for the system to operate. Some configurations use a translation mechanism on the Border Router to accept an IPv4 network on the outside. However this is not desired for our test setup. Primarily because it introduce an extra uncertainty on both performance and stability. Therefore an IPv6 network was established in the test LAN where the Border Router is connected.

In terms of an server the nodes can communicate with, a local server is chosen. In practice the test LAN can be connected to Internet through a router and all of the nodes could reach every IPv6-enabled server on the Internet. But this isn't wanted because of the uncertainty in terms of response time, network path variations and other factors that could not be controlled during the test. The implemented server have been connected to the same LAN network as the border router and therefore it is a layer 2 path directly between the server and the border router (i.e. only

switched network, no routers). This ensures an ”as ideal” network condition as possible.

The server is a Linux Debian 8 machine hosted on a virtual development environment. The physical host server have allocated a lot of CPU and RAM capacity to the virtual machine so the Debian server will not be a bottleneck under test. In terms of application the MQTT protocol have been implemented. Therefore the nodes will communicate and exchange data with the server through MQTT. The server is a MQTT broker and runs Mosquitto (Eclipse Mosquitto, 2016). Mosquitto is an open source MQTT message broker. MQTT is also supported by Contiki OS and an example implementation found in the ”cc26xx-web-demo” was used.

Chapter 5

Energy consumption sources

In this chapter the energy consumption sources will be identified together with necessary key parameters for the test program.

A wireless sensor node usually consists of a MCU and a radio as a minimum. Other elements such as sensors is also often found. It is highly interesting with respect to energy consumption to analyze the microprocessor, memory system and radio. Their operation modes will depend on the applied techniques and the potential energy savings must be tested. Other components such as the sensors isn't relevant for testing the hypothesis (sec. 1.3). The reason for this is that the sensors have the same operating mode regardless of the applied techniques for possible energy conservation (i.e for a sensor, same data sampling frequency, sleep-mode and such).

Section 3.1 have already introduced several basic concepts for testing. Initially, the store and forward method must be tested. Several individual sub parameters should be tested to verify if the method have any energy savings potential. The following list summarize the individual parameters to be tested:

- Packet size vs Energy consumption
- Packet frequency vs Energy consumption
- Data storage vs Energy consumption

At first all the individual parameters should be tested on the node generating data. In the next step, tests should be performed on multiple nodes throughout the node tree. The reason for doing a multi node test is to see how the individual parameters also affect the energy consumption on other nodes in the tree.

If the individual tests indicate possibilities for energy savings, the complete store and forward method should be benchmarked. This test must quantify the difference in energy consumption when the ratio between store and forward is changed. The test will probably give a sweet spot for the optimum distribution between storing

and sending packets. The sweet spot is the best case of the store and forward method when we focus on energy optimization.

Chapter 6

Measurement setup

With the established WSN platform from chapter 4, it was necessary to measure certain parameters of the sensor nodes to achieve the test objectives from chapter 3.2. The following two sections will cover how the measurements is set up.

6.1 Energy Measurement

To be able to perform the tests presented in chapter 5, there is need for a system to measure the energy consumption of one or multiple nodes in a WSN. One important thing to notice is that the most interesting energy parameter is the long term energy consumption. Not necessarily the ”instant” power consumption. In a real world scenario, a WSN node often have a battery or an other constrained energy source. Therefore the energy consumption over a long period is important and necessary to collect. When running tests the theorem *Law of Large Numbers* as presented in section 2.4 is used. Each measurement is performed with multiple trials. If testing with enough trials is preformed, the results will be close to the expected real world scenarios.

A short review of the available measurement solutions available was done. This review found out that there exists few systems targeted energy measurement of small wireless nodes. Available systems was either unnecessary large, too bulky or very expensive. When doing a multi node measurement the large systems would require huge physical space and makes them unsuitable.

Therefore, a completely specialized measurement solution have been designed and implemented. This solution is targeted towards multi node measurement in wireless sensor networks. The implemented version is designed for use together with the SensorTag, but only minor interface changes is necessary in order to use it on different nodes. Part III of the thesis will describe the system in detail.

6.2 Network Measurement

In the first tests described in chapter 5 it's only necessary to establish a simple network measurement. In some of the tests, the network measurements is used to verify that the sent data is correct according to the test. In those experiments, packet size and packet count need to be measured. This have been implemented in two ways. The first method is to let the firmware on the node (SensorTag) locally store the number of packet's that is sent. And the packet size is given from the code itself. Despite the simplicity of the method it has one major flaw, it can't be used to verify that the actual data is sent from the node, it can only assume it's been sent. Therefore a second method have been used. This method simply use the software Wireshark on an external computer to monitor the network traffic on the test LAN. Wireshark is a powerful network protocol analyzer that can record all traffic on a network. In this way the traffic sent on the LAN network is measured.

Although the implemented Wireshark set up can measure what happens on the LAN network, this method isn't capable to deliver 100% accurate analysis of what's sent in the air. Therefore a 6LoWPAN packet sniffer was implemented. This packet sniffer consists of a hardware module and associated software. The hardware module must be capable of receiving 6LoWPAN packets (basically support of IEEE 802.15.4).

This method have been tried by using an Texas Instruments CC2531 USB Dongle and a python script converting the data from the radio module to a PCAP formatted file (George Oikonomou, 2016). Wireshark does support this PCAP format and the data sent wirelessly in the air could be monitored in a very similar way as the network LAN monitoring. Unfortunately this method isn't currently used due to some compilation error and time constraints. The firmware running on the CC2531 must be a special Contiki example called "Sniffer" ("examples/cc2538dk/sniffer"). Due to some bug, the example code doesn't work properly and fixing the packet sniffer haven't been prioritized since the Wireshark LAN measurement is sufficient for the current testing.

Part III

Custom Energy Measurement System

Chapter 7

Measurement solution

To be able to fulfil the desired tasks described in chapter 6 a custom energy measurement system specifically designed for this application have been implemented and later used as a tool in part IV. The current part (III) of the thesis will describe this measurement system. The system have been implemented with a goal to easily identify and measure the energy consumption of multiple nodes concurrently in a wireless sensor network.

First a set of requirements have been established before the actual implementation is presented in chapter 8. Thereafter, the hardware and PCB layout is given in chapter 9 before the assembly process is covered in chapter 10. The central elements of the firmware implementation is given in chapter 11 and finally the analyzer software is described in chapter 12.

Hereafter the custom energy measurement system is referred to as CMS or measurement system.

7.1 Requirements

When developing an embedded system it is important to map the requirements for the desired system. This phase lay an important foundation for further development and implementation.

The main goal with the CMS is to measure the energy consumption under different circumstances of one or more nodes in a wireless sensor network. When multi node measurement is desired it's accepted that it's one CMS per node as long all the systems can cooperate in a such way that the data could be used and compared in analyzation.

A list of minimum requirements have been made:

- Measure both voltage and current of device under test
  - Required precision specified in detail in section 7.1.1 & 7.1.2

- Record continuously for up to 24 Hours
- Relative time stamping of the recorded data
- Operate on battery power
- Is a daughter board for TI SensorTag
- Physical size comparable to TI SensorTag

In addition to the minimum requirements, a set of optional requirements have been suggested. This list consists of multiple ”nice to have” features, but none of them are required for a minimal working CMS:

- Transfer result data wireless back to a central server
- If an analogfrontend is used, automatically calibration of this
- Serial communication with DUT for automation purposes
- Absolute time stamping of the recorded data
- Automatically transfer metadata from DUT to CMS
- Time and operational state synchronization wirelessly between multiple CMS

7.1.1 Needed current precision

The target device for the CMS is the SensorTag with a CC2650 SimpleLink wireless MCU. From CC2650's datasheet the standby power consumption is 1 μA and the maximum is 9.1 $mA + 0.818\ mA = 9.918\ mA$ ($I_{core} + I_{peripherals}$) (Texas Instruments Incorporated, 2015a, p. 14). But the SensorTag includes other devices such as sensors in addition to the MCU. To verify the total consumption, a simple test have been carried out as viewed in figure 7.1. The test used a μCurrent Gold together with both a Agilent U1273A and a Saleae Logic Pro 16 (analog channel used). The maximum current level was 25 mA peak.

Due to this, the needed current range is from minimum 1 μA to $25mA$ with a resolution of at least $1\mu A$. The resolution doesn't necessarily need to be 1 μA over the full range, but it's important that there is sufficient resolution below 100 μA to avoid large inaccuracies.

7.1.2 Needed time precision

To measure the power consumption of a device we need to sample the voltage and current at a high enough speed to capture the peaks and sudden changes. If this isn't done, the result could view a false truth by not capturing the higher frequency components of the power consumption. However, if the system don't need to identify the instantaneous values, but rather measure the average consumption over a period of time, a low pass filter could be added and thus reduce the requirement for high sampling speed.

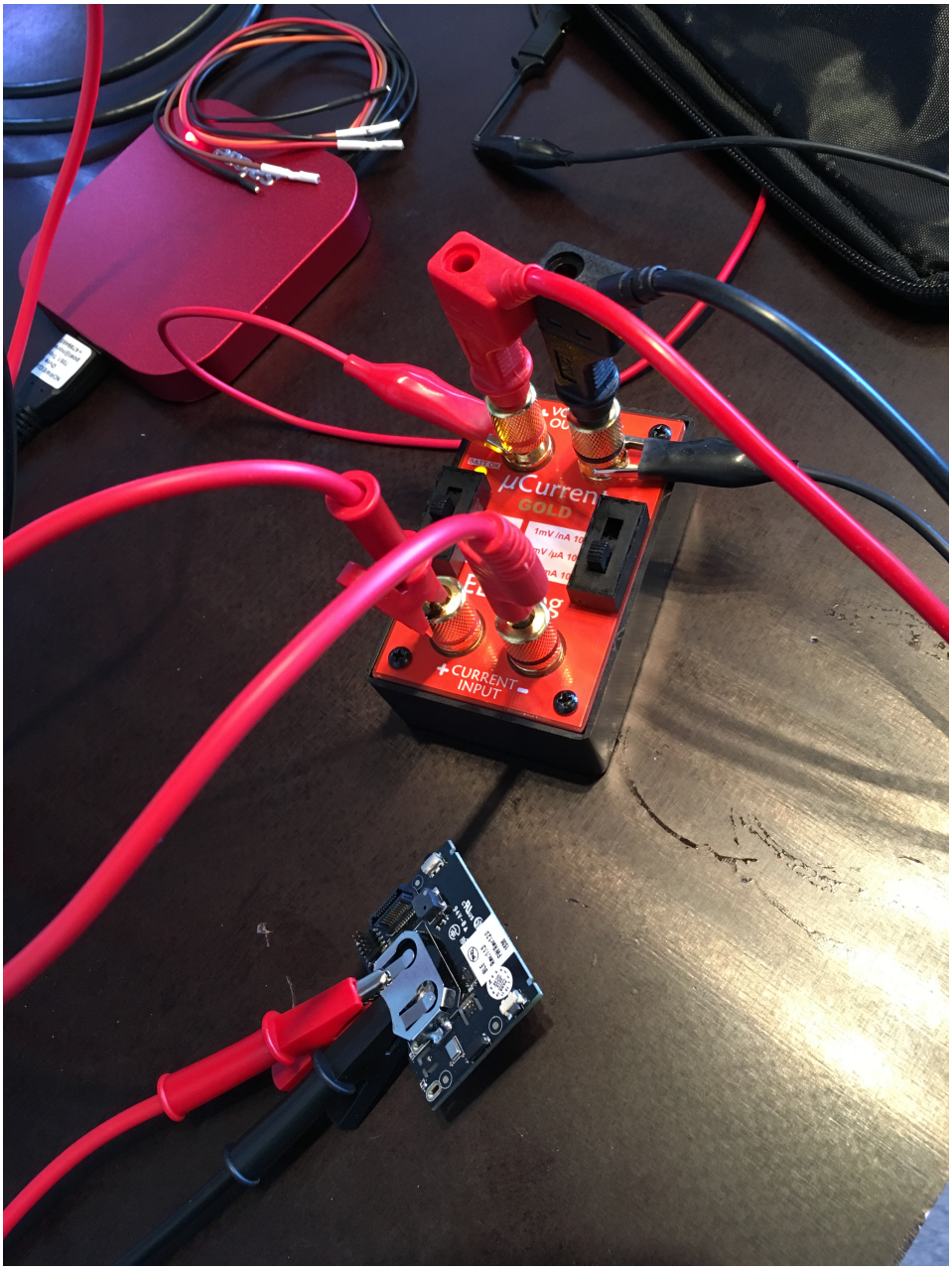


Figure 7.1: Measuring the maximum current drawn by the SensorTag

As mentioned in section 6.1 the primary need is to measure the energy consumption over a longer period of time. To concretize this, it's more interesting to measure the consumption at a per second basis rather than per microsecond. The measurement must of course be an accurate average of the energy used in the time period. The appnote (Lindh and Lee, 2015) from Texas Instruments goes through a measurement system setup with an Texas Instruments CC2640 MCU. This MCU is a superset device of the CC2650 and therefore this appnote is highly relevant. In the appnote they use an Agilent N6705B DC Power Analyzer with an N6781A internal module. The system have capabilities of measuring with a sample speed of 200 kHz. And with that sampling speed, Texas Instruments shows that they can capture rapid changes such as fast recharge events done to retain the internal RAM. This sampling speed is very high compared to the needs when measuring over longer time. Regardless this appnote shows that to be able to measure and differentiate the consumption at a fine level approximately 200 kHz sampling speed is needed.

Even if most of the CMS measurements could be done with "a second" resolution, there could be certain scenarios it could be interesting to have higher resolution. Therefore as a trade off, a sampling speed between 1 kHz and 10 kHz is suitable for the measurement system. With this speed it is still possible to measure certain system events such as a radio wake up, interrupts and similar, but not capture the fastest events. To ensure correct measurement this must be combined with an appropriate low pass filter according to the selected sampling speed.

7.2 Data logging capabilities

In an acquisition system the needed performance of the storage and processing capabilities are important to identify as well as the storage requirement.

Since this system is meant to be used in a research context, all of the data processing and analysis could be done in retrospect. Therefore ideally the raw content from the ADC is stored directly to minimize the on board computing.

To give this some numbers let's take an example of a 16-bit ADC with a sampling frequency of 5 kHz. Then the absolute minimum storage speed needed is 10 kByte/s (without overhead) for one channel. Table 7.1 summarizes the required storage space for different lengths. As we can see at 1 hour, the storage requirement is 36 megabytes, while for 24 hours of logging, a whopping 864 megabytes is needed.

For a small embedded system, this storage capacity requirement is quite demanding. Especially when multiple ADC channels are added to the system. Therefore a technique to reduce the storage requirement is needed. There exists numerous different compression techniques solving this. One of the most simple techniques have been chosen for implementation in this measurement system. The method is an X-axis (time) averaging algorithm where we offer resolution on the X-axis for reduced storage amount. This technique is a lossy compression method where we reduces the storage requirement by removing information on the time axis.

An example implementation is presented in figure 7.2. Briefly explained, the

Logging length (seconds):	Storage requirement (kBytes):
0.5	5
1	10
10	100
60	600
600	6000
1800	18000
3600	36000
86400	864000

Table 7.1: Example of storage space needed

```
246     sumResult.numberOfSamples += 1;
247     sumResult.adcSumVoltage += currentRead.adcRawVoltage;
248     sumResult.adcSumCurrentR1 += currentRead.adcRawCurrentR1;
249     sumResult.adcSumCurrentR2 += currentRead.adcRawCurrentR2;
250
251     if(sumResult.numberOfSamples >= ACQUISITION_CONTAINER_SIZE){
252         cb_push_back(&adcSumQueue, &sumResult);
253
254         sumResult.numberOfSamples = 0;
255         sumResult.adcSumVoltage = 0;
256         sumResult.adcSumCurrentR1 = 0;
257         sumResult.adcSumCurrentR2 = 0;
258     }
259
```

Figure 7.2: Lossy compression with X-axis averaging algorithm

Logging length (seconds):	Storage requirement with N samples average (kBytes):			
	100 samples	250 samples	500 samples	1000 samples
0.5	0.15	0.06	0.03	0.015
1	0.3	0.12	0.06	0.03
10	3	1.2	0.6	0.3
60	18	7.2	3.6	1.8
600	180	72	36	18
1800	540	216	108	54
3600	1080	432	216	108
86400	25920	10368	5184	2592

Table 7.2: Example of storage space needed with simple average algorithm

algorithm does sum multiple samples on the X-axis and later divides the combined sum (sumResult in figure 7.2) on N samples. This will give the average of all the summed samples. Since this will be implemented in an embedded system, floating point operations could have a big impact on processing speed. Therefore the division is done later and thus is not a part of the implementation viewed in figure 7.2.

$$\frac{ADCSamplingfrequency}{NSamplesAverage} * NBytesstore * Time \tag{7.1}$$

If we use the same example from start of chapter 7.2, the new storage requirements could be viewed in table 7.2. Four different sizes of compressions is viewed in the table, equation 7.1 is used to calculate table 7.2. Since the division is done externally we must store a counter of how many samples a sum contains. Therefore an extra two byte value need to be stored. The total number of bytes needed to store for each summation is: *Sum Result (4 bytes) + Number of samples (2 bytes).*

According to table 7.2, there is a lot of space to save. To illustrate this, figure 7.3 combine the data from table 7.1 and 7.2 in one graph. To view the difference between the possible averaging sample sizes, figure 7.4 present a zoomed version of figure 7.3.

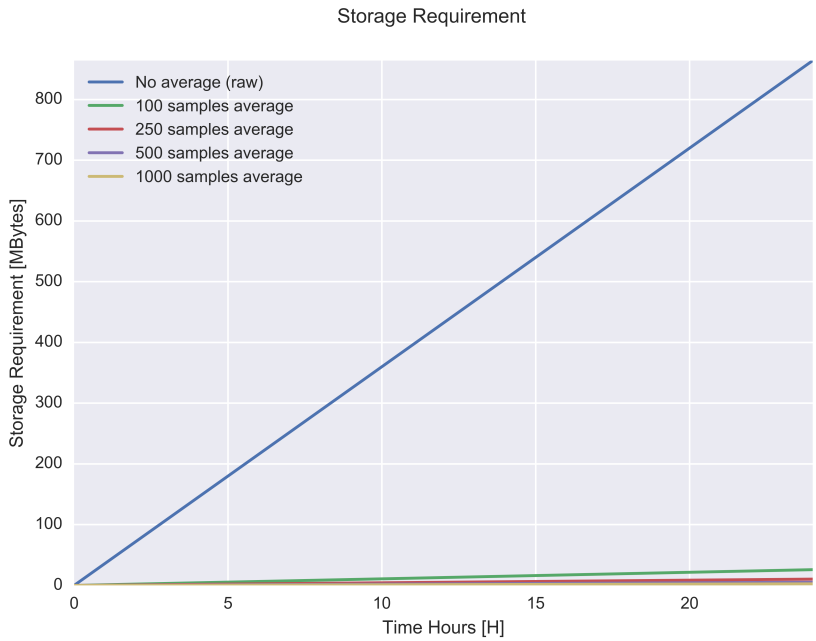


Figure 7.3: Storage requirement with and without the averaging algorithm

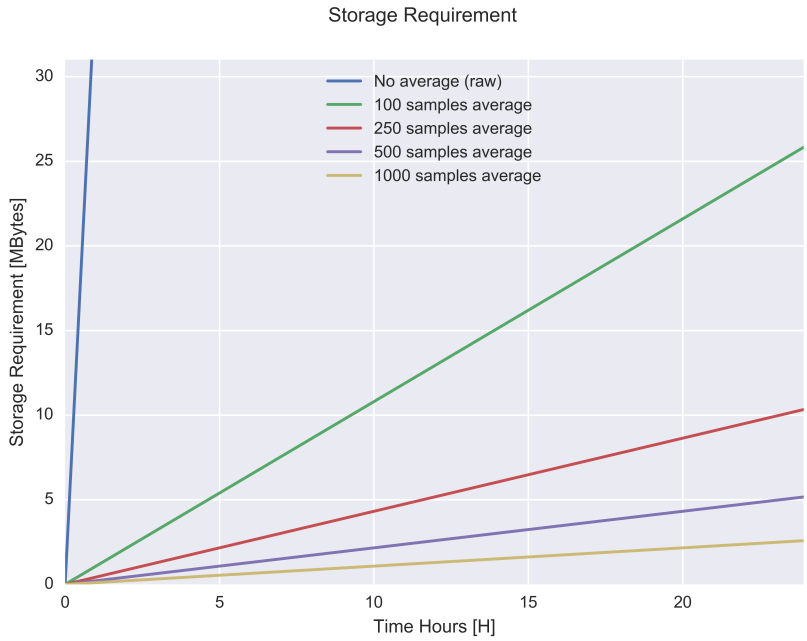


Figure 7.4: Zoomed version of figure 7.3

Chapter 8

Implementation

In this chapter you will find a detailed description of every sub module in the developed measurement system. Each subsection will describe the rationale behind all the major parts of the system. Description of more common elements such as decoupling capacitors and pull-up resistors will be reduced as this is not special for this application.

8.1 Microcontroller

Selection of the microcontroller have a large impact on the capabilities of the measurement system. In mass-produced products the main importance is to use a cheapest possible microcontroller that solves the task. Why include and pay for a DMA and MMU when the application never use it? This is also one of the reasons there exists a lot of different varieties in the microcontroller market. Each optimized for different market areas. However, in this application we are not so interested in lowering the unit cost. It is much more important to select a microcontroller that have all of the functionality needed and a little extra for further development. This without selecting a excessively large and complex microcontroller.

8.1.1 Microcontroller selection

The first major selection is microcontroller architecture. The 32-bit ARM Cortex M3 / M4 families was selected as a start. The main reason behind this choice are features such as more peripheral modules, higher performance and a DMA module compared to the more resource constrained 8-bit architectures. As mentioned in chapter 7.1.2 we need a moderate ADC sampling frequency with consecutive processing before saving the results to non-volatile memory. Therefore the added performance is preferred. The author does also have earlier experience with different Cortex M3 / M4 controllers which is convenient.

Many of the large companies such as Analog Devices, Atmel, NXP, Silicon Labs, STMicroelectronis and Texas Instruments have each families with different varieties of Cortex processors. To decide which controller to use, several key properties was considered:

- Processing power
- Memory capabilities
- Energy saving features
- Support from manufacturer
- Toolchain
- Existence of hardware modules:
  - ADC module
  - Timer counter modules
  - DMA
  - Serial communication (USART, SPI, $\boldsymbol{I^2C}$)

After a short review of different Cortex M3 / M4 possibilities, the EFM32 family from Silicon labs was the prefered one. The family have a lot of energy saving features, and have at the same time performance comparable to similar processors without the energy saving features. Since the CMS is desired to have the possibility of battery operation this seemed to be a perfect fit. Therefore a Cortex M3 variant of the EFM32 family was selected. More specifically EFM32 Leopard Gecko.

This model have several relevant features: 12-bit ADC, DMA, PRS (Peripheral Reflex System), USART, I^2C and several timer counter modules.

In the process of iplementation of the MCU, several appnotes from Silicon Labs has been used followed: AN0002, AN0004, AN0016, AN0026, AN0062.

8.1.2 Debug connection

To be able to program and debug the MCU, two interfaces have been added. Serial Wire Debug (SWD) is a 2 wire debug interface with the same protocol as JTAG and is found in most controllers featuring a ARM Cortex-M core. For extended trace possibility, SWO have also been added with the cost of one extra wire (ARM Ltd., 2009, ch. 11). Figure 8.1 show how the SWD/SWO connector is implemented. This pinout follows the standard (ARM Ltd., 2016).

As a second interface UART have been added. This allows for a more simplistic debug output to a terminal viewer on an external target while running the application. Typical usage scenario could be seen in figure 8.2 where user readable debug prints is outputted to the terminal. As an extra feature, the EFM32 comes with an pre-installed UART bootloader. Therefore the selection of which UART module is

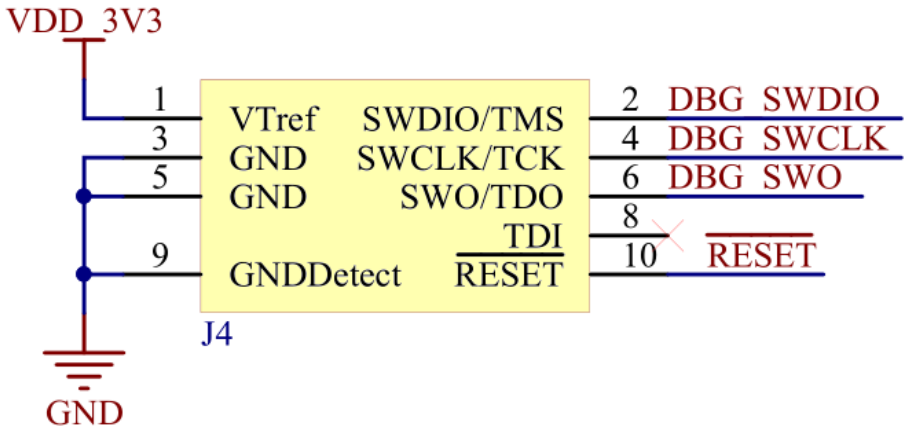


Figure 8.1: SWD/SWO connector

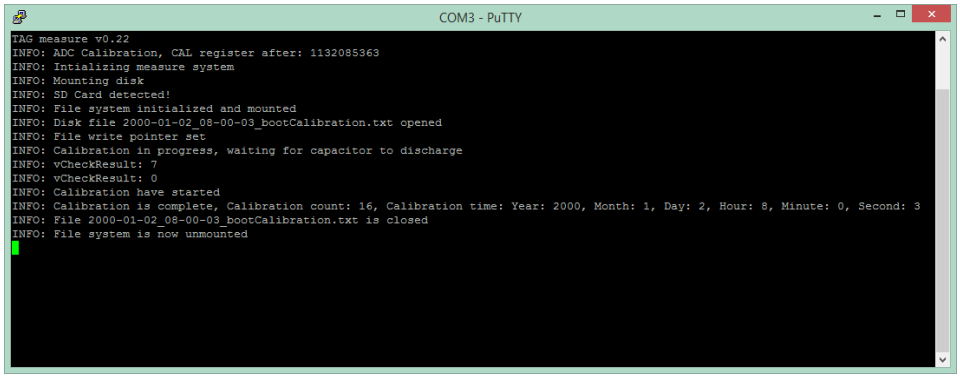


Figure 8.2: Example of debugging with external terminal viewer

restricted to "US0_LOC0" (Silicon Labs, 2014). This combination is also known as "BOOT_TX / BOOT_RX" in the datasheet and in the pin matrix. To be able to enter the bootloader the pin "PF0" must be held high at boot. Therefore the two pads called "Bootloader pads" is added to the CMS.

8.2 Analog to Digital converter

The CMS must have an ADC module which converts the measured analog parameters and digitalize these. Since our chosen MCU have a internal 12-bit ADC, the possibility of using this is naturally considered. The specification of the ADC tells us that it is a successive approximation (SAR) ADC, with a native resolution of 12-bit and up to 1 MS/s (Silicon Labs, 2014).

As described in subchapter 7.1.1 the current measurement range should go from 1 uA up to minimum 25 mA. According to formula 8.1 it is possible to check what the smallest increment the digital result can represent with the given range. As we can see with our 12-bit MCU ADC and a range up to 25 mA, each step in the ADC will represent 6.1 uA. Bear in mind this is without considering noise, nonlinearity, temperature drift and so on. When we want a resolution down to at least 1 uA we could see that the solution is not sufficient.

$$\frac{Range}{Resolution} = \frac{25 * 10^{-3} A}{2^{12} Bits} = 6.1 uA/Bits \tag{8.1}$$

Now there is two main approaches, use an external ADC with higher resolution or try to improve the use of the internal ADC. From chapter 7.1 there is a requirement in terms of physical size. The use of an external ADC will therefore be more space demanding and thus this solution is undesirable.

The ADC implementation have been based on dividing the total current measurement range into two separate ranges. One lower range and one higher range where two separate ADC inputs is used. In this way, 12-bit resolution is still high enough without loosing the needed precision on the lower end of the total range. The implementation of this analog front-end and range separation is in depth explained in section 8.4 below.

As chapter 7.1.2 specifies, the requirement for sampling speed is up to 10 kHz on all channels. The internal ADC is capable of 1 million samples per second (after internal MUX). This implementation is using three channels and without considering overhead, the sampling speed will be $\frac{1\ MS/s}{3} = 333.33\ kS/s$. This resulting sampling speed is in order of magnitudes higher than the requirement, meaning the sampling speed of the internal ADC is sufficient. As a consequence of the large difference, the resulting overhead is not calculated any further.

The ADC implementation have considered several important elements for stable and accurate operation. When developing a mixed system with both analog and digital parts isolation of the supply is important. This is done to prevent supply noise to propagate from digital parts of the system into the analog parts. Figure 8.3 shows how the AVDD for the internal ADC is filtered from the regular VDD.

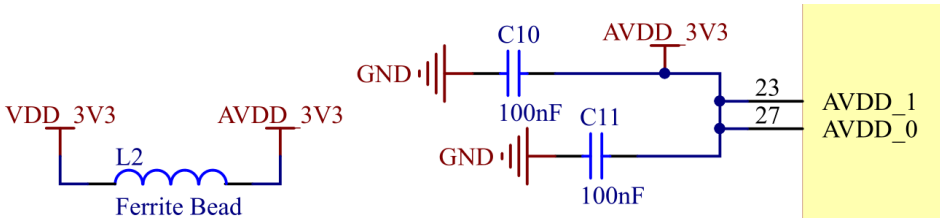


Figure 8.3: Supply filtering for the ADC

This is according to the principles from the application note "AN0002" (Silicon Labs, 2016, ch. 1).

8.2.1 Voltage Reference

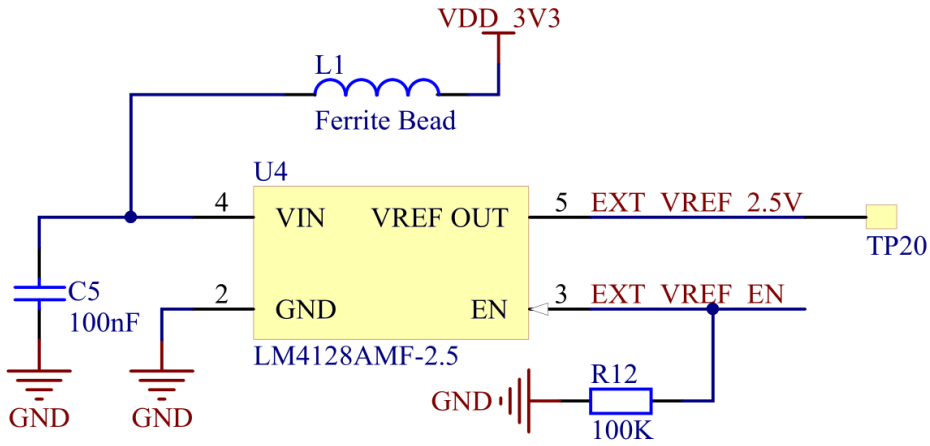


Figure 8.4: Implementation of external voltage reference for the ADC

To further improve the performance and stability of the ADC an external voltage reference have been implemented. The implemented voltage reference is a LM4128 precision voltage reference from Texas Instruments (Texas Instruments Incorporated, 2013). It is a 2.5 V reference, and the implementation is shown in figure 8.4.

8.3 Voltage Measurement

To measure the voltage a voltage divider was implemented. Figure 8.5 shows the actual implementation. The voltage divider is necessary because the ADC voltage reference is 2.5 V and the voltage measurement must at least be able to measure 3.3 V. To have some headroom over 3.3 V the divider is selected with a output voltage

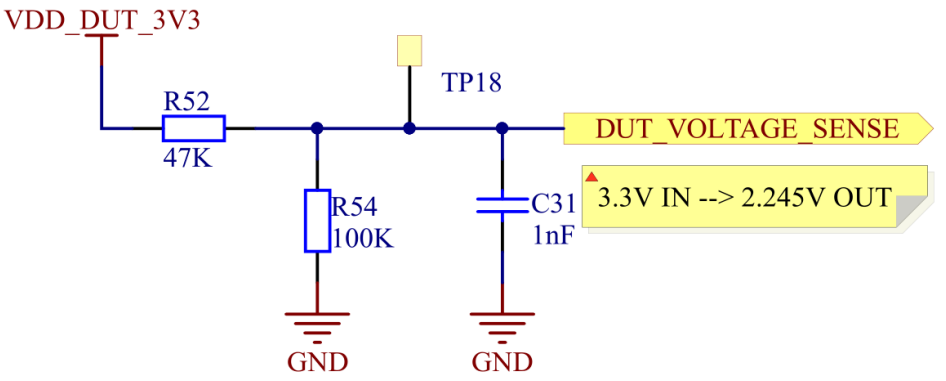


Figure 8.5: Voltage divider for voltage measurement

slightly below the maximum of 2.5 V. Equation 8.2 shows what the minimum step size we could measure at the ADC input. In equation 8.3 this step size is translated to the resolution before the voltage divider by multiplying the result from equation 8.2 with the gain of the voltage divider.

$$\frac{Range}{Resolution} = \frac{2.5V}{2^{12}Bits} = 0.61mV/Bits \tag{8.2}$$

$$0.61mV/Bits * \frac{47k\Omega + 100k\Omega}{100k\Omega} = 0.8967mV/Bits \tag{8.3}$$

The result from equation 8.3 shows that we could measure the voltage with a resolution of 0.9 mV. This have been considered as sufficient for this system.

The reason for C31 in figure 8.5 is to filter out high frequency components from the measured signal. Equation 8.4 shows the formula for calculating the cutoff frequency for an first order low-pass filter (Horowitz and Hill, 2008, ch. 1.19). The RC filter value have been calculated from equation 8.5.

$$f(-3dB) = \frac{1}{2 * \pi * R * C} \tag{8.4}$$

$$f(-3dB) = \frac{1}{2 * \pi * 100K\Omega * 1nF} = 1591.55Hz \tag{8.5}$$

8.4 Current Measurement

To measure the current in a circuit there exist several solutions. After a short study, a solution based on a current shunt was selected. This method work by translating the current to a proportional voltage with the use of a shunt resistor. It is crucial that the shut resistor have adequate precision and tolerance. Under the study, similar systems was checked. The circuit implemented here is inspired

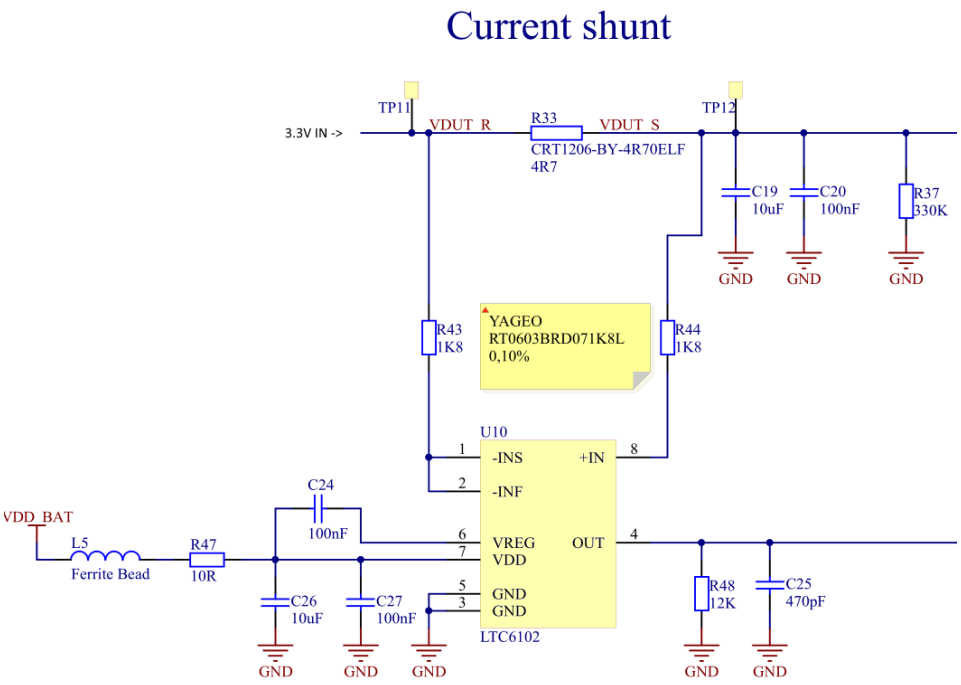


Figure 8.6: Implemented current shunt and amplifier schematics

by (Silicon Labs, 2013, ch. 11) which is a circuit from a EFM32 development board that has a built in current measurement capabilities.

To amplify the voltage over the shunt resistor, a LTC6102 precision zero drift current sense amplifier has been selected (Linear Technology, 2014). This amplifier will translate the voltage difference over the shunt resistor to a proportional ground-referred current. Figure 8.6 shows the implementation.

Some of the key component values and formulas is summarized in the formulas 8.6, 8.7, 8.8, table 8.1 and figure 8.6.

$$Gain = \frac{R_{out}}{R_{in}} \tag{8.6}$$

$$V_{out} = V_{sense} * Gain \tag{8.7}$$

$$Gain = \frac{12K\Omega}{1.8K\Omega} = \frac{20}{3} = 6.6667 \tag{8.8}$$

To ensure that noise isn't propagating further into the circuit an output RC-Low pass filter was added. This filter is formed by R48 and C25 in figure 8.6. The calculations for the -3dB point is presented in equation 8.9.

Sense Current	V_{sense}	V_{out} (From LTC amp)
33 mA	155.1 mV	1.034 V
3.30 mA	15.51 mV	103.4 mV
320 uA	1.504 mV	10.026 mV
220 uA	1.034 mV	6.8933 mV
100 uA	0.47 mV	3.1333 mV
3.30 uA	15.51 uV	0.1034 mV

Table 8.1: Output voltage from the LTC6102 with relevant sense currents ($Gain = \frac{20}{3}$)

$$f(-3dB) = \frac{1}{2 * \pi * 12\Omega * 470pF} = 28219Hz \tag{8.9}$$

Theoretically the output from the LTC6102 amplifier could be connected to an ADC input. As described in section 8.2 the current range must be divided into two separate ADC inputs in order to achieve the wanted precision over the desired current range. Therefore a second set of signal conditioning was implemented. This conditioning consists of two CMOS operational amplifiers. The AD8656 from Analog Devices was selected. This is a low noise precision CMOS operational amplifier with two separate amplifiers in one package. The implementation is shown in figure 8.7.

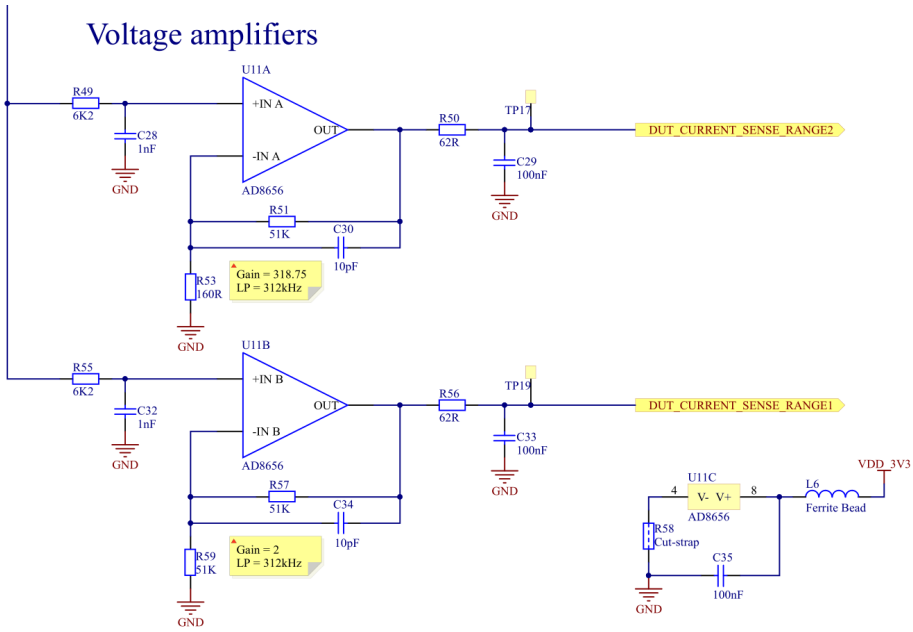


Figure 8.7: Voltage amplifiers implemented in the current measurement circuit

To calibrate the system before usage a calibration network was added. This

network could apply known currents through the shunt resistor and thus the read values from the ADC could be calibrated against this. Figure 8.8 shows the implementation. The network consists of a set of precision resistors together with two analog switches. The U7 analog switch in figure 8.8 selects whether the sensing voltage rail should be connected to the DUT or the calibration network. This means that the DUT will loose power when the calibration network is selected.

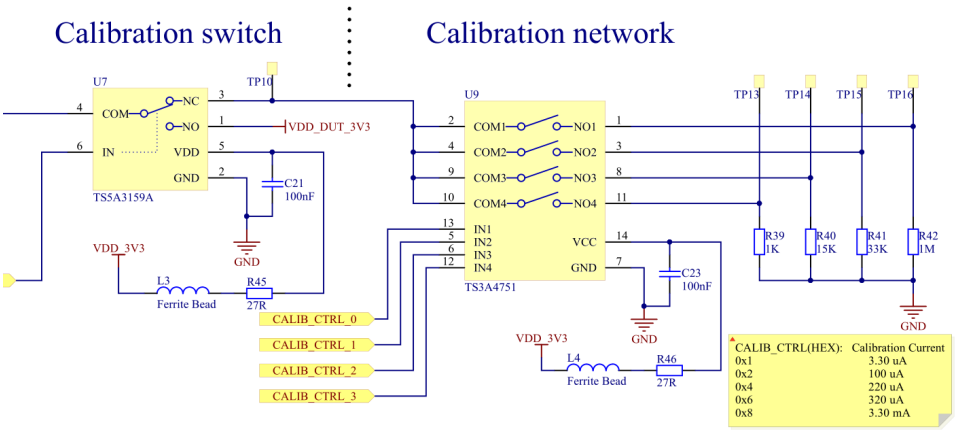


Figure 8.8: Implementation of the calibration network in the CMS

8.5 Storage

When implementing the storage solution, several different methods was considered. The methods considered was internal flash, external flash IC and external flash SD card. From section 8.2 it's now clear that the implemented ADC have a resolution of 12-bit and the system need to use three channels with a desired sampling speed of 5 kHz. The total required storage speed is then 30 kBytes/s when storing the raw content. If a compression technique is applied this will decrease further, so 30 kBytes/s is the maximum without overhead taken into account. None of the three storage methods should have problems with this storage speed and therefore all is still plausible.

When it comes to storage space there is more difference between the considered methods. With the 24 hour recording requirement from chapter 7.1 combined with the highlights from chapter 7.2, require a total of 2.6 gigabyte storage space. This is without any compression, just storage of the raw result for 24 hours. Immediately we can exclude use of the internal flash. The large difference between common internal flash sizes (somewhere between 128 to 4096 kilobytes) and the requirement, even with compression, makes this method impossible to implement. When it comes to the external flash IC some comes with SPI or I^2C interface, but none with the required capacity were found. All of the found external flash chip's with gigabyte capacity have parallel interface and often BGA package. This makes them

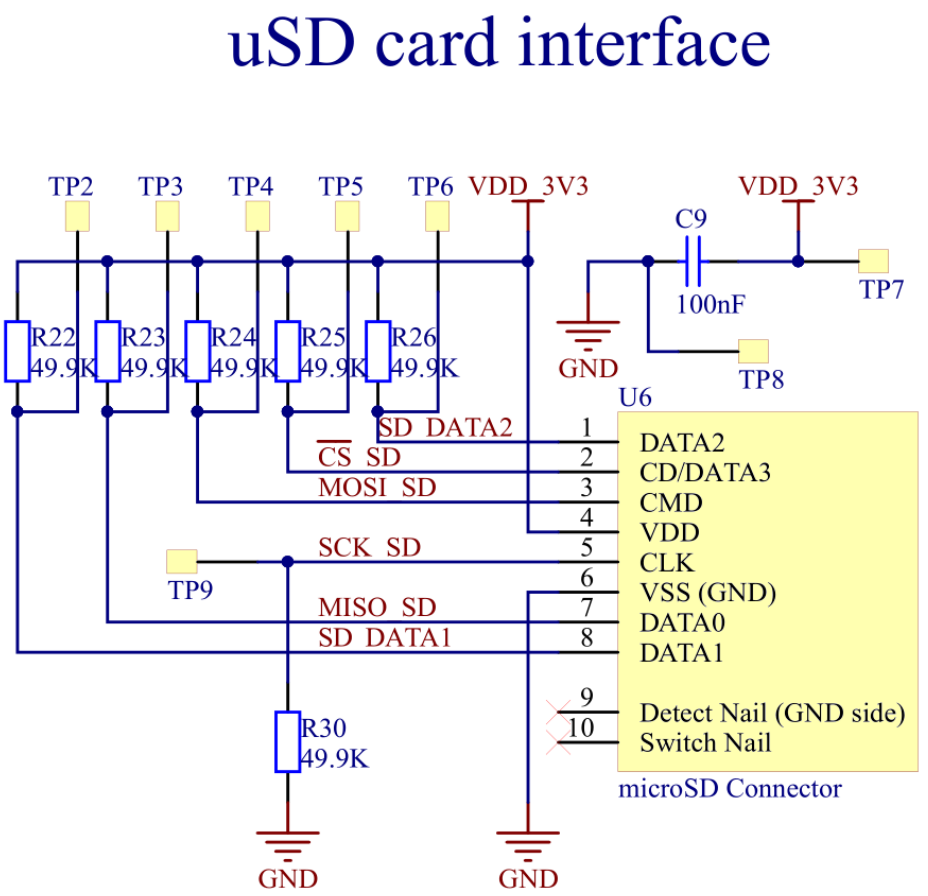


Figure 8.9: Micro SD card implementation

complex to implement, especially when the increased storage speed with a parallel interface is not required. The parallel interface also requires a lot of I/O pins which the selected MCU don't have. Therefore we only have external flash SD card left as a relevant solution.

The implemented solution is based on a micro SD card operated in SPI mode. Micro SD cards comes in different storage sizes. Cards with 4 to 8 gigabyte is very common and easily available. Therefore the requirement from chapter 7.1 is fulfilled with this implementation, even when the raw result from the ADC is stored. Figure 8.9 shows the implementation. Several test pads have been added to enable the possibility of easily debugging the SD card interface under operation. The interface have been implemented in accordance with the specification (NXP Semiconductors, 2013).

8.6 Real Time Clock unit

In a measurement application where data is stored it could be important to add absolute time information. In scenarios with multiple measurement systems being used at the same time on different nodes, it could be especially important to be able to synchronize the recorded data. Therefore a high precision RTC module have been added to the system. The chosen clock module is DS3231 from Maxim Integrated. It uses an I^2C interface, have an internal precision crystal and could automatically switch to a backup power supply to keep time when the rest of the system is shut down (Maxim Integrated, 2015).

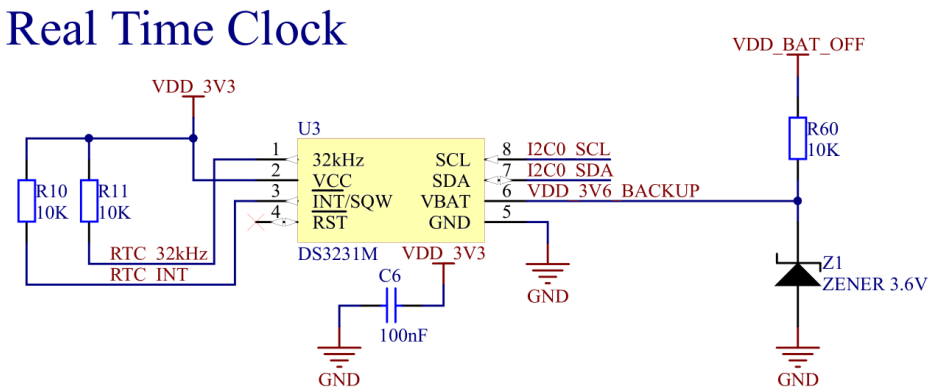


Figure 8.10: RTC implementation

Figure 8.10 shows the implemented layout. When using I^2C it's always important to be aware of the addresses to all slave devices and avoid collisions. DS3231 have a pre-programmed address, but in this implementation the DS3231 were connected to it's own I^2C bus with no other slaves on the bus. Pin 1 and 3 is also connected to the MCU. Pin 1 is outputting the same 32.768 kHz clock as the internal precision clock and therefore subsystems within the MCU can also benefit the precision from the DS3231 module. Pin 3 is outputting a signal when the DS3231 want to interrupt the MCU. For example a predefined alarm could be set and the module will interrupt when the alarm condition occurs. Both signals is active low with a open-drain output which means that external pull-up resistors is required (R10 and R11 in figure 8.10).

8.6.1 Backup Battery connection

The backup power connection on DS3231 is named "VBAT". When the power is lost on the primary "VCC" input the module automatically switches to VBAT for further operation. Normally the VBAT is connected to it's own battery for timekeeping after system power is lost. But in this application we already have a battery as our primary source. This is further described later in section 8.7. Unfortunately, our battery voltage is above the absolute maximum voltage of 5.5

V (Maxim Integrated, 2015, p. 2). Therefore a Zener diode have been used as a voltage regulator as seen in figure 8.10. The Zener diode approach is not a very efficient voltage regulator. Regardless, in this case simplicity have been prioritized as this is not a major feature of the system. In worst case, if the power losses in the regulator is too high, R60 and Z1 could be de-soldered to disable the functionality completely.

8.7 Voltage Regulators and Battery

The measurement system have preliminary two main voltage rails. One 3.3 V rail for the measurement card itself and one 3.3 V rail for the device under test (DUT). Both rails need a voltage regulator and linear regulators are chosen. The two main reasons is lower complexity and less noise on the outputs. Noise on the voltage rails could possibly have a rather large impact on the sensitive parts of the measurement system, this could lead to more inaccurate measurements which is not wanted. One big disadvantage with this choice is the efficiency of the solution compared to a switching power supply.

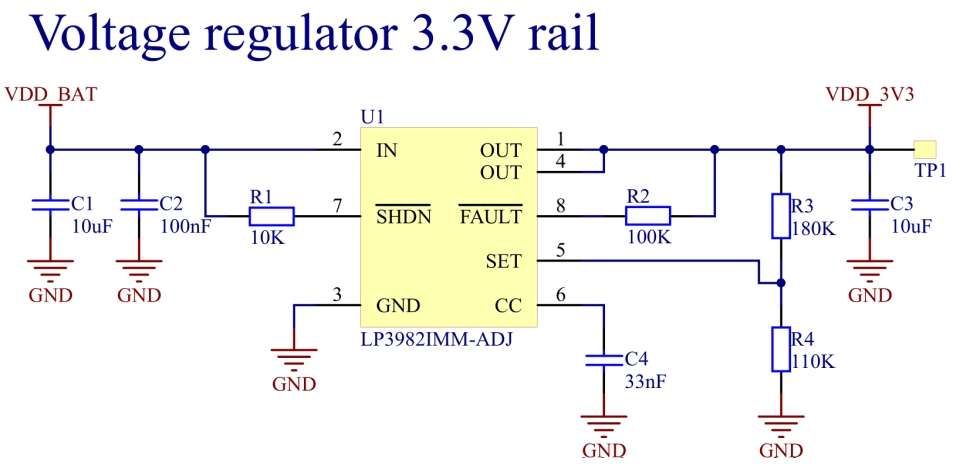


Figure 8.11: Voltage regulator connection

To minimize the power losses an ultra-low-dropuot linear regulator from Texas Instruments have been chosen (Texas Instruments Incorporated, 2015b). The implementation of the LP3982 regulator could be seen in figure 8.11. A similar implementation is also done for the DUT voltage rail.

To power the system, a battery solution with dual CR2032 batteries was implemented. The batteries are wired in series which gives a rated voltage of 6.0 volts. The dual battery solution allows both voltage regulators to have enough ”voltage room” for regulation. This ensures that both voltage rails could maintain a stable 3.3 volt even under heavy load and throughout the battery life time.

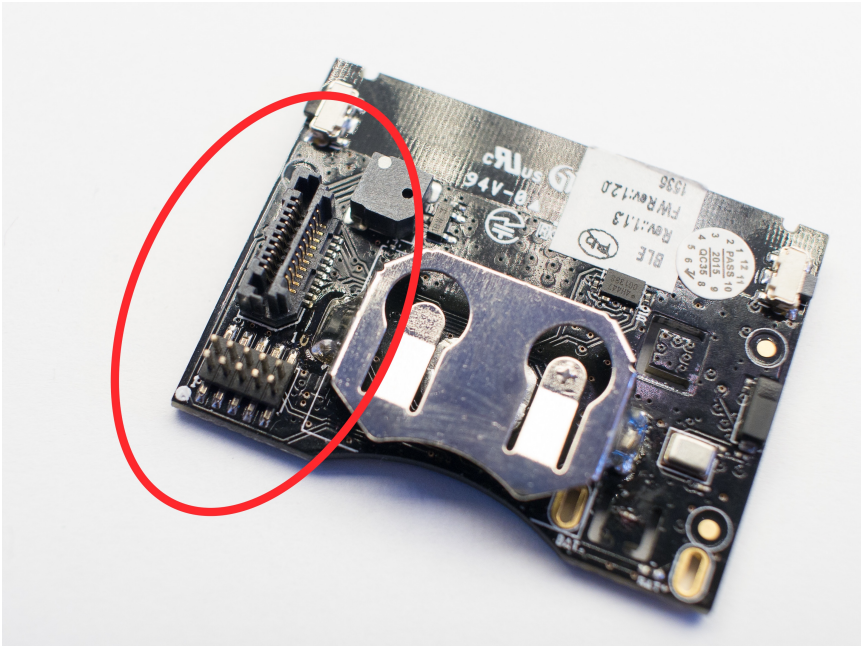


Figure 8.12: DevPack interface connectors on the SensorTag

8.8 DevPack connections

The implemented version of CMS are as described in chapter 7.1 targeted as a measurement platform specially made for the SensorTag. Texas Instruments have already made an interface on the SensorTag for daughterboard connection, often called ”DevPacks”. Therefore the CMS have implemented this interface. The interface consists of two connectors. One Samtec LSS-110 connector and one 10 pin 50mil dual row header. The two connectors can be seen in figure 8.12. The complete interface pinout is documented by Texas Instruments in (Texas Instruments Incorporated, 2015c, ch. 7).

The DevPack interface support stacking of multiple boards. Therefore the CMS have both an input and output interface. As a consequence of this, almost all signals are passed through to the next board. To be able to automate the test procedures several I/O pins in the interface have been connected to the CMS. The three I/O lines called ”DP0, DP1 and DP2” have all been dedicated to the CMS. UART, SPI and I^2C is also connected, but these are not dedicated for the CMS. Therefore they are connected through resistors so that the CMS user could select which serial interface to use by soldering suitable resistors on the desired pins. In the implemented version used in this research work, only UART RX have been used and R32 have been fitted with a value of 49.9 $k\Omega$.

To be able to measure the energy usage of the SensorTag the CMS is required to feed power to the SensorTag through this interface. Therefore, it is a requirement

that the CMS is the first board connected in the stack, otherwise the CMS may measure the current consumption of multiple boards.

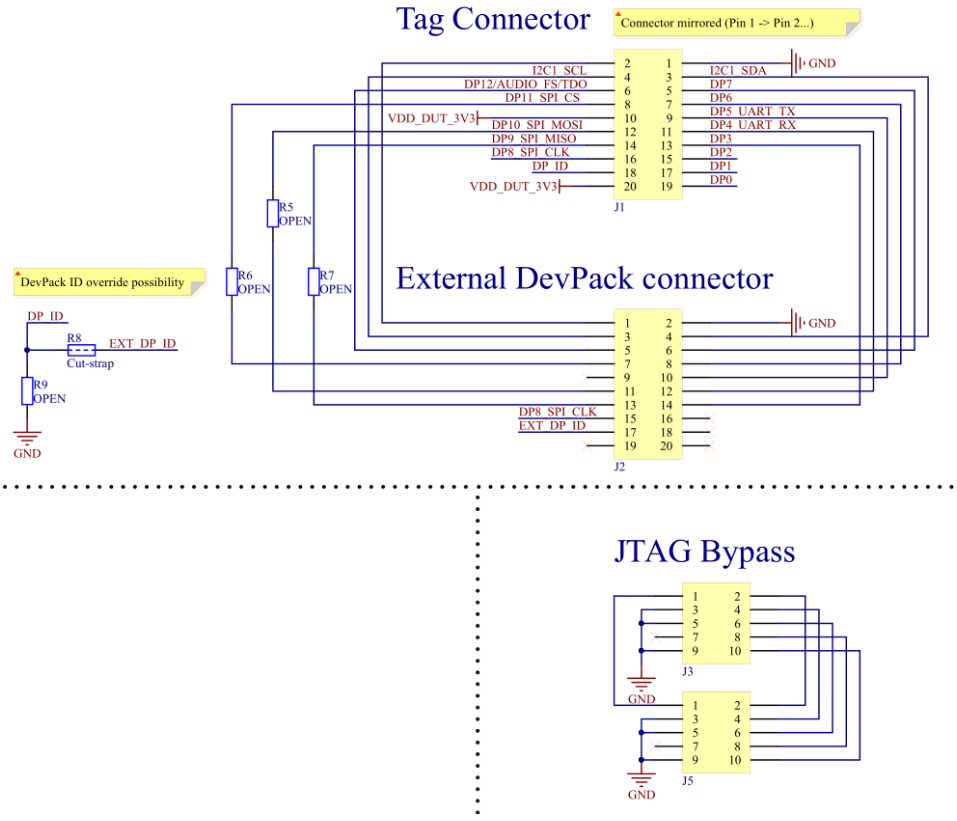


Figure 8.13: Connection of the external DevPack interface

Figure 8.13 shows the actual implementation of the interface. One important notice is that the Samtec LSS connectors (J1 & J2) are intended to be mirrored in the circuit (Samtec, Inc., 2015). Therefore, when you mate two connectors pin 1 on the input equals pin 2 on the output, pin 3 to pin 4 and so on.

The DevPack ID override possibility in figure 8.13 is just a simple resistor that could identify itself to the SensorTag. The ID pin is simply a signal where the SensorTag measure the connected resistance to ground. This is done to automatically identify connected daughterboards. In the current implementation there is not specified any resistance, and the ID pin is now bypassed through the next daughterboard.

Chapter 9

Hardware and PCB layout

The hardware and PCB layout have been realized with the CAD software Altium Designer. The finalized schematics, PCB prints and 3D model could be found in appendix A and B.

Many components comes in different packaging and sizes. When selecting packages for use in the CMS, two main criteria have been balanced. The trade off between physical size and the possibility for manual assembly. All components have been selected so large that manual assembly is possible without major problems. But at the same time as small as possible to fulfill the requirements established in chapter 7.1. For example all resistors and capacitor's that could be SMD 0603 metric package have been selected. There exist several smaller SMD packages, but 0603 Metric was chosen as the perfect fit in the trade off.

The PCB is designed as a 4-layer board with a total thickness of approximately 1.6 mm. All layers is used as mixed signal layers. But the two inner layers have large ground planes with fewer signals routed on them compared to the outer layers. The boards was designed for external PCB production with minimum capabilities summarized in table 9.1.

Figure 9.1 and 9.2 is pictures of a empty PCB straight from the PCB manufacturer.

Table 9.1: Requirements needed to produce the PCB

Layers	4
Material	FR-4
Minimum board dimmension	50 mm X 45 mm
Minimum conductor width	0.15 mm
Minimum conductor spacing	0.15 mm
Copper to edge	0.3 mm
Plated coponent / via diameter (mechanical)	0.3 mm - 6.30 mm
Minimum hole spacing	0.25 mm
Minimum hole to edge	0.4 mm
Minimum annular ring	0.15 mm
Solder resist clearance	0.1 mm
Minimum silkscreen line width	0.15 mm

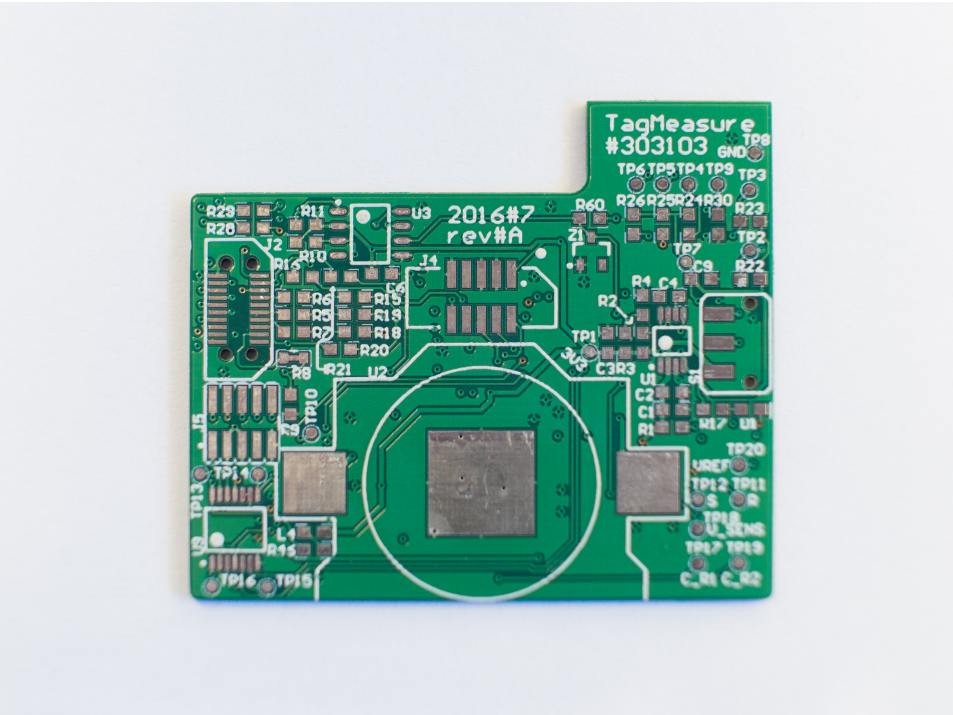


Figure 9.1: Top side of the designed and manufactured PCB

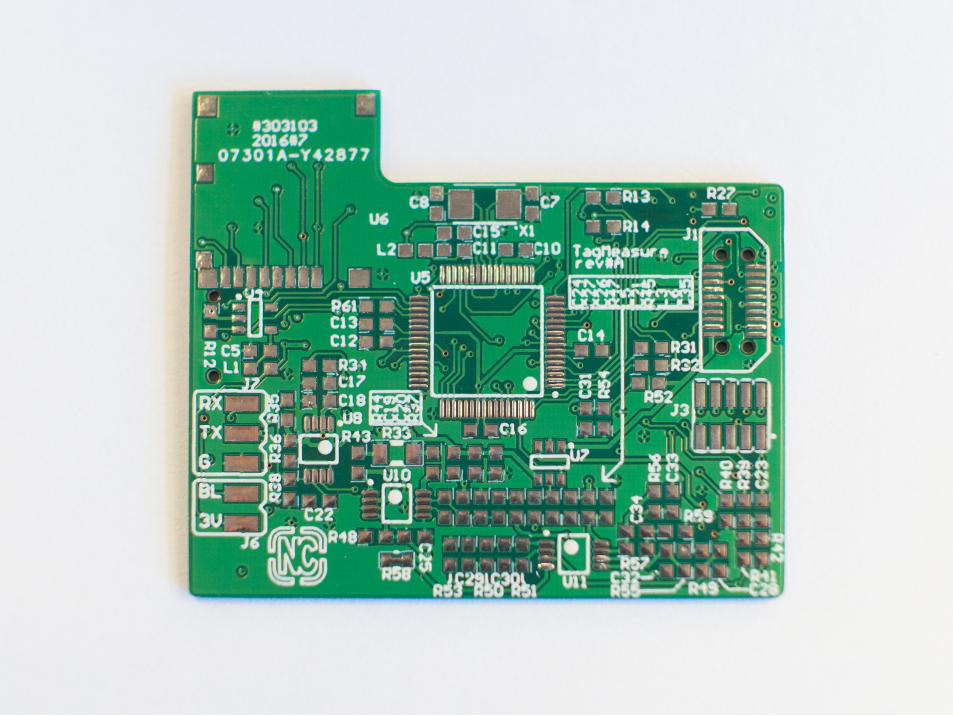


Figure 9.2: Bottom side of the designed and manufactured PCB

Chapter 10

Assembly

This chapter will go through the assembly process of the CMS. This includes how things were soldered, tested and challenges that have arisen during the assembly. Only the key takeaways will be explained here, central soldering techniques and similar procedures will not be explained.

When the PCB's arrived from the manufacturer, the author first soldered one single PCB. It was done to quickly check for errors in the schematics, PCB-layout and to verify correct operation. Later in the process, the assembly of 4 cards was outsourced to an external assembly laboratory at the Norwegian University of Science and Technology called "Elprolab". It was done to release time resources when the laboratory soldered the boards.

10.1 Manual assembly by author

The manual assembly was done in a incremental method. The different logical parts in the circuit was soldered on and verified before the next part was soldered. In this way both circuit flaws (schematic error, pinout error ...) and soldering errors could be detected early. This does also simplifies the troubleshooting, which is highly appreciated. Figure 10.1 and 10.2 shows pictures from the assembly process.

The only error found in the assembly process was an silkscreen error. The silkscreen markings for "R35" and "R36" had swapped place. This lead to assembly of wrong resistor values with an result of wrong output voltage from one of the regulators. Figure 10.3 shows the swapped resistors in schematic. The fix was simply swapping resistors back. In the PCB prints found in appendix B the error has been corrected.

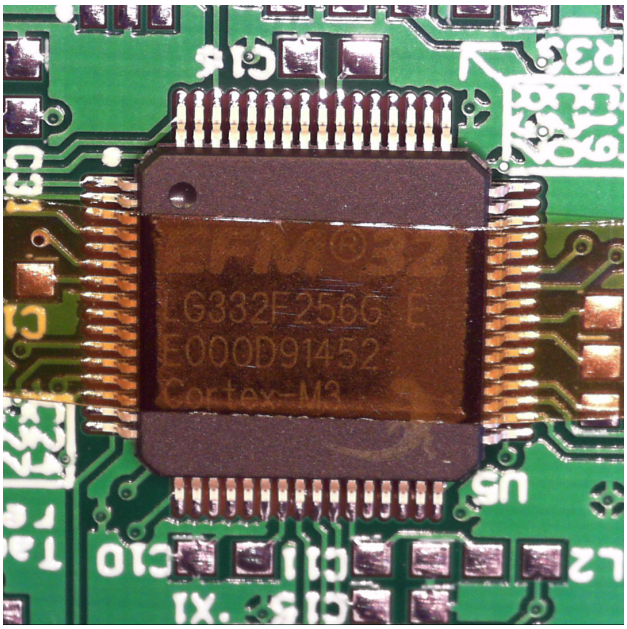


Figure 10.1: Assembly of the MCU, hold in place with kapton tape

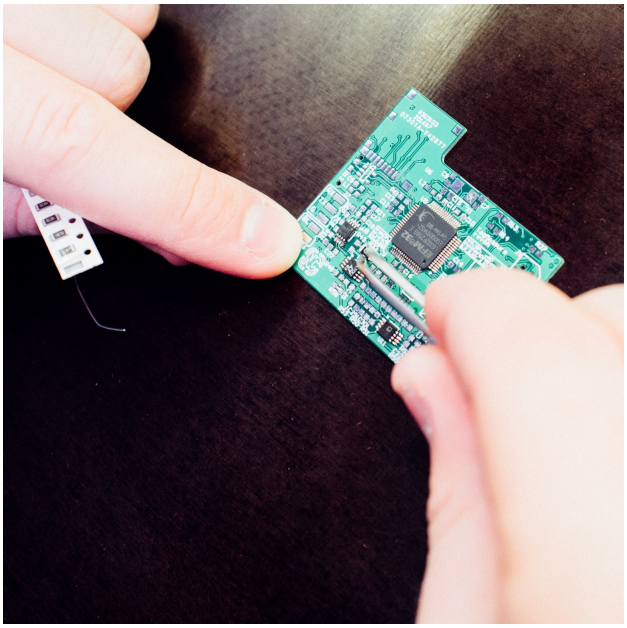


Figure 10.2: Manual placement by usage of a tweezers

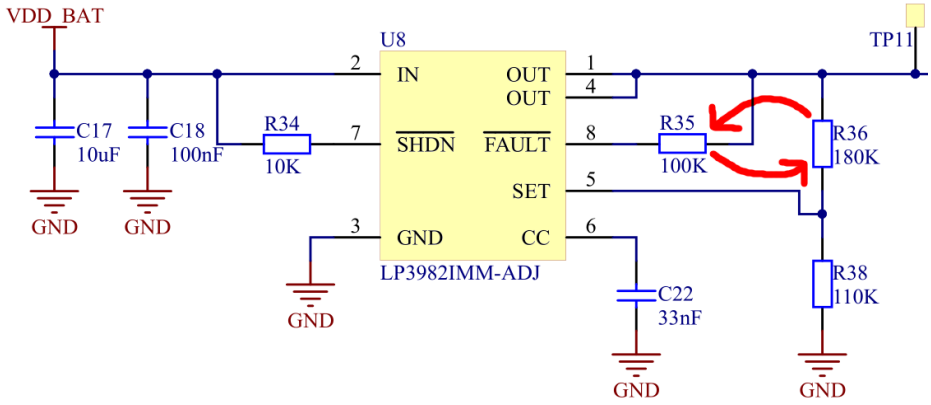


Figure 10.3: R35 and R36 silkscreen markings swapped on PCB layout

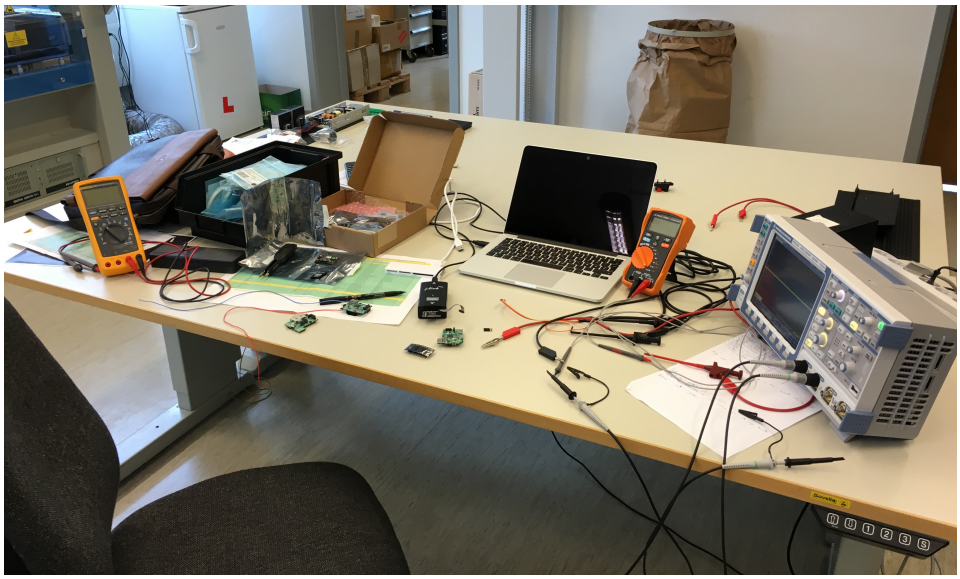


Figure 10.4: Lab bench while debugging the external assembled boards

10.2 External Assembly

Unfortunately when the boards came back from the external assembly, none of them was working as expected. Investigation was started to localize the various problems with the different boards. Figure 10.4 shows the bench used by the author while debugging the cards. Figure 10.5, 10.6 and 10.7 illustrates the condition of the passives when the boards was received by the author. As the figures shows many of the components isn't soldered properly. Therefore the soldering on the affected components was reworked. Even some components wasn't properly aligned with the PCB. Figure 10.8 shows how multiple components is "flying". They was only soldered on one end, with the other end not aligned parallel with the PCB surface ("hanging in the air!"). The result of this was that the components did not function as intended. On one card some components was mounted with incorrect polarization, figure 10.9 shows how U1 (voltage regulator) and J4 (SWD connector) is mounted the opposite direction.

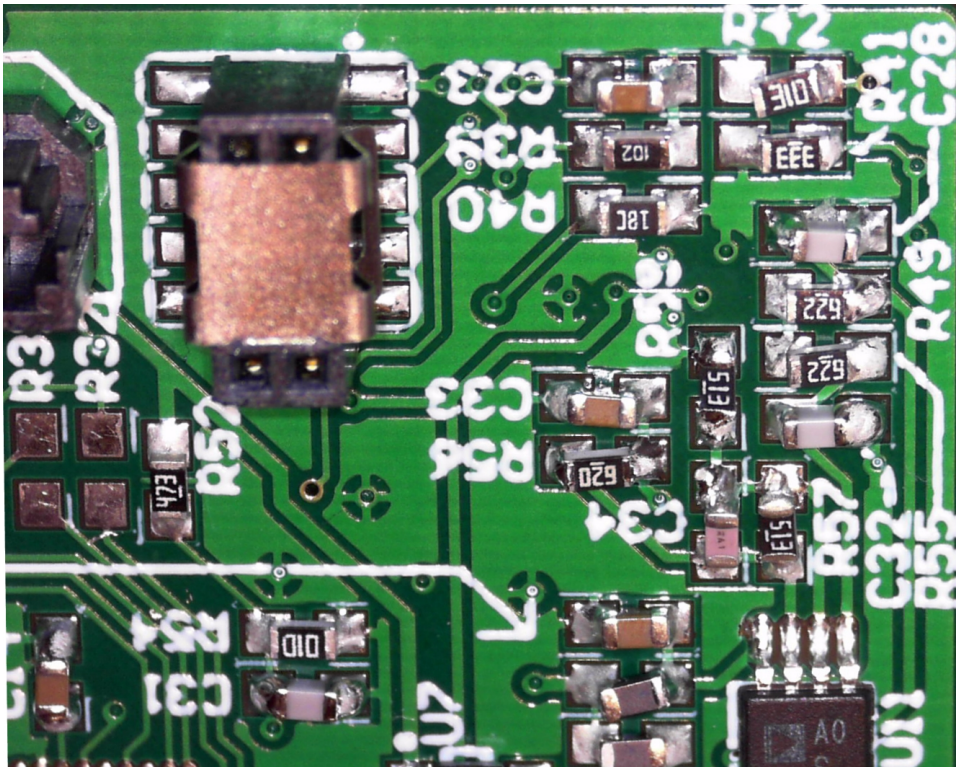


Figure 10.5: Inadequate soldering of 0603 passives

After these errors was corrected all of the cards did power on and programming through SWD did work. When the firmware from chapter 11 was flashed to the boards, none of the boards did pass the analog power on self test or the analog

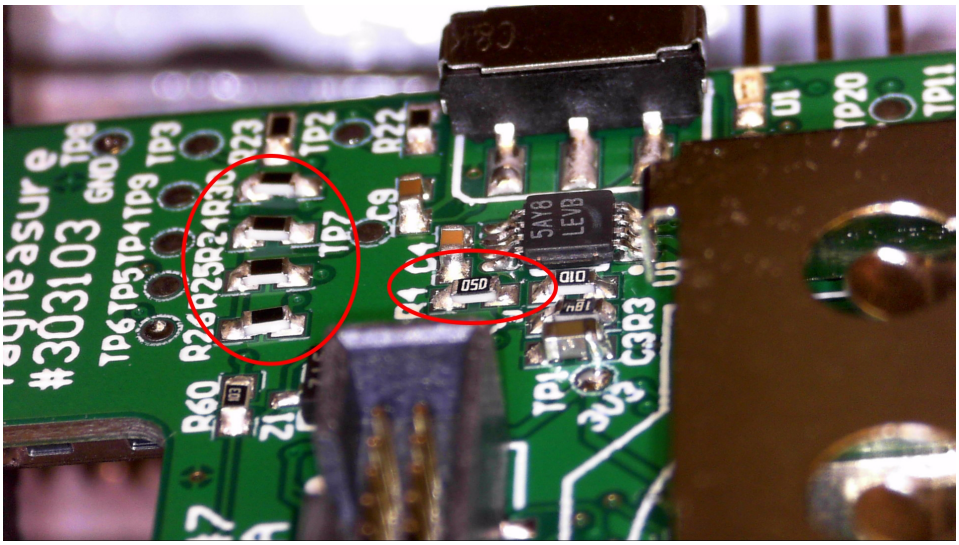


Figure 10.6: Several 0603 resistors with inadequate soldering

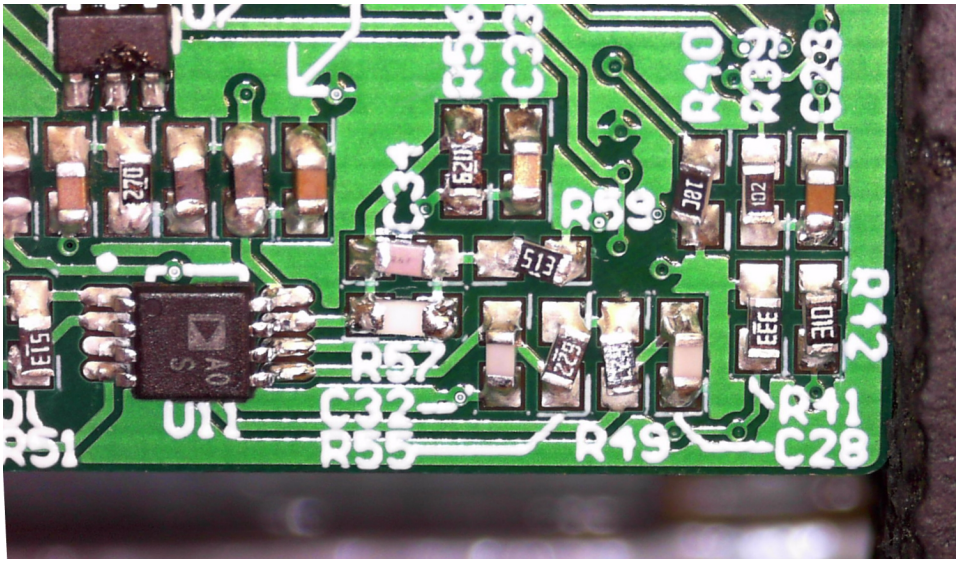


Figure 10.7: Not proper alignment and soldering of 0603 passives

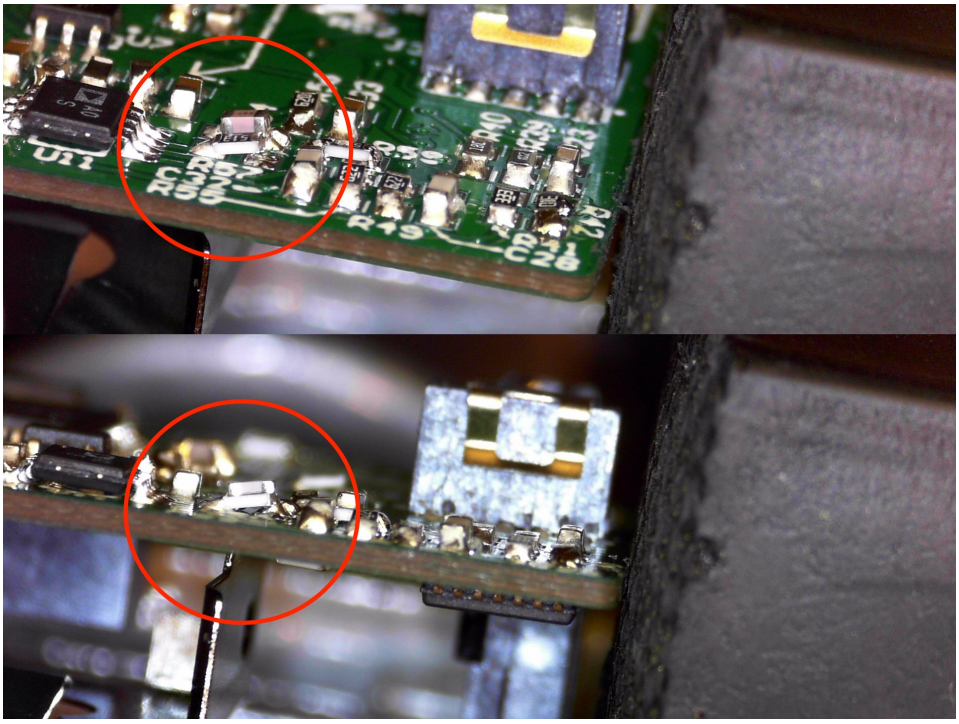


Figure 10.8: Multiple 0603 passives not properly aligned and ”flying” off the PCB

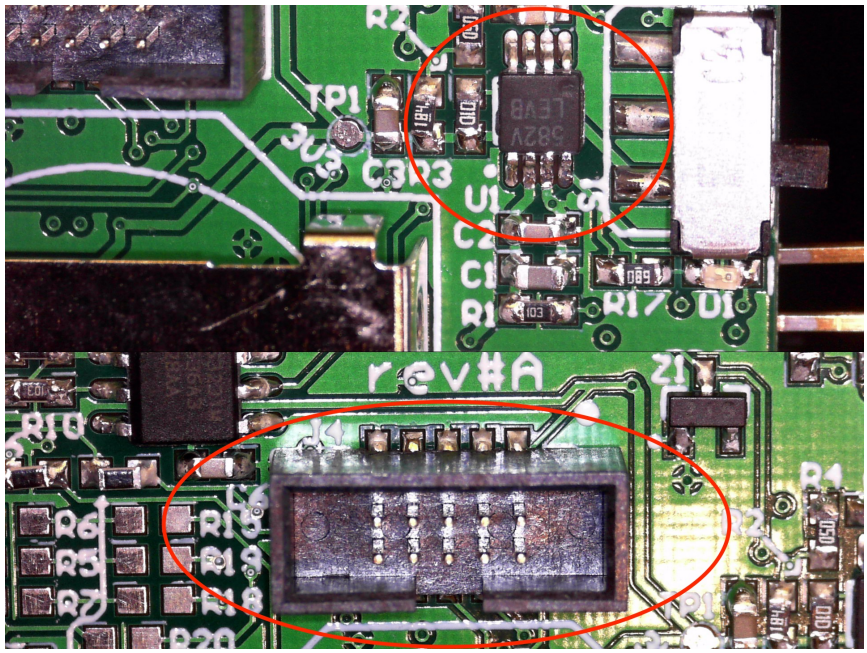


Figure 10.9: Several components soldered wrong way (U1 & J4 in this picture)

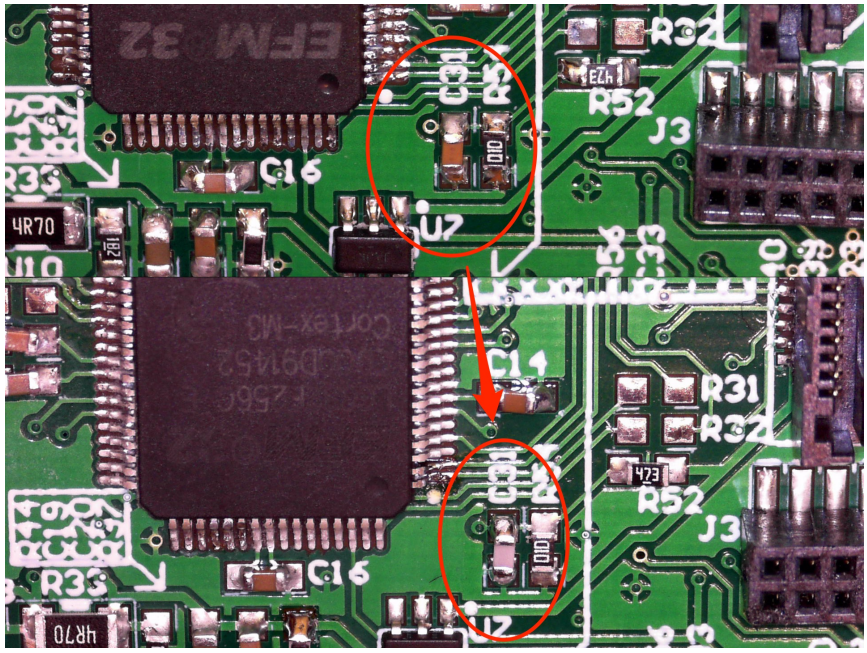


Figure 10.10: Capacitors changed in assembly, unknown C31 top, correct C31 bottom

front end calibration. The digital parts of the circuit was working at this stage. The boards was forced to continue even if the self test failed. Analog measurements did show that ADC channels had noise on some boards, while some other boards had ADC channels that was constantly stuck on a fixed position. Further debugging found more issues with the soldering. Some resistors didn't have any metal on one end, and thus wasn't making connection. This fault wasn't clearly visible and therefore the visual inspection didn't discover this. In addition, components supplies had been mixed in assembly. Supplied precision NP0 capacitors was replaced with unknown capacitors. Figure 10.10 shows one example of a capacitor that is changed in assembly. The top capacitor (brown) is a unknown capacitor and the bottom capacitor (white/light pink) is the correct one. In the analog front end circuit several capacitors have a purpose of filtering noise before and in the amplification circuit. When those capacitors is changed to an potential unknown value and tolerance, the performance of the circuit is degraded. All of the capacitors that had a visual deviation was replaced with new capacitors. But the boards still had too much analog noise. The likely cause is that even more precision components is changed, but with components that is visually identical, and therefore not discovered by visual inspection.

As a result of this none of the four external assembled boards had an adequate performance. Further debugging is suspended because of time restrictions. If debugging and performance testing is resumed at a later stage, the root cause of the issues will probably be found and corrected. The debugging and error corrections described here has primarily been done by the author. Some of the physical corrections have the "Elprolab" done for the author.

Chapter 11

Firmware

The firmware running on the CMS is written in standard C with additional usage of Silicon Labs ”emlib” library made for EFM32. Emlib is a low level peripheral support library that provides a streamlined API between multiple different devices. The main reasons for choosing to use ”emlib” is streamlined usage of low level peripherals and less implementation time. This chapter will only go through central elements of the firmware such as key design concepts, modules and system behaviour. The actual implementation is best described by looking at the full source code given in appendix C (Digital attachment).

11.1 Design concepts

The firmware in the CMS have been implemented as simple as possible. This is done to minimize the potential errors. The system have been designed to do few tasks, but do them well. As described in chapter 7.1 the system have a goal to easily enable the user to take one or more energy measurements and later analyze them. Therefore it is not necessary to make the CMS firmware too complex. Under implementation each sub module have been tested separately to ensure that the modules works before implementation together with the rest of the system.

11.2 Firmware modules

The firmware consists of multiple different software modules. Figure 11.1 illustrates which central firmware modules that have been implemented and used. At the top is the measurement state machine, controlling the program flow and operation modes for the whole system. The colored libraries below is necessary sub modules needed by the state machine to perform it’s tasks. These libraries have been implemented in a such way to give a clear separation between the modules and to enable upgradeable and easy replacement of modules. For example: At a

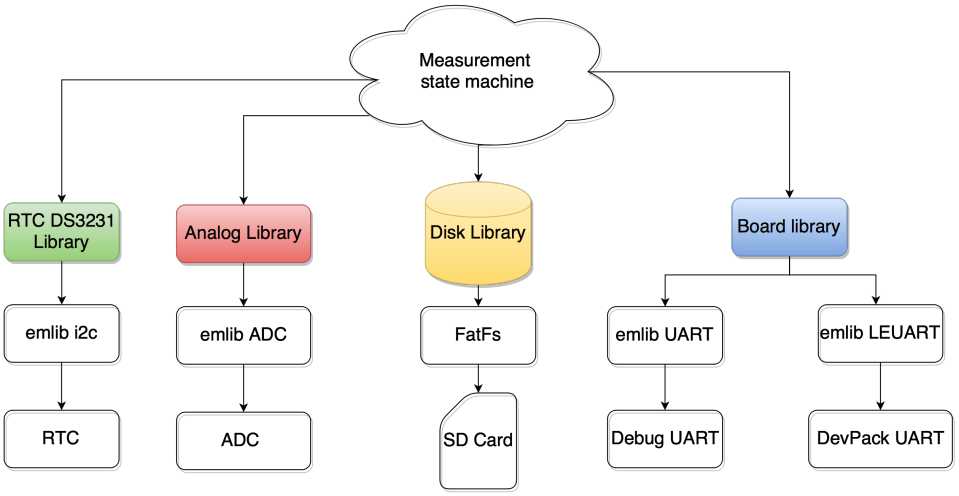


Figure 11.1: Map over central firmware modules implemented in the CMS

later stage we want to replace the SD card storage with a different new storage method. With the implemented solution, the disk library could be changed to use the new storage system instead of the FatFs module. From the perspective of the measurement state machine, such a change will not be noticeable as long as the API for the disk library is not changed.

11.3 System behaviour

Figure 11.2 illustrates the possible operation modes for the internal state machine. This is a rather simple state machine where the CMS is either in ”Idle” mode or in ”Acquisition”mode. When going to or from the acquisition phase, certain tasks must be done in order to correctly do an acquisition run. Task examples is calibration, mounting the filesystem and recording the start time. Algorithm 11.1, 11.2 , 11.3, 11.4 and 11.5 shows the implemented states with pseudocode. Note that the pseudocode is grossly simplified, for example data handling between the interrupt routine and main execution thread is protected, but this level of details isn't shown. As mentioned earlier the actual implementation can be viewed in appendix C (Digital attachment).

Algorithm 11.1 Pseudocode for IDLE state

```
if getDUTCommand() == INIT_ACQUISITION then
  state = ACQUISITION_INIT
end if
```

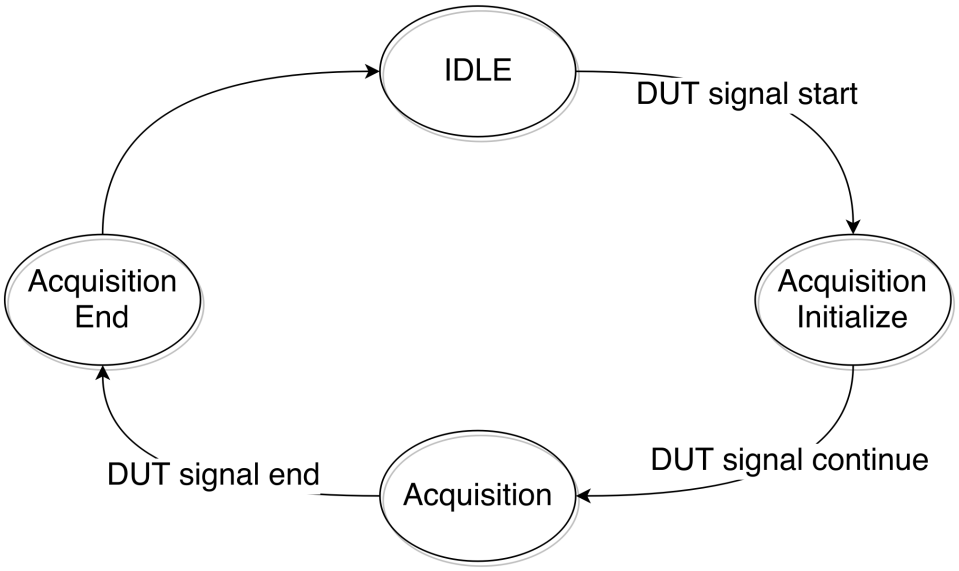


Figure 11.2: Implemented state machine in the CMS

Algorithm 11.2 Pseudocode for ACQUISITION_INITIALIZE state

```
filenamePrefix = getFilenamePrefix()
diskMount()
currentTime = getTime()
filename = formatFilename(currentTime, filenamePrefix)
diskOpenFile(filename)
storeCalibration(analogDoCalibration())
while getDUTCommand() != START_ACQUISITION do
  //Wait for DUT to start test
end while
state = ACQUISITION
```

Algorithm 11.3 Pseudocode for ACQUISITION state

```
if newResult then
  storeAnalogRead(storeResult)
  newResult = 0
end if
```

Algorithm 11.4 Pseudocode for Interrupt Service Routine, ACQUISITION state

```
result = getAnalogResult()
sumResult += result
sumCount += 1
if sumCount > ACQUISITION_CONTAINERSIZE then
  storeResult = sumResult
  newResult = 1
  sumResult = 0
  sumCount = 0
end if
```

Algorithm 11.5 Pseudocode for ACQUISITION_END state

```
storeCalibration(analogDoCalibration())
diskCloseFile()
diskUnmount()
state = IDLE
```

Chapter 12

Analyzer software

The analyzer software is targeted to run on a standard computer and therefore not running on the CMS itself. The software have been developed to take the generated data from the CMS and then sort, process and analyze the dataset. The software may also graph results according to which parameters the user want to study. The analyzer is implemented with the language Python, version 3.5. It enables the user to easily run the software on any platform where Python 3.5 is supported. This chapter will go through the design concepts, software construction, software operation and finally a detailed look into the calibration adjustment software. The developed source code can be viewed in appendix D (Digital attachment).

12.1 Design concepts

The analyzer software is made as modular and flexible as possible. Then the software may easily support analysis of different aspects of the raw data. The internals of the software is divided into several classes and functions with distinct logical separations. The analyzer software is designed for users that is able to do some minor manipulation of the source code directly. Therefore there isn't implemented any menu system or user interface. To select what and how the data should be analyzed, different function calls must be selected in the main loop. In this way the analyzer software could still be flexible without the need for implementation of a complex menu system.

To transfer the raw data between the CMS and the analyzer software, a common file format is necessary. CSV was the selected format. This is a simple and flexible format where the data is stored in ASCII encoded files with comma separated values.

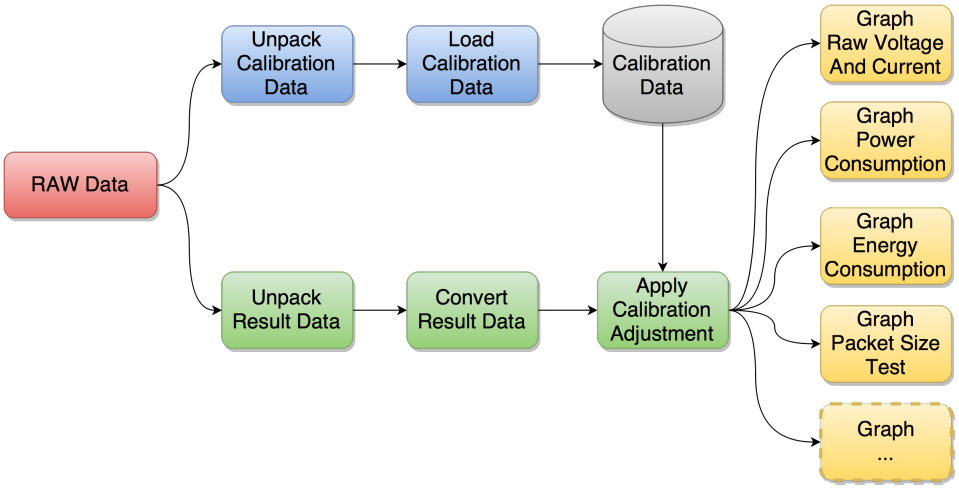


Figure 12.1: Primary data flow in the analyzer software

```
nalyzer>python src\main.py data\packetsizeZero160516 1
INFO:root:Initializing FileFramework
DEBUG:root:<class 'framework.FileFramework'>.__init__ (FileFramework)
INFO:root:Found 2 files for Node 0
INFO:root:Initializing CalculationFramework
```

Figure 12.2: Example of analyzer software operation

12.2 Software construction

The primary data flow in the analyzer software is shown in figure 12.1. The RAW data is first loaded from the CSV files. Thereafter, the calibration and result data is separated and unpacked. The calibration data is then loaded, and a set of transformation functions is generated and stored for later use. The result data is converted before the transformation functions from the calibration is applied. After the calibration adjustment have been applied, the result dataset is ready for use. In figure 12.1 several different graphing scenarios is shown. At this stage, it's up to the user to select what to do with the dataset.

12.3 Software operation

This section will give a very brief introduction to operation of the software. To run the software, *"main.py"* is called from a python shell. Figure 12.2 shows an example of starting the analyzer software from a Windows shell. The first argument to the analyzer software is a path to the dataset folder. The second argument is the number of nodes that should be analyzed. The dataset from the different nodes must be in separate folders. An example is found in appendix D (Digital attachment).

Figure 12.3 and 12.4 shows two example graphs generated from the analyzer

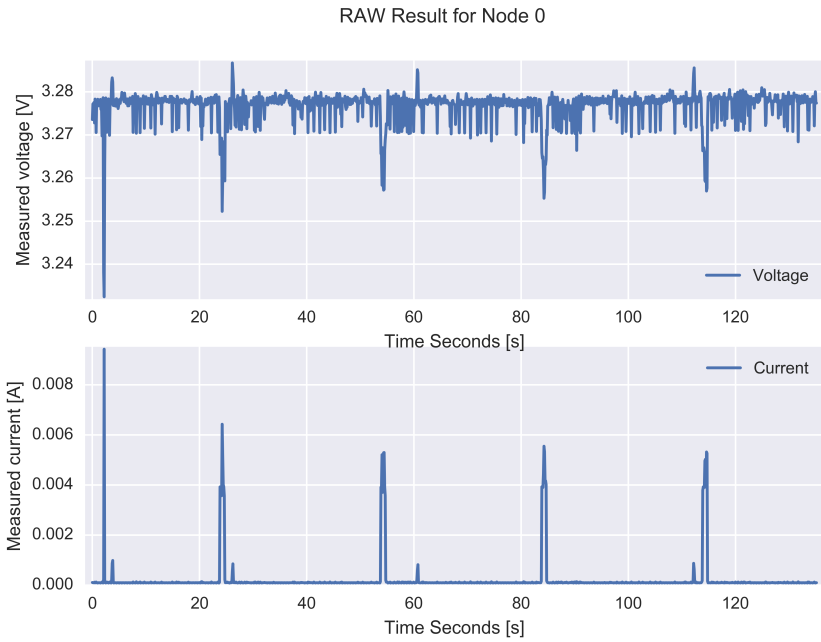


Figure 12.3: Example of graph from analyzer, voltage and current is shown

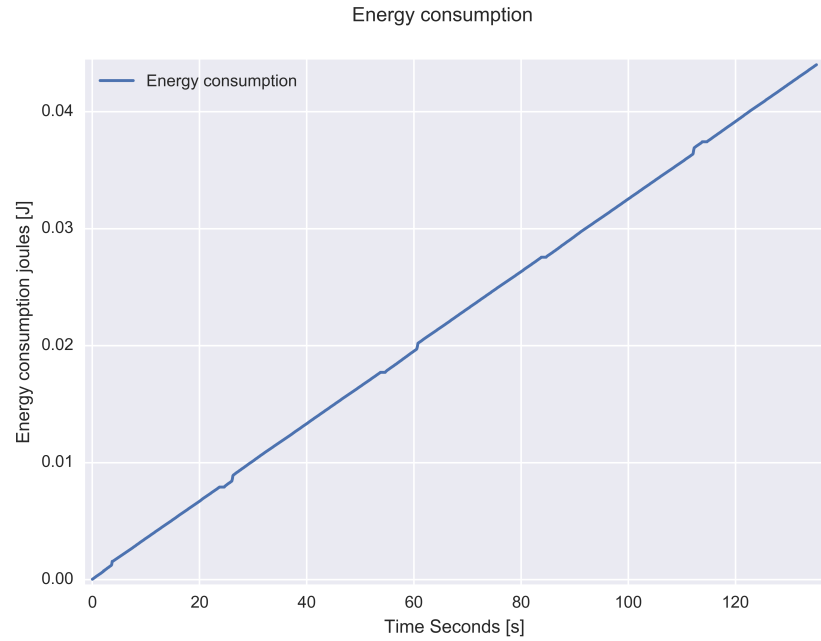


Figure 12.4: Example of graph from analyzer, energy consumption is shown

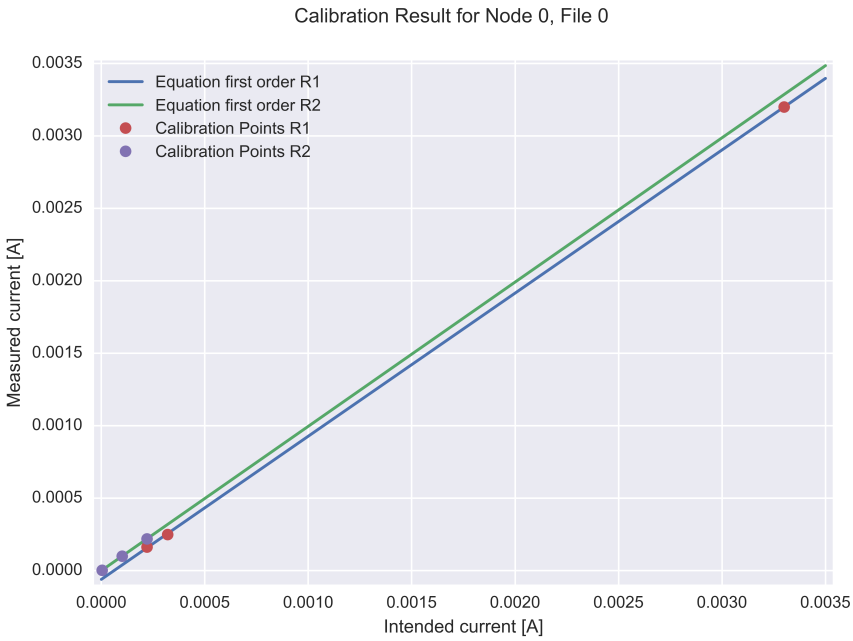


Figure 12.5: Calibration data plotted

software. This figures show how the final result from the analyzer software might look like.

12.4 Calibration adjustment & Dual Current Range

The dual current range introduces a challenge. Due to non linearity and offset between the ranges the transition must be done correctly. In figure 12.5 the data points from the calibration is shown. ”R1” is a code for ”Current Range 1” and respectively the same for ”R2”. The shown data points is the actual measured values from the calibration. From these points, two separate linear functions is made. As figure 12.5 shows, the two linear functions (ranges) have a considerable offset. In order to transfer between the two ranges a mathematical expression must be made. Figure 12.6, 12.7 and 12.8 shows three different orders of the transfer function. As we can see, the first order has smallest deviation from the original ranges. Therefore the first order transition is used (figure 12.6).

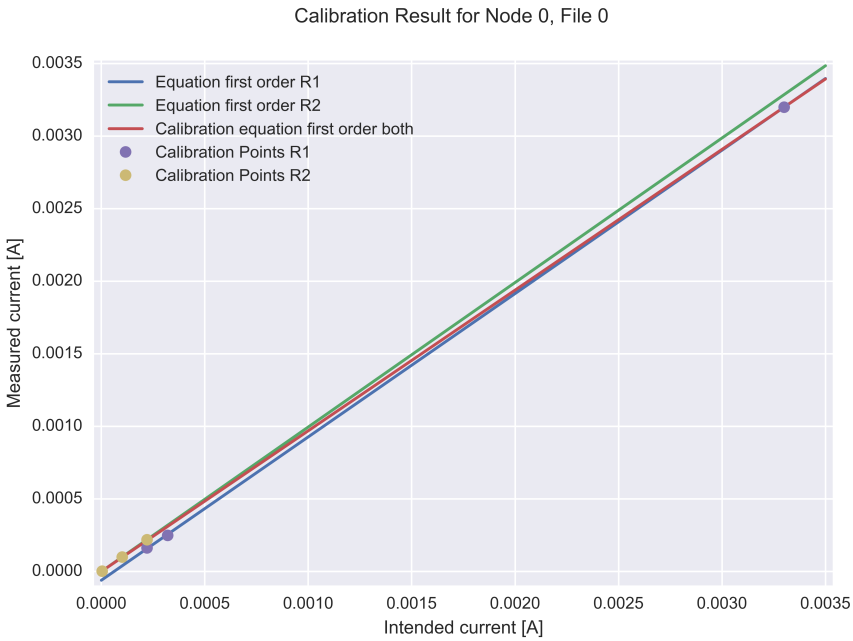


Figure 12.6: Transition function between current ranges, first order

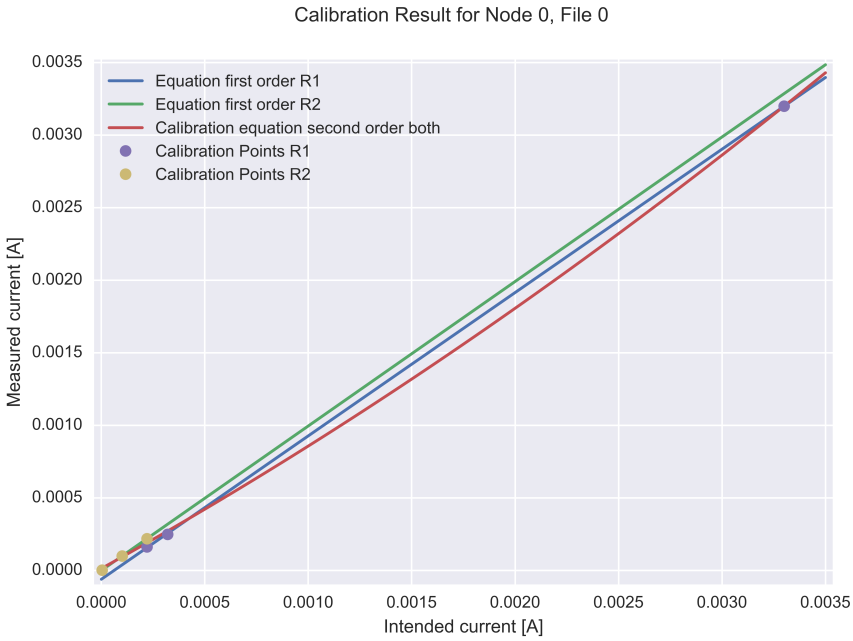


Figure 12.7: Transition function between current ranges, second order

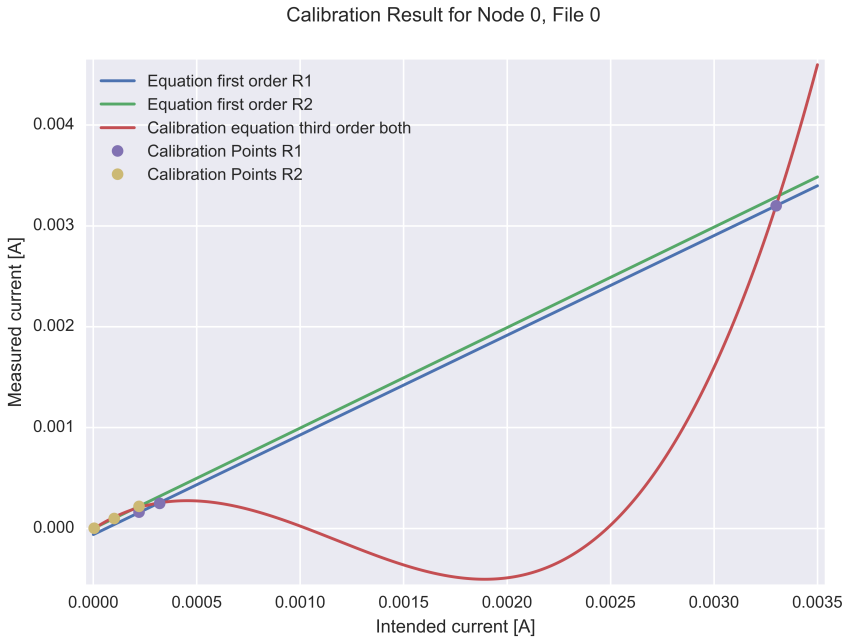


Figure 12.8: Transition function between current ranges, third order

Part IV

Test Execution & Analysis

Chapter 13

Packetsize test

This test will measure the energy consumption while sending different packet sizes over the network. This will show how big difference there is between small and large packets in energy consumption. This test is one of the essential tests for mapping the possibility of the store and forward method.

13.1 Implementation

The core of this test is a state machine running on the SensorTag. This state machine automates the testing so the system will test different packet sizes without any manual interaction. To control the behaviour of the test, several parameters are defined in advance. This is parameters such as: *packet_size_increment, packet_size_end , packet_send_count* and *send_interval*. Figure 13.1 illustrates the operation modes of the state machine.

To actually send packets over the network, UDP packets has been used. The packets is sent to a server and the packet size is controlled by the state machine. It's important to notice that the referred packet size is the actual payload size in the UDP datagram. In other words, the headers added by UDP and IPv6 isn't counted as a part of the packet size. The reason for this is that an application also have the same overhead and therefore this test is comparable to actual application usage. If wanted, it is also possible to adjust the result data in retrospect as this overhead is static.

When it comes to the energy measurement, the system actually send multiple UDP packets at each packet size. The parameter *packet_send_count* controls how many times each packet size should be sent. For example if *packet_send_count = 50* the system will send 50 packets of size N thereafter the system sends 50 packets of size N+INCREMENT. The reason for doing this is described earlier in section 6.1.

The complete code could be found in appendix C (Digital attachment). The files "packetSizeTest.c" and "packetSizeTest.h" contains the state machine and test

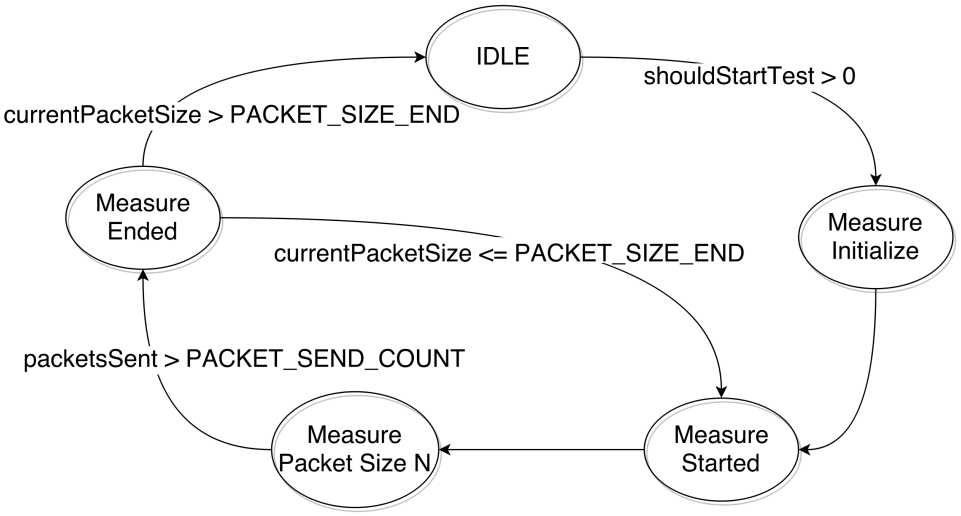


Figure 13.1: Implemented state machine for packet size test

code. The functions are called from ”cc26xx-web-demo.c” which is the ”main” file in this project.

13.2 Execution

This test was conducted in two rounds. The first round was executed with the following parameters:

Packet size increments:	5	Bytes
Packet size end:	1025	Byte
Packet size send count:	100	
Send interval:	1	Second

When this test was finished, the network analyzer actually showed that the system had stopped sending packets when it came over a packet size of 640 bytes. The network analyzer also verified that until this, all packets was received successfully. Figure 13.2 shows the last packets received in the network analyzer. The reason for the packets dropping is probably that this specific WSN setup reaches a limit for max allowed packet size transferred on the 6LoWPAN net. Since 640 bytes without overhead is a reasonable amount, this limit wasn't investigated further for now. The execution time was $1\ Second * 100 * 1025/5 = 20500\ Seconds$, this is $\approx 5,7$ Hours execution time.

After the first round a second round was done. This time with smaller packet increment and only up to 640 bytes in packet size. The parameters for the second round was:

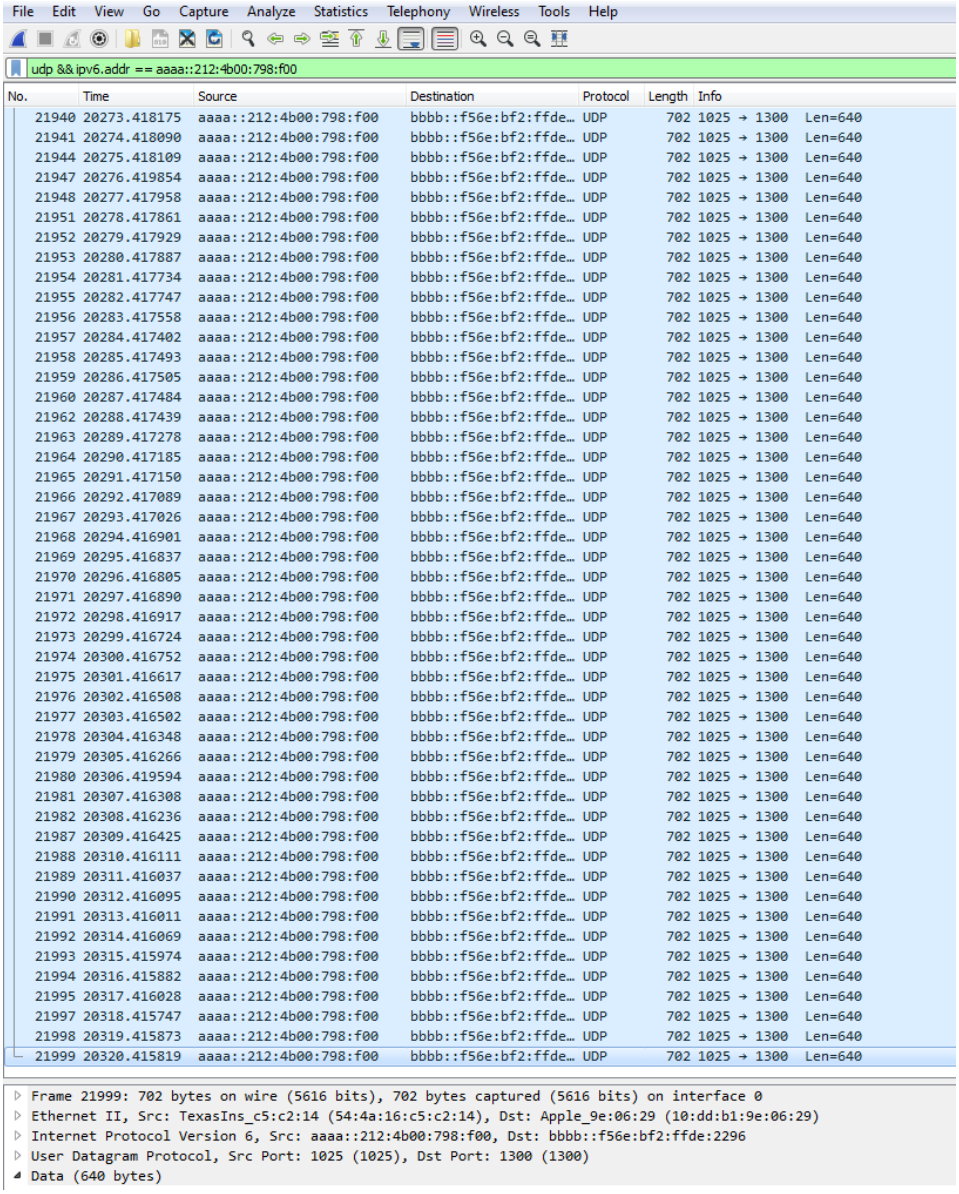


Figure 13.2: Screenshot of Wireshark after first round with the packet size test

Packet size increments:	1	Byte
Packet size end:	640	Byte
Packet size send count:	100	
Send interval:	1	Second

The execution time of round two was $1\ Second * 100 * 640/1 = 64000\ Seconds$, this is $\approx 17,8$ Hours.

13.3 Results

The result data is presented in the figures 13.3, 13.4, 13.5, 13.6, 13.7, 13.8, 13.9 and 13.10. They are analyzed in section 13.4. Figure 13.11 is presenting the total energy consumption of two selected packet sizes, this test is analyzed and compared in section 14.4.

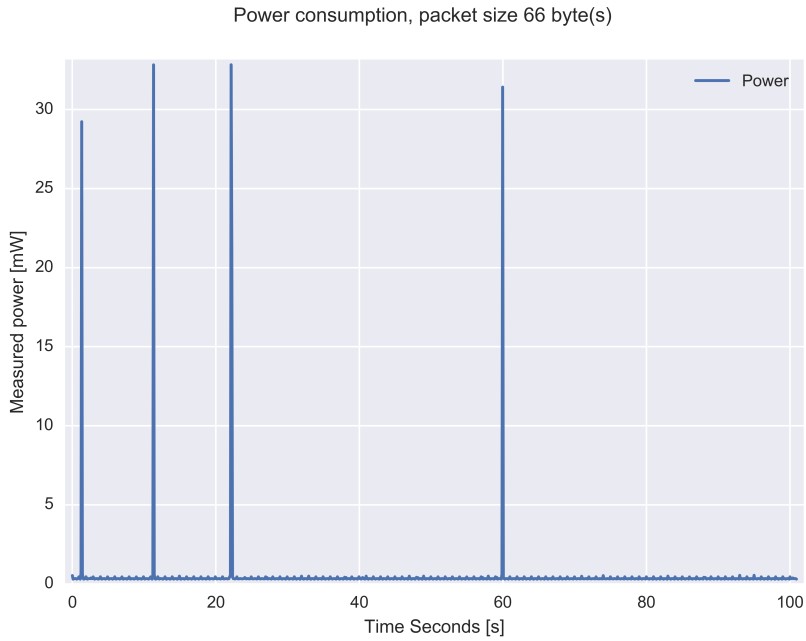


Figure 13.3: Power consumption during test with packet size = 66 bytes, result before correction

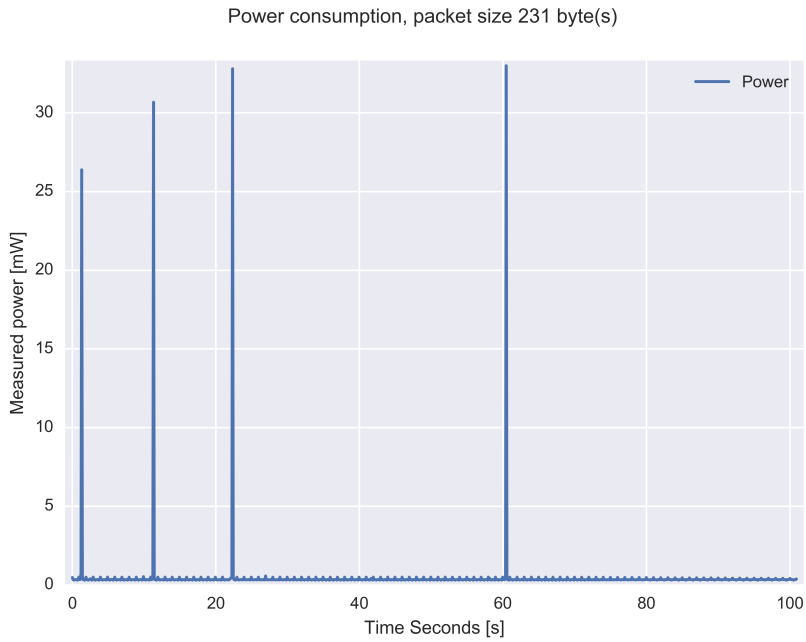


Figure 13.4: Power consumption during test with packet size = 231 bytes, result before correction

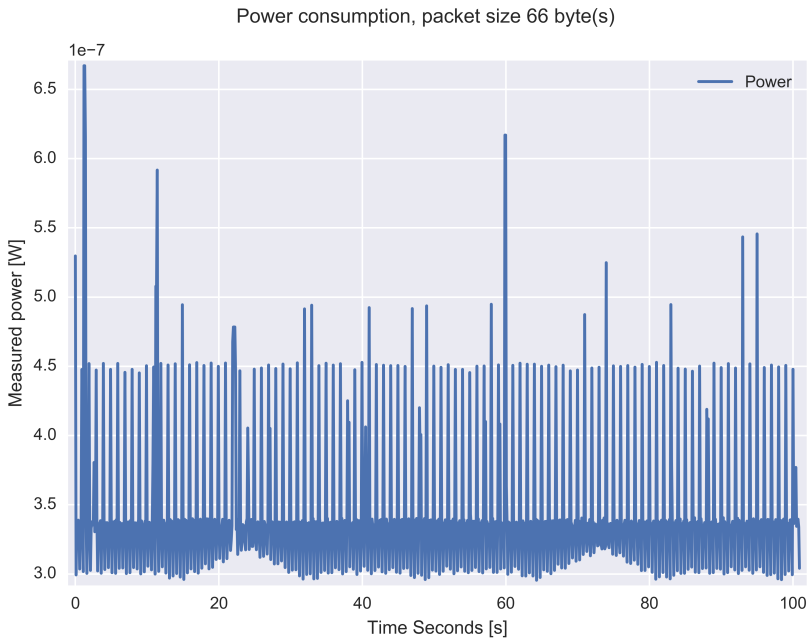


Figure 13.5: Power consumption during test with packet size = 66 bytes, result after correction

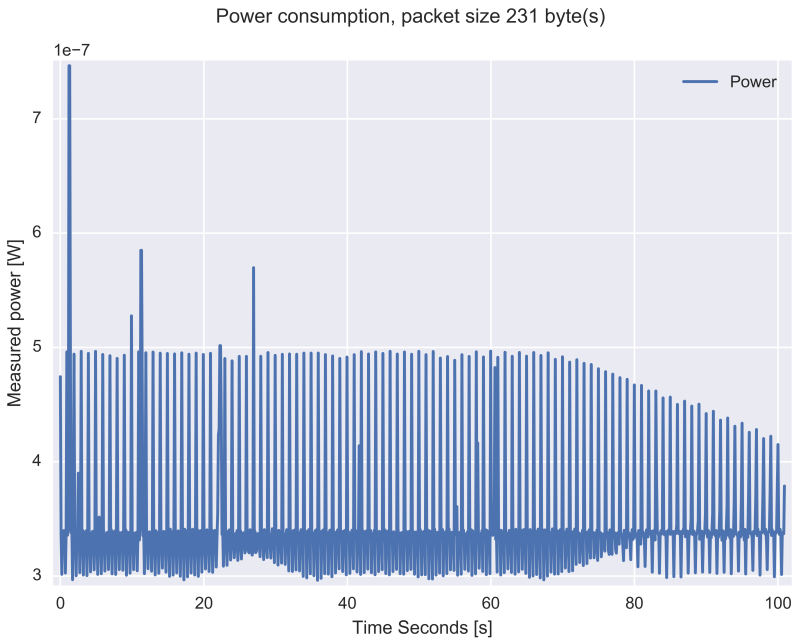


Figure 13.6: Power consumption during test with packet size = 231 bytes, result after correction

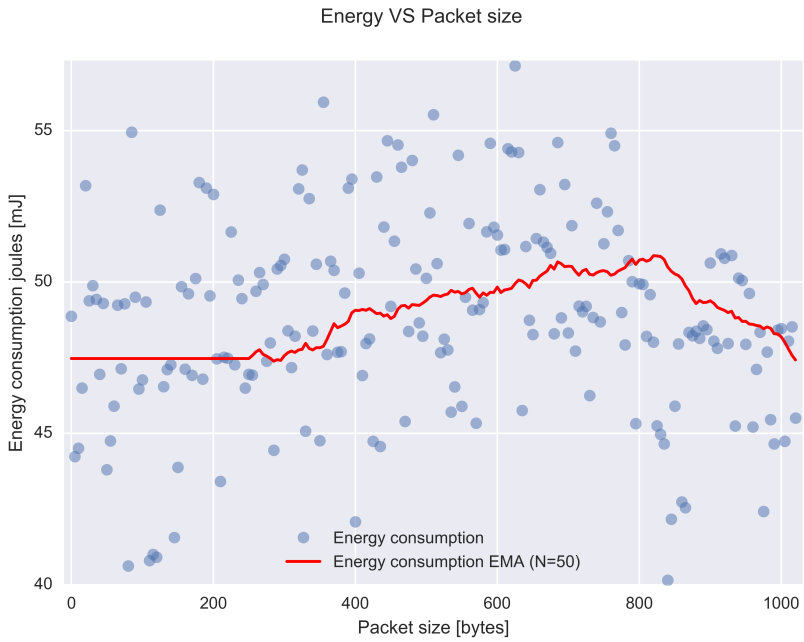


Figure 13.7: Energy consumption compared to packet size, first test round, before correction

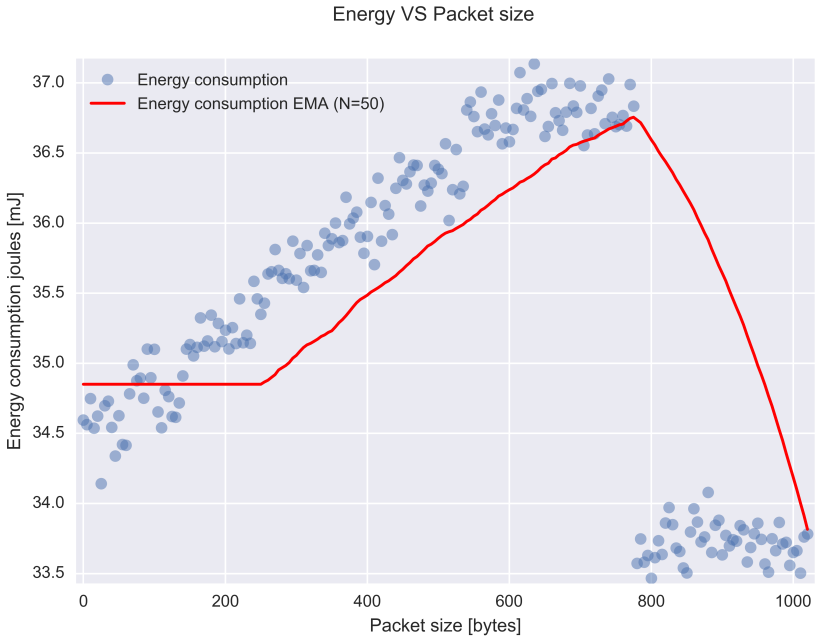


Figure 13.8: Energy consumption compared to packet size, first test round, after correction

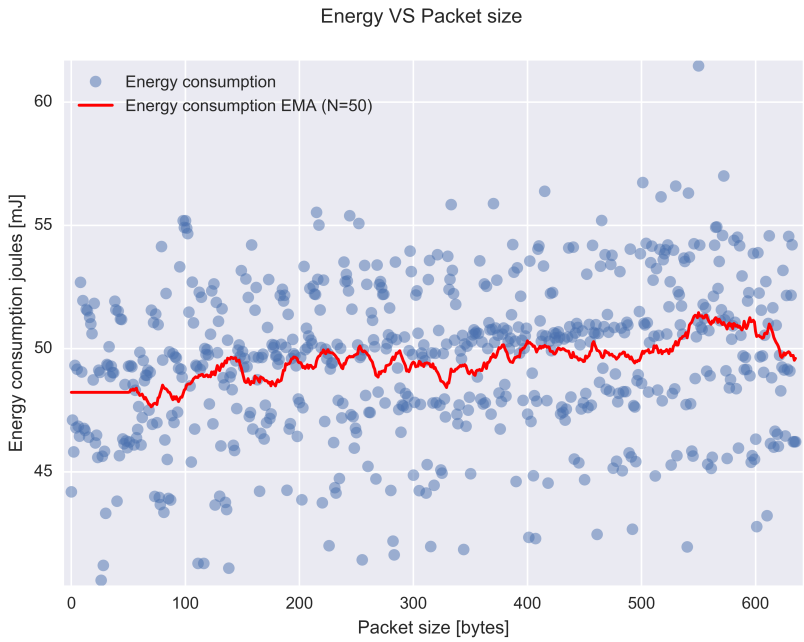


Figure 13.9: Energy consumption compared to packet size, second test round, before correction

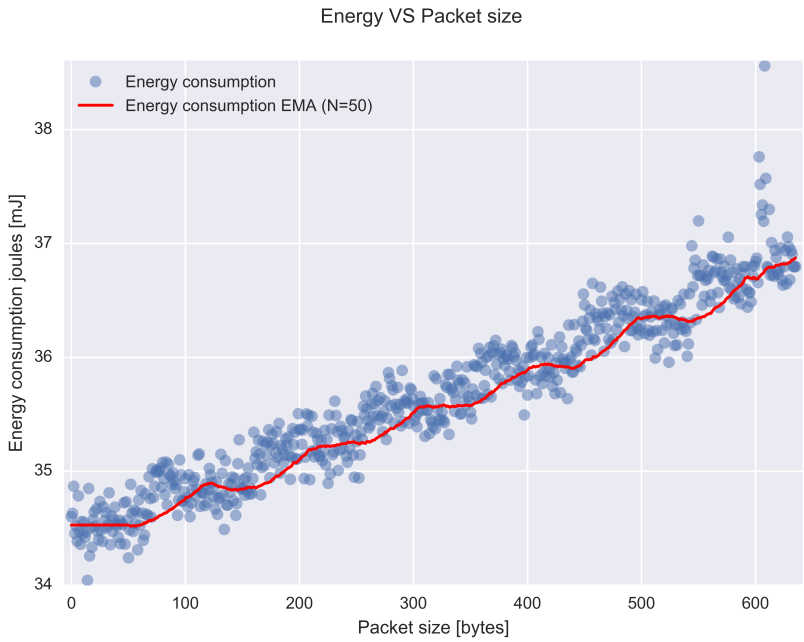


Figure 13.10: Energy consumption compared to packet size, second test round, after correction

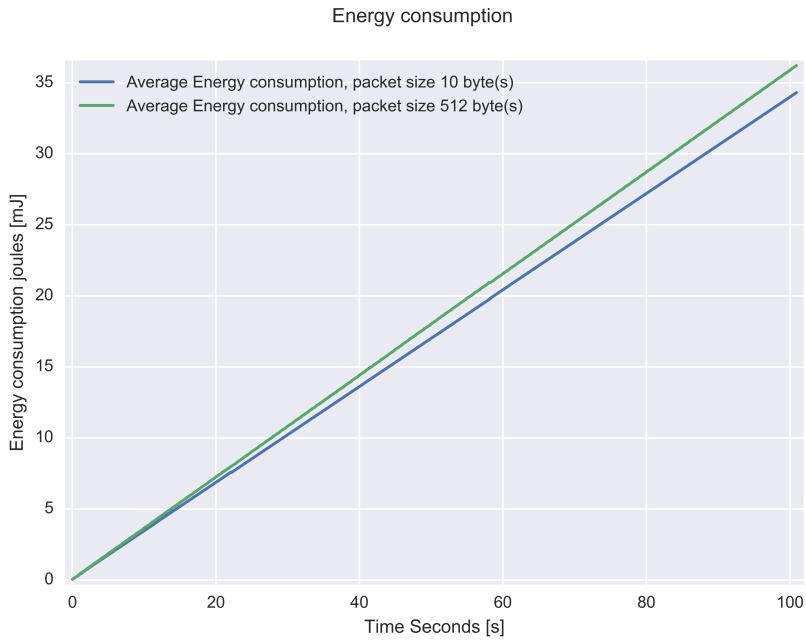


Figure 13.11: Total energy consumption of packet sizes 10 & 512 bytes, after correction

13.4 Analysis

Figure 13.3 and 13.4 is showing the power consumption through the test with an packet size of 66 bytes and 231 bytes. Both figure 13.3 and 13.4 is from the first test round as described in section 13.2. The interesting with these two plots is the four huge spikes found in the plots. They are orders of magnitude larger than the power consumption throughout the rest of the test. And they doesn't correlate to any of the packets sent on the network. In this test the packets was sent each second and as we can see the spikes occur approximately at $\approx 1, \approx 11, \approx 21$ and ≈ 60 seconds.

Since the spikes draws so much more power than the rest of the test, they could actually camouflage the difference in power consumption between the packet sizes. The spikes seem to be some kind of "housekeeping" that Contiki does. They does appear regularly in all of the packet sizes that have been investigated by the author. The viewed packet sizes (66 and 231 bytes) is randomly selected for the purpose of presenting the spikes in this report.

To remove the spikes a correction algorithm was applied. This does simply detect if the measured power consumption is over a threshold, if so, it ignores the "high" values and replaces them with the last known "normal" power consumption (normalization). Figure 13.5 and 13.6 is the power consumption after the correction has been applied. Hereafter this correction technique is referred to correction or power correction.

As figure 13.5 and 13.6 shows, the differences in power consumption is now evident from the plots. This does in fact substantiate that the spikes isn't relevant for this test and is some kind of "housekeeping" the system does regardless of the test.

Figure 13.8 is a complete energy consumption versus packet size plot from the first test round. This plot shows that the energy consumption does increase with the packet size. In this test, round the packets did drop after 640 bytes (as described in section 13.2). This result does also in a way reflect this behaviour. After a packet size of $\approx$ 790 the energy consumption drops and stay constant for the rest of the test. Why this drop in energy usage first happen at a packet size of $\approx$ 790 bytes isn't know.

The EMA in the plots is a exponential moving average of the data points. The EMA is added in the analyzer software and the plots indicate how many data points the EMA is calculated from. For example in figure 13.10 the EMA subset size is 50 data points (N=50). The EMA is added to highlight the trends in the data set.

Figure 13.7 is the same as figure 13.8 but without the correction technique. In figure 13.7 the data seems random and little is possible to grasp from the plot. However the EMA in figure 13.7 does in fact show a trend similar to the trend we observe in figure 13.8. Nevertheless, this also substantiate that the spikes isn't relevant. The same observations is also done from figure 13.9 and 13.10 which is from the second test round.

Figure 13.10 is the result from the second round with correction applied. The smaller increments used is really showing in this plot. From the plot we can clearly observe that the average energy consumption does increase with the packet size. The EMA in figure 13.10 does also indicate that the energy consumption does increase with $\approx$ 100 byte increments.

Chapter 14

Long Packetsize test

The test presented in chapter 13 did send each individual packet size 100 times. To verify that this is enough to give a result close to the expected value (*Law of Large Numbers* sec. 2.4), a much longer packet test was done. This test does increase the number of packets sent significantly. If the results from this test is similar to the result from chapter 13 this indicates that sending 100 times at each individual packet size is enough.

14.1 Implementation

This test is using exactly the same implementation as described in section 13.1. The only difference is the parameters defined for the state machine.

14.2 Execution

This test was conducted with the following parameters:

Packet size increments:	512	Bytes
Packet size end:	511	Bytes
Packet size send count:	25000	
Send interval:	1	Second

The test was started with a packet size offset of 10 bytes. This means only two packet sizes was tested and that was $packetSize = 10\ Bytes$ and $packetSize = 512\ Bytes$. The reason for only testing two packet sizes is the long execution time. Beside these differences, the test was conducted in the same way as the execution in section 13.2.

The execution time of the test was $1\ Second * 25000 * 2 = 50000\ Seconds$, this is $\approx 13,8$ Hours execution time.

14.3 Results

The result data is presented in the figures 14.1, 14.2, 14.3, 14.4 and 14.5. They are analyzed in section 14.4. Unfortunately the test with a packet size of 512 bytes did of unknown reason stop after 18872 packets sent and therefore the data set stops there. The data is still considered valid but not complete as the test was intended to send 25000 packets. Table 14.1 shows the total energy consumption used by this test and the test done in chapter 13.

		Packet size (Bytes):		Delta difference in percentage:
		10	512	
Total energy consumption:	Short test (data from figure 13.11)	34,3 mJ	36,2 mJ	5.39 %
	Long test (data from figure 14.5)	8,456 J	6,69 J	4,90 % (Sample size adjusted)
Number of packets sent:	Short test	100	100	-
	Long test	24998	18872	-
Average energy consumption per packet sent:	Short test	0,343 mJ	0,362 mJ	5,39 %
	Long test	0,338 mJ	0,355 mJ	4,90 %

Table 14.1: Result from long and short test compared

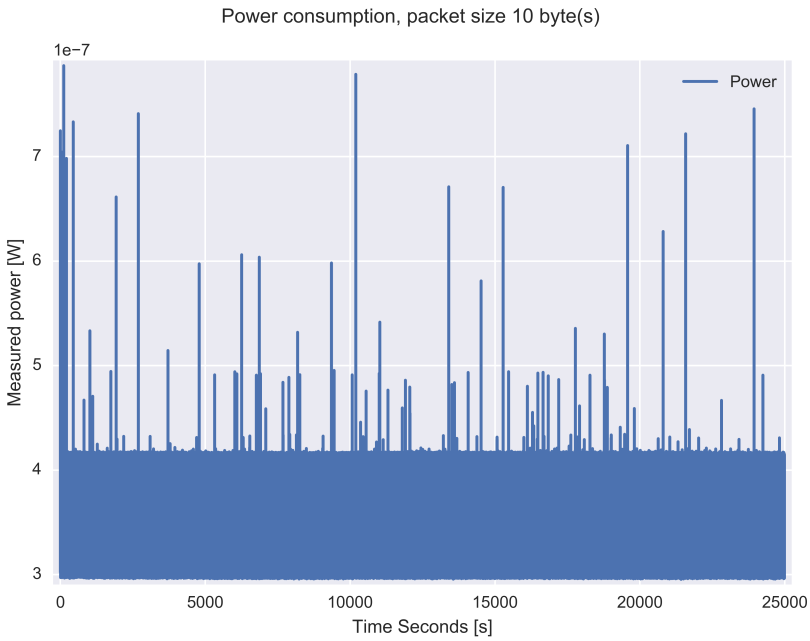


Figure 14.1: Power consumption during test with packet size = 10 bytes, after correction

14.4 Analysis

Figure 14.1 and 14.2 is the power consumption throughout the complete test. For reference 25000 $Seconds \approx 6,95$ $Hours$ and $\approx$ 18000 $Seconds = 5$ $Hours$. The interesting here is that the power consumption does actually stabilize rather quickly and forms a repetitive power pattern. Figure 14.3 and 14.4 is a zoomed version of figures 14.1 and 14.2. In addition this is combined with a overlay from the short test done in chapter 13 . When the packet size is 10 bytes the power consumption is stable after $\approx$ 45 seconds with a periodic power consumption cycle of $\approx$ 200 seconds. With a packet size of 512 bytes the power consumption is also stable after $\approx$ 45 seconds with a periodic power consumption cycle of $\approx$ 200 seconds. Besides some spikes in the start of the test, both packet sizes has approximately identical power consumption patterns. As we can see from figure 14.1 and 14.2 there is more spikes at the beginning than throughout the rest of the test.

In the combined figures 14.3 and 14.4 we can observe that the power consumption from the short test have similar characteristics as the long test. But they doesn't last long enough for one full period, figure 14.4 clearly illustrates this with the missing "tail" of the power consumption cycle. In both packet size tests, the short test have similar power spikes as the long, but fewer of them.

The total energy consumption for both packet sizes is shown in figure 14.5. The problems with the 512 bytes test could clearly be observed as the plot stops after $\approx$ 18000 seconds. From figure 14.5 we can observe that the difference in energy

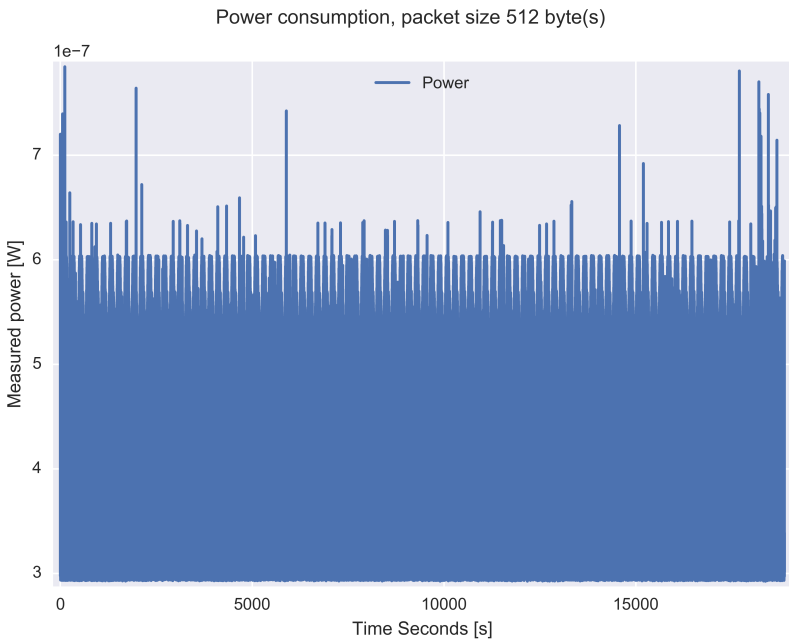


Figure 14.2: Power consumption during test with packet size = 512 bytes, after correction

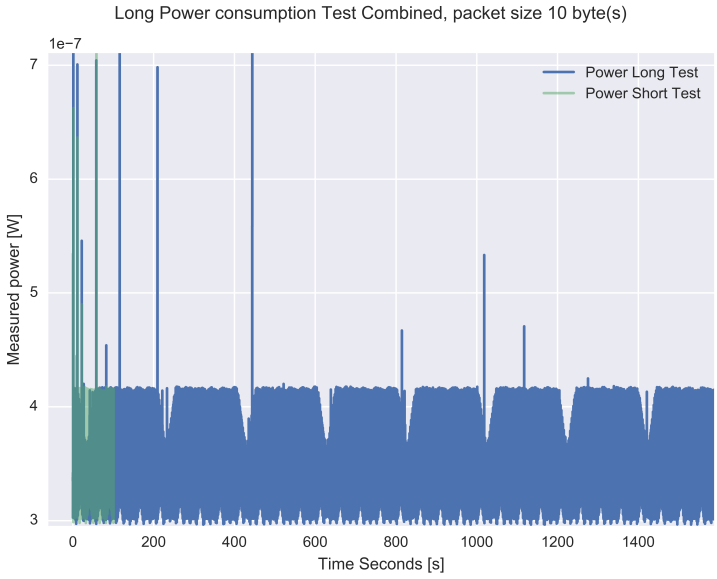


Figure 14.3: Power consumption during test with packet size = 10 bytes, after correction, combined with data from short test in chapter 13

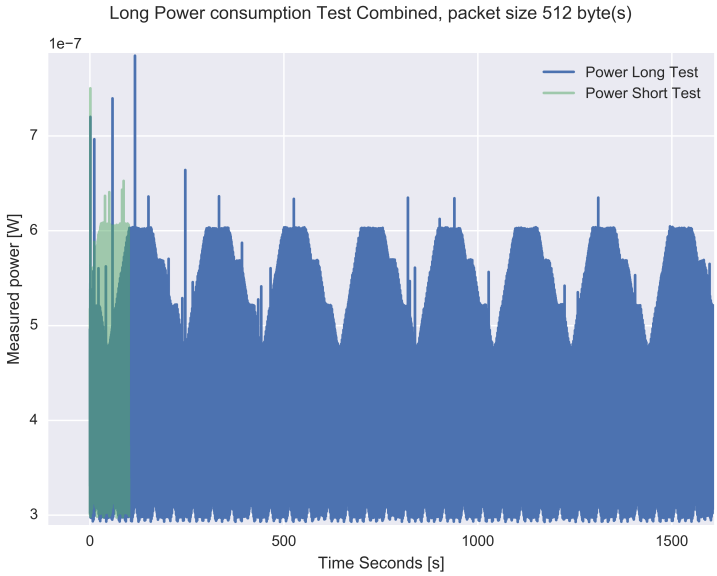


Figure 14.4: Power consumption during test with packet size = 512 bytes, after correction, combined with data from short test in chapter 13

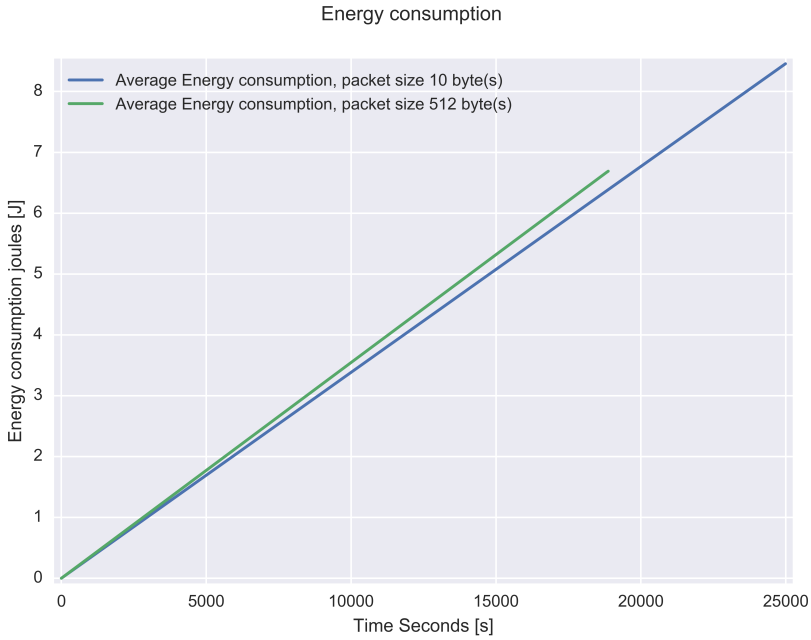


Figure 14.5: Total energy consumption of both packet sizes, after correction

consumption increases with time. This is according with the results obtained in chapter 13 figure 13.11.

Table 14.1 summarizes key energy consumption values from both tests (chapter 13 and 14 combined). As an end result the table shows the average energy consumption per single packet sent in both tests with both packet sizes. The result is showing that the average energy consumption is lower in the long test compared with the short test. This applies to both packet sizes. If we combine this information with the observations made earlier in this section (14.4), the result indicate that the short test is either affected by the power spikes in the start of the test or the length of the test.

Chapter 15

Zero packetsize test

This test is measuring the energy consumption while no packets is sent. The reason for doing this is to get an overview of how much energy the system consumes without utilizing the radio module.

15.1 Implementation

In order to have a test comparable to the test described in chapter 13 this test implement the same state machine. The major difference is that the actual sending of UDP packets is commented out. In this way the test will do the same tasks except the radio will not be sending UDP packets.

15.2 Execution

This test was conducted in two rounds with 100 imaginary "packets" sent. In this way the state machine will behave in the same way as the tests in chapter 13 and 14.

The following state machine parameters was used:

Packet size increments:	1	Byte
Packet size end:	2	Bytes
Packet size send count:	100	
Send interval:	1	Second

The execution time of the test was $1\ Second * 100 * 2 = 200\ Seconds$, this is $\approx 3,34$ Minutes execution time.

15.3 Results

The result is presented in the figures 15.1, 15.2 and 15.3. Figure 15.3 does plot the total energy consumption from this zero packet test combined with all packet sizes from the second test round in chapter 13. The zero packet test is plotted in red and the 640 different packet sizes from chapter 13 is plotted in blue. Some key numbers is found in table 15.1.

	Total energy used after 100 seconds (mJ):	Note:
Zero packet test:	≈ 30.6	
Packet test from chapter 13	≈ 33.5	Minimum
	≈ 37.5	Maximum

Table 15.1: Key numbers from figure 15.3

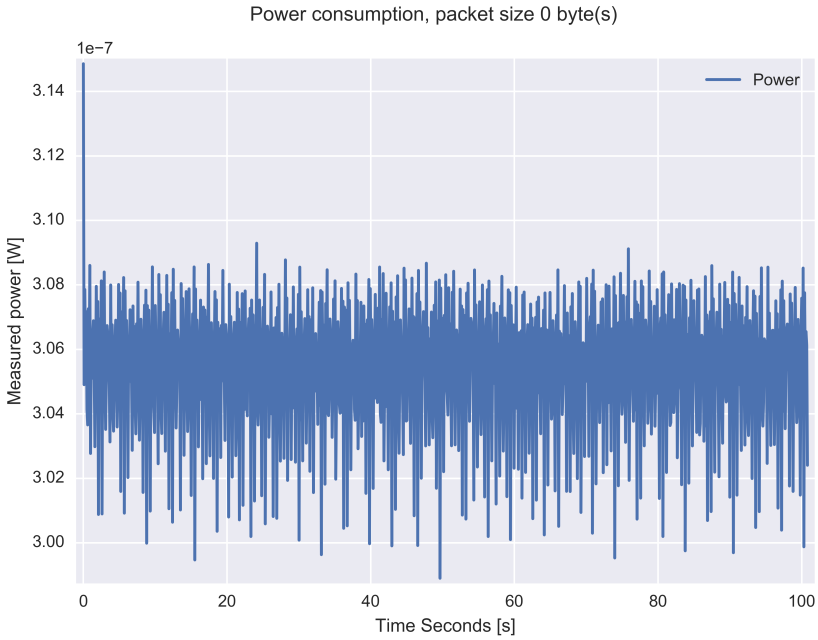


Figure 15.1: Power consumption during test with zero packet size, after correction

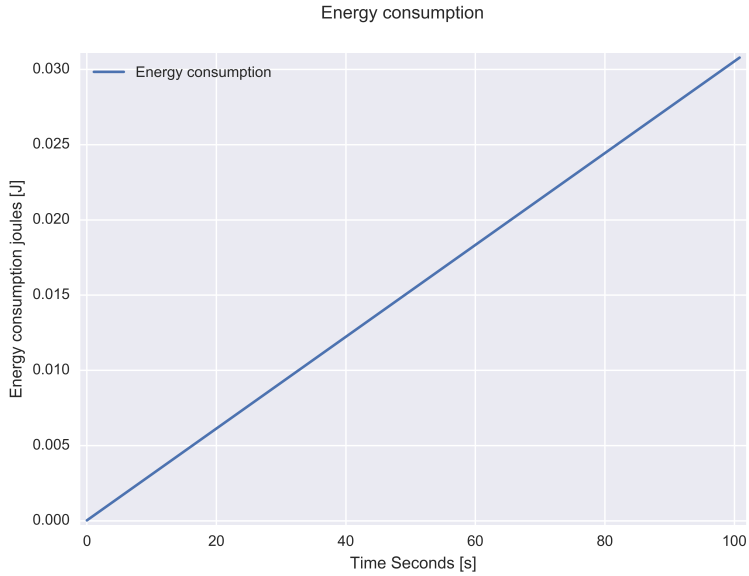


Figure 15.2: Total energy consumption of zero packet size test, after correction

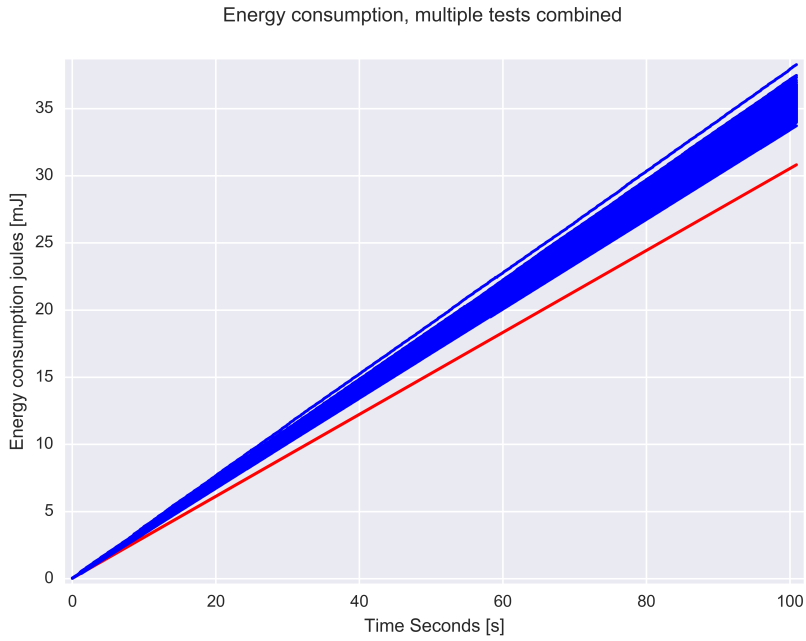


Figure 15.3: Total energy consumption of zero packet test (RED) combined with all packet sizes from second test round in chapter 13 (BLUE), all data is after correction

15.4 Analysis

Figure 15.1 shows the power consumption throughout the test. From an visual approximation the plot shows that the energy consumption is averaging around $\approx 0.305\ \mu W$ constantly through the test.

The total energy consumption is illustrated in figure 15.2. The interesting here is to see how the energy consumption is compared to the other tests. Therefore in figure 15.3 the data from the second test round has been combined with the data from figure 15.2.

As we can observe in figure 15.3 and from table 15.1 the total energy consumption in the zero packet test is lower compared with the test sending UDP packets. The gap between red and blue in figure 15.3 shows that there is a minimum "penalty" when sending packets of $\approx 2.9\ mJ$ compared to a "zero" packet. In percentage this corresponds to $\approx 10\%$ difference in energy consumption with the described test parameters and 100 seconds test length.

Chapter 16

Test Summary

This chapter will present a brief summary of the results presented in chapter 13, 14 and 15.

From the packet size tests in chapter 13 we observed that big power consumption spikes camouflaged the difference between the various packet sizes. After the result was corrected with an algorithm, the difference in energy consumption emerged. Figure 13.11 illustrates the final result from the chapter. With the applied EMA, we also observed the energy consumption trends.

In chapter 14 the long term power and energy consumption was presented. We observed that the power consumption has a periodic cycle of ≈ 200 seconds. We did observe that the short test from chapter 13 does have similar power characteristics, but the test is too short to cover a full periodic cycle. The energy consumption between the different packet sizes was consistent between the short and long test. But the short test has a slightly increased average value. In analysis this indicated that the short test is either affected by the power spikes or the length of the test.

The results from the zero packet test in chapter 15 did show that the power consumption was stable throughout the test. The energy consumption is approximately 10% lower compared with the smallest packet size from the second test in chapter 13. Figure 15.3 illustrated this well. The key values was also presented in table 15.1.

Part V

Discussion & Conclusion

Chapter 17

Discussion

This chapter is divided in two separate parts. The test setup, test execution and analysis is disused in the first section (17.1). Thereafter the custom measurement system is discussed in section 17.2.

17.1 Test system and testing

The wireless sensor network as described in chapter 4 was successfully established and did work as a test bench throughout the testing. The setup with Coktiki did work, but the toolchain is quite cumbersome with regards to flashing the target. The current setup relies on two different operating systems. And while developing on a such setup, the switching between the operating systems disrupts the work flow and consumes a decent amount of time.

A measurement setup was established to measure both energy consumption and network usage. The CMS is discussed below in section 17.2. The network measurement did work as expected and did discover several key findings. But the implemented network measurement could not measure what's actually sent in the air, as described in section 6.2. With an added wireless packet sniffer the network measurement can observe what happens in the WSN directly. With such capabilities, the measurement could reveal more details from the testing compared to the used solution.

The big power spikes observed in the power consumption plots before correction (for example figure 13.3) is assumed to be some kind of "housekeeping" that Contiki OS does. As earlier stated it camouflages the energy difference measured in this thesis. This also imply that the energy consumption conserved by those spikes have a much more significance in term of the total energy consumption compared to the different packet sizes.

In the test presented in chapter 13 the results did show a difference in energy consumption over the various packet sizes. This result was expected and the trends indicated by the EMA suggests that the system does actually consume energy with

respect to blocks of 100 Bytes. This is probably related to the actual packet size that is transferred wirelessly as the radio consumes more energy when more packets is transferred.

The longer test in chapter 14 does indicate that the short test (ch. 13) was actually too short to be in accordance with the *Law of Large Numbers* (sec. 2.4). This means that the test results from chapter 13 only could be used as an indication and not as a final result. On the other hand it should be noted that the results in chapter 14 doesn't differ so much from the results in chapter 13, and this reinforces the indication.

In the zero packet test (ch. 15), the power consumption was stable throughout the test. This seems wrong. The expected behaviour was that the system should have slept periodically between the processing. Therefore the presented result may be much higher than what's possible to achieve. If so, this also means that the difference in energy consumption between the zero packet and regular radio packets is potentially larger.

In figure 15.3 the comparison clearly shows that the potential energy savings between the "zero packet" and radio packets is $\approx 10\%$ in a test that have a duration of 100 Seconds. Despite that this result only could be regarded as an indication, this indication does show that there is a potential for the store and forward method. The potential savings at longer duration is even bigger than the presented indication. One of the key reasons is that the energy difference between the various packet sizes is equal or smaller than the difference to the "zero packet". In other words by not using the energy consumption of a small packet X times, we instead use the energy consumption to the "zero packet" X times, and only use the energy consumption to a larger packet one time! In this way the total energy consumption could be reduced.

Further work To be able to actually check how the radio operates while testing, the packet sniffer should be fixed. It's an important and invaluable opportunity to actually "see" what's sent in the air during a test and therefore this should be prioritized. For example during the zero packet test we could ensure that no packets is transmitted by the radio module.

The zero packet test had stable power consumption throughout the conducted test. This should be investigated further to clarify the explanation and look at possible improvements for power consumption in such system states.

As discussed before, the test in chapter 13 was to short. Therefore the test should be repeated with a longer duration on each packet size. In that way the results will be more accurate and in accordance to *Law of Large Numbers*(sec. 2.4).

The big power spikes in the power consumption graphs should be investigated. The source of the spikes should be mapped and thereafter it should be examined how it will affect the total energy consumption relative to the store and forward method.

If there is positive indications after the already mentioned further work, the complete store and forward method should be implemented. The method's energy and network consumption should be benchmarked.

Testing with multiple WSN nodes should be carried out. When multiple nodes is added the potential for energy savings could theoretically be multiplied with several orders of magnitude. This is of course when you consider the total energy consumption throughout the whole WSN. This testing will give a final verdict on the possibility for this method of saving energy consumption in a WSN.

When the WSN is used in a real world scenario there are tradeoffs. When an ideal store and forward method is found, the adaptive logic should be implemented and examined. One key property that is especially important to map is how much the energy consumption increases when the adaptive implementation have been added.

17.2 CMS

The CMS was successfully implemented and used as a part of the tests conducted in this thesis. All minimum requirements from section 7.1 has been fulfilled in addition to multiple of the optional requirements.

The solution with automatic transfer of metadata between the SensorTag and CMS did work really well. The solution made it possible for the SensorTag to send a file name prefix before a measurement sequence. With this possibility, the tests was made as autonomous as possible and this saved a lot of manual work. In addition, the data on the SD card was already sorted and named in a systematic manner.

The implemented current measurement solution have proven to work very well. Despite some more effort, in the analysis software to correctly treat the dual current range, the solution was a fair compromise between hardware requirements and software processing. Getting the software to correctly transition between the ranges was a challenge at first but did actually work very well in the end.

The firmware was made as simple as possible to avoid complex bugs. Although some simple ”power on self tests” and simple analog front end tests was made. Surprisingly those self test came to use when several boards had hardware faults. This does demonstrate that even simple tests is a powerful tool to have in an embedded system like this.

Unfortunately, only one fully working CMS was made. As section 10.2 describes, the external assembly did not assemble the boards as expected. Therefore, numerous hours of work was spent on debugging and correcting all of the various malfunctions. And despite the effort, the debugging had to be stopped in order to have enough time for other important tasks of this thesis. One positive outcome despite the malfunctions is that the author has gained a lot more knowledge and experience on debugging, analysing and correcting hardware faults.

From the authors point of view the external assembly lab had three large problems regarding the delivery. One: They didn’t solder components appropriate (bad reflow?). Two: They didn’t have full control over the supply chain with the consequence of assembling wrong parts. Three: The quality control failed and none of the errors was detected. Later the author of this thesis have received an apology from the lab and was told that those boards should never have passed the internal

quality control.

Further work The backup battery solution for the RTC did work as planned but a minor drawback with this solution is that when the battery need to be changed we also lose the track of time. When powering the CMS from a bench power supply time is also lost when it's removed. Therefore the solution with only one battery should be reconsidered.

The current revision of the CMS could only be powered through the battery terminals. While debugging it could be necessary or convenient to power the CMS through an external power supply. In fact this was done several times while working with the current revision. To be able to feed power to the CMS, alligator clips was used. But the use of alligator clips is impractical and it's easy to accidentally short circuit parts of the system. Therefore an external power connector or header would be preferable in addition to the battery.

Chapter 18

Conclusion

In order to answer the problem description a test platform was created. The platform consists of hardware, software, identification of the energy consumption sources and planning of a test program. The implemented wireless sensor network is based on 6LoWPAN and Contiki OS. In order to measure network and energy consumption, a complete measurement setup was established. The network measurement was based on existing solutions. However, no satisfactory energy measurement solution was found. So is was designed and implemented a working custom measurement system. This system was tailored for energy measurement of small wireless sensor nodes. The presented version does have an interface for a Texas Instruments SensorTag module. The system is complete with hardware design files, firmware and analysis software. The intention was to build five physical measurement systems, but unfortunately due to external assembly problems only one physical system was working.

Despite the problems, there was conducted tests in accordance to the specified test program. Due to time limitations, all of the specified tests couldn't be conducted. Therefore the essential tests was prioritized. The result from the tests was an indication of possible energy savings with the store and forward method. The indication was ≈ 10 % in potential energy savings over a duration of 100 Seconds. The tests also discovered several unpredicted issues. A broad list of tasks for further work is presented together with thoughts of possible improvements. The analysis of the results indicated that the actual potential for energy savings could be many orders of magnitude higher than the presented results. In order to actually confirm the indications, further work must be conducted.

Bibliography

ARM Ltd., Jul. 2009. CoreSight Components - Technical Reference Manual, ARM DDI 0314h.
URL http://infocenter.arm.com/help/topic/com.arm.doc.ddi0314h/DDI0314H_coresight_components_trm.pdf

ARM Ltd., Jan. 2016. Cortex-M Debug Connectors.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.352.5877&rep=rep1&type=pdf

BeagleBoard.org Foundation, Feb. 2016. BeagleBoard.org - latest-images.
URL https://beagleboard.org/latest-images

CETIC asbl, Mar. 2016a. 6lbr Project on GitHub.
URL https://github.com/cetic/6lbr

CETIC asbl, Feb. 2016b. 6lbr Wiki - Home.
URL https://github.com/cetic/6lbr/wiki

Contiki Open Source Project, Feb. 2016. Contiki: The Open Source Operating System for the Internet of Things.
URL http://www.contiki-os.org/

Eclipse Mosquitto, 2016. An Open Source MQTT v3.1 Broker.
URL http://mosquitto.org/

George Oikonomou, Apr. 2016. Sensniff, Live Traffic Capture and Sniffer for IEEE 802.15.4 networks., on GitHub.
URL https://github.com/g-oikonomou/sensniff

Gunnar Ranøyen Homb, Dec. 2015. Low energy wireless sensor networks, Literature study. Project Report, Norwegian University of Science and Technology.

Horowitz, P., Hill, W., 2008. The art of electronics, second edition Edition. Cambridge University Press, New York, NY.

Intel Corporation, Apr. 2014. Switches — What are forwarding modes and how do they work?
URL https://web.archive.org/web/20140419110752/http://www.intel.com/support/express/switches/sb/cs-014410.htm

Lindh, J., Lee, C., Feb. 2015. Measuring Bluetooth Smart Power Consumption, SWRA478a.
URL http://www.ti.com/lit/an/swra478a/swra478a.pdf

Linear Technology, Jun. 2014. LTC6102/LTC6102hv DATASHEET , LT 0614 REV E.
URL http://cds.linear.com/docs/en/datasheet/6102fe.pdf

Maxim Integrated, Mar. 2015. DS3231 Extremely Accureate RTC, DATASHEET.
URL https://datasheets.maximintegrated.com/en/ds/DS3231.pdf

NXP Semiconductors, Apr. 2013. SD(HC)-memory card and MMC interface conditioning, AN10911.
URL http://www.nxp.com/documents/application_note/AN10911.pdf

Samtec, Inc., Aug. 2015. LSS Series Product Specification.
URL http://suddendocs.samtec.com/productspecs/lss.pdf

Shiryayev, A. N., 1992. On The Law of Large Numbers. In: Shiryayev, A. N. (Ed.), Selected Works of A. N. Kolmogorov. No. 26 in Mathematics and Its Applications (Soviet Series). Springer Netherlands, pp. 11–12, dOI: 10.1007/978-94-011-2260-3_2.
URL http://link.springer.com/chapter/10.1007/978-94-011-2260-3_2

Silicon Labs, Oct. 2013. EFM32gg-DK3750 USER MANUAL.
URL https://www.silabs.com/Support%20Documents/TechnicalDocs/efm32gg-dk3750-ug.pdf

Silicon Labs, Jun. 2014. EFM32lg332 DATASHEET.
URL https://www.silabs.com/Support%20Documents/TechnicalDocs/EFM32LG332.pdf

Silicon Labs, Jan. 2016. AN0002: Hardware Design Considerations.
URL http://www.silabs.com/Support%20Documents/TechnicalDocs/AN0002-efm32-and-ezr32-hardware-design-considerations.pdf

SourceForge, Aug. 2015. The Contiki Operating System - Browse /Instant Contiki/Instant Contiki 3.0.
URL https://sourceforge.net/projects/contiki/files/Instant%20Contiki/Instant%20Contiki%203.0/

Texas Instruments Incorporated, Apr. 2013. LM4128/LM4128q DATASHEET.
URL http://tec.icbuy.com/uploads/2012/10/9/lm4128.pdf

Texas Instruments Incorporated, Apr. 2014. CC Debugger User's Guide, SWRU197h.
URL http://www.ti.com/lit/ug/swru197h/swru197h.pdf

Texas Instruments Incorporated, Oct. 2015a. CC2650 SimpleLink Multistandard Wireless MCU Datasheet, SWRS158a.
URL http://www.ti.com/lit/ds/symlink/cc2650.pdf

Texas Instruments Incorporated, Oct. 2015b. LP3982 Ultra-Low-Dropout CMOS Regulator, DATASHEET.
URL http://www.ti.com.cn/cn/lit/ds/symlink/lp3982.pdf

Texas Instruments Incorporated, Mar. 2015c. Multi-Standard CC2650 SensorTag Design Guide.
URL http://www.ti.com/lit/ug/tidu862/tidu862.pdf

Texas Instruments Incorporated, Feb. 2016a. CC-DEVPACK-DEBUG - SimpleLink SensorTag Debugger DevPack.
URL http://www.ti.com/tool/cc-devpack-debug

Texas Instruments Incorporated, Mar. 2016b. Contiki-6lowpan - Texas Instruments Wiki.
URL http://processors.wiki.ti.com/index.php/Contiki-6LOWPAN

Texas Instruments Incorporated, Feb. 2016c. Simplelink SensorTag - TI.com.
URL http://www.ti.com/ww/en/wireless_connectivity/sensortag2015/?INTC=SensorTag&HQS=sensortag

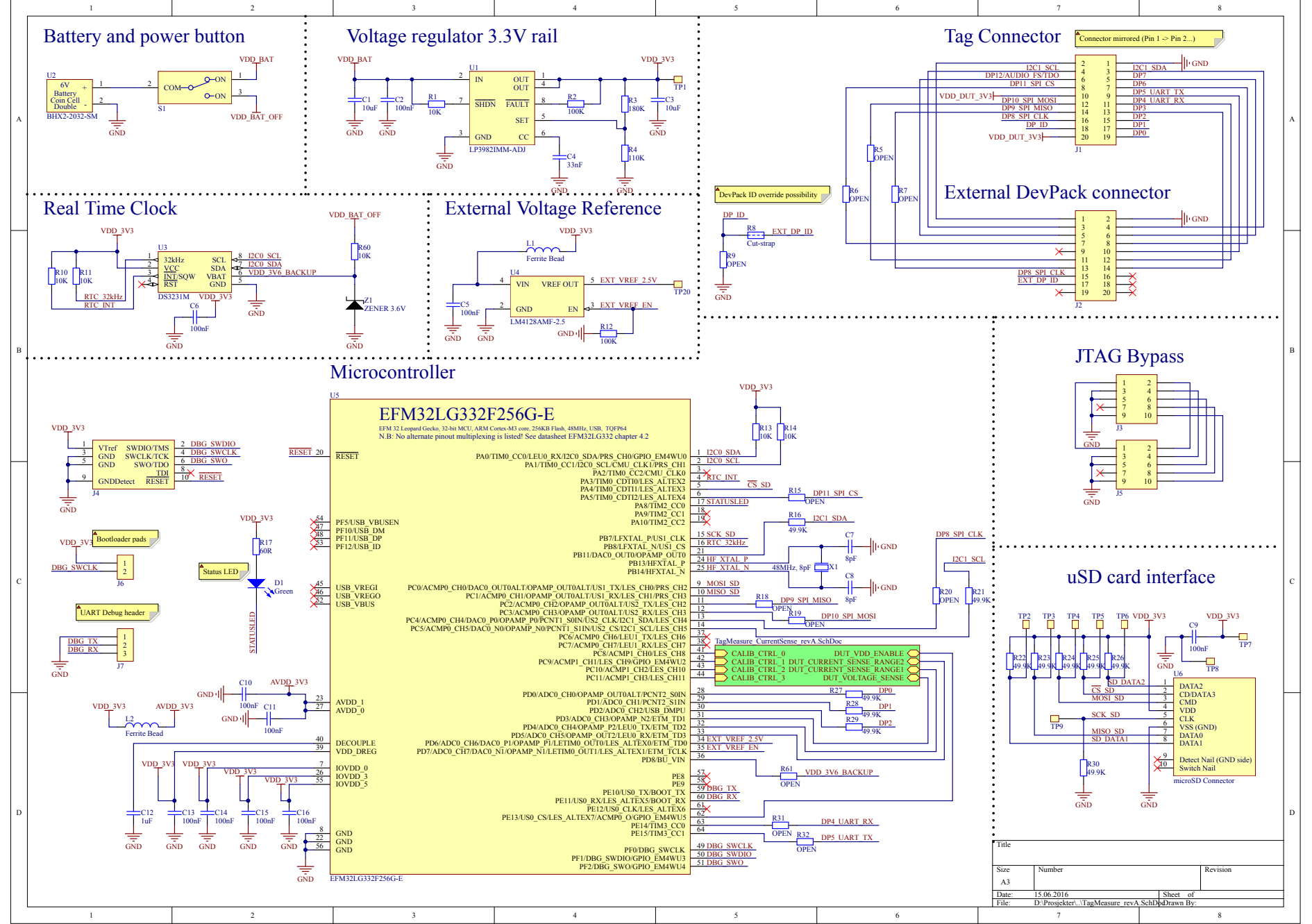
Part VI

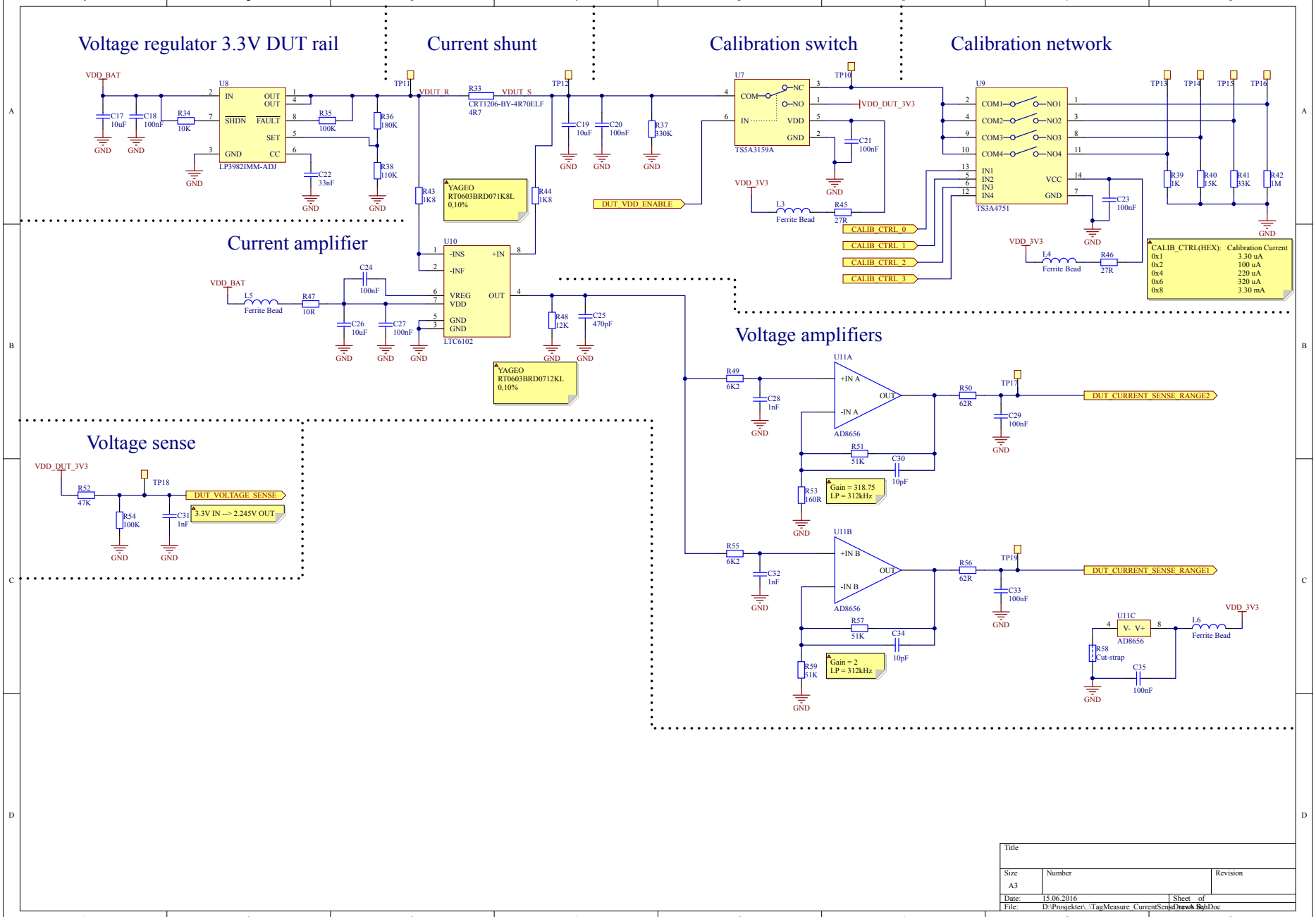
Appendix

Appendix A

CMS Schematic

This appendix consists of the CMS schematic. The schematic is included on the next two pages. The appendix is also available in the digital attachment folder, path within the folder: "CMS Hardware/revA/Project Outputs for sensornetworks-TagMeasure-hardware/ Schematic Print"

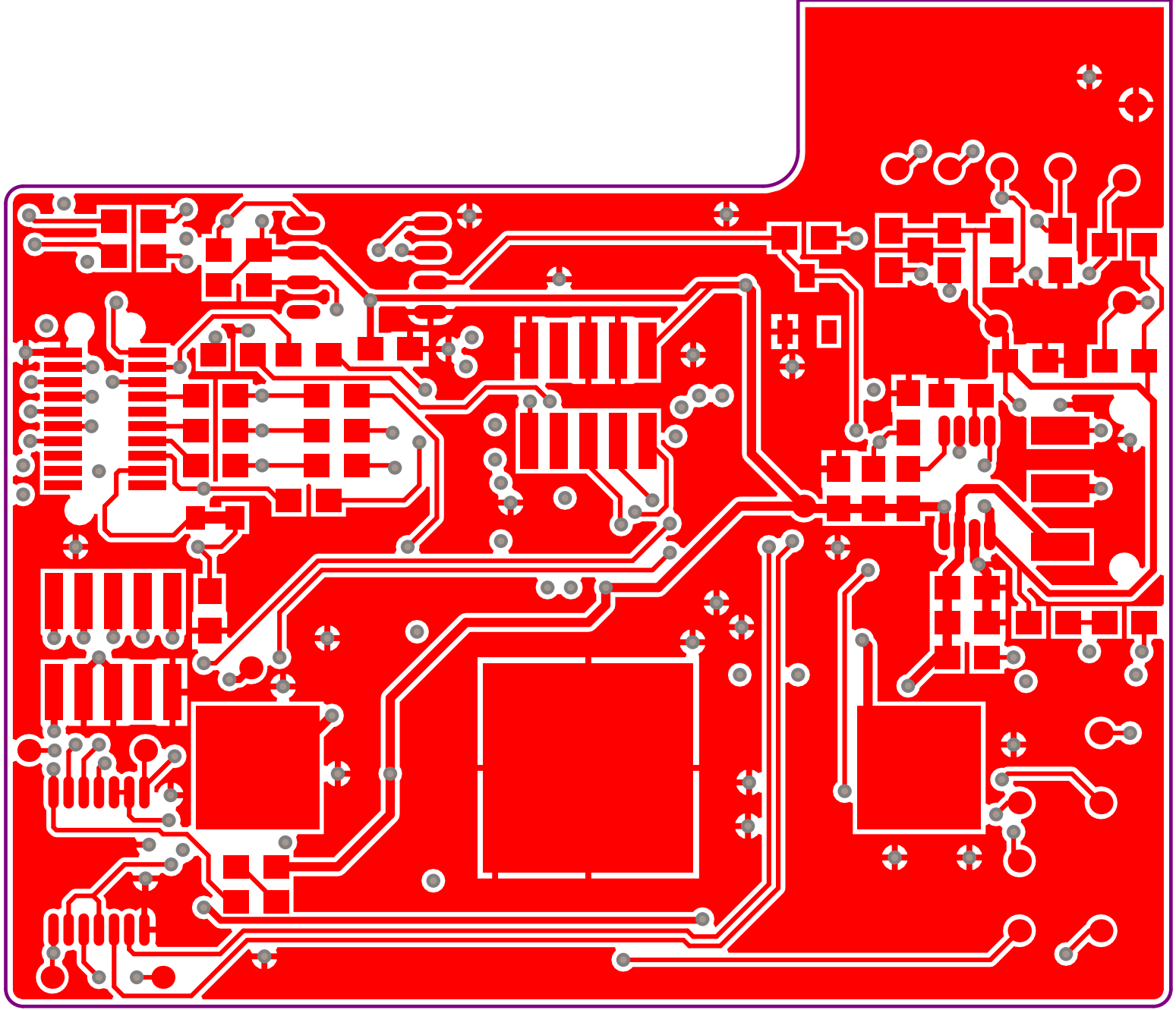

Battery and power button
Voltage regulator 3.3V rail
Tag Connector
Connector mirrored (Pin 1 -> Pin 2...)
Real Time Clock
External Voltage Reference
External DevPack connector
DevPack ID override possibility
Microcontroller
EFM32LG332F256G-E
EFM 32 Leopard Gecko, 32-bit MCU, ARM Cortex-M3 core, 256KB Flash, 48MHz, USB, TQFP64
N.B: No alternate pinout multiplexing is listed! See datasheet EFM32LG332 chapter 4.2
JTAG Bypass
uSD card interface
Bootloader pads
Status LED
UART Debug header
TagMeasure_CurrentSense_revA.SchDoc
microSD Connector
Title
Size
A3
Number
Revision
Date: 15.06.2016
Sheet of
File: D:\Prosjekter\..\TagMeasure_revA.SchDoc
Drawn By:

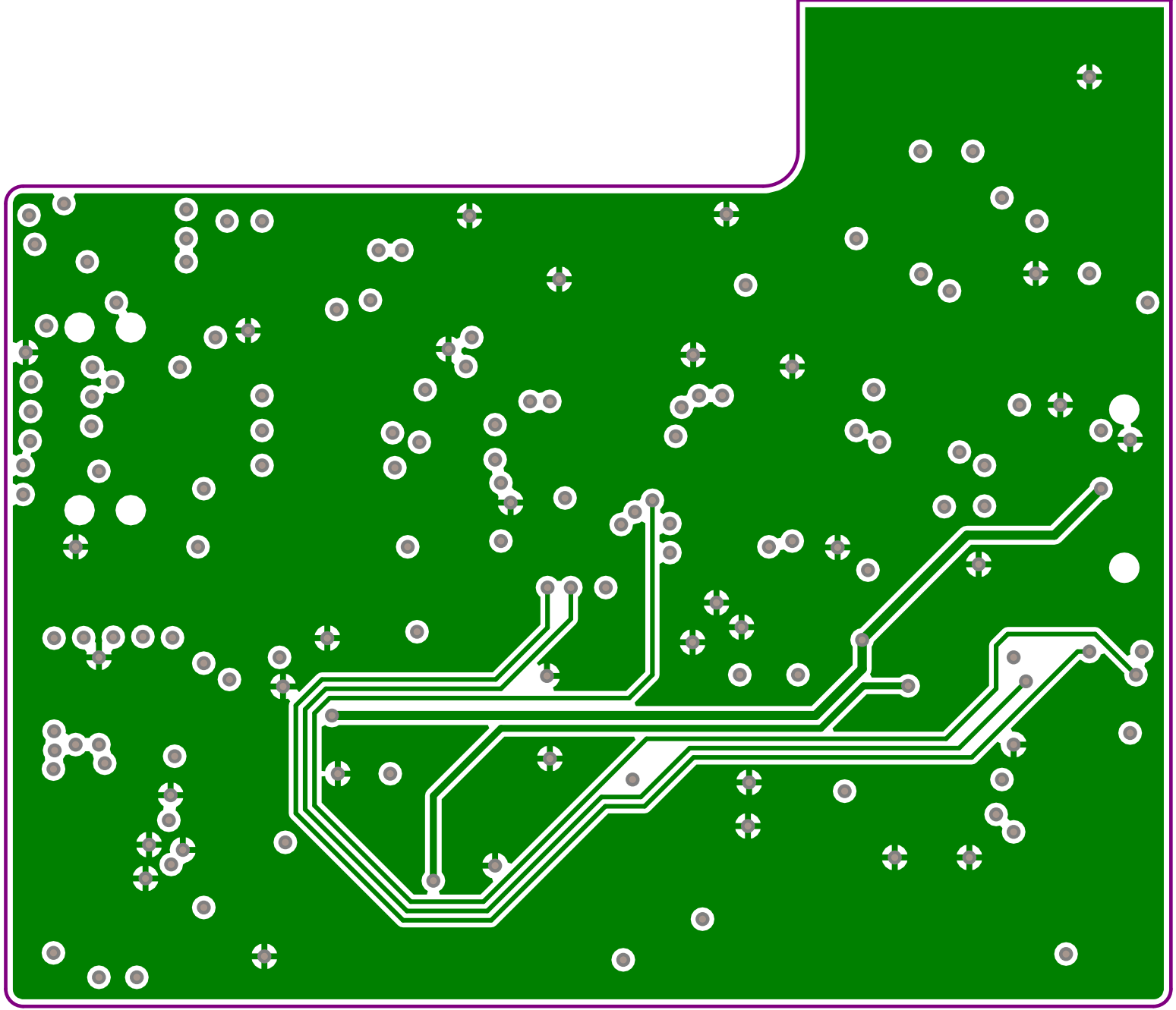

Voltage regulator 3.3V DUT rail
Current shunt
Calibration switch
Calibration network
Current amplifier
Voltage amplifiers
Voltage sense
VDD_BAT
U8
LP3982IMM-ADJ
TP11
VDUT_R
R33
CRT1206-BY-4R70ELF
4R7
VDUT_S
TP12
U7
TS5A3159A
TP10
VDD_DUT_3V3
DUT_VDD_ENABLE
VDD_3V3
L3
Ferrite Bead
U9
TS3A4751
TP13
TP14
TP15
TP16
CALIB_CTRL_0
CALIB_CTRL_1
CALIB_CTRL_2
CALIB_CTRL_3
CALIB_CTRL(HEX): Calibration Current
0x1 3.30 uA
0x2 100 uA
0x4 220 uA
0x6 320 uA
0x8 3.30 mA
YAGEO
RT0603BRD071K8L
0,10%
U10
LTC6102
YAGEO
RT0603BRD0712KL
0,10%
U11A
U11B
U11C
AD8656
DUT_CURRENT_SENSE_RANGE2
DUT_CURRENT_SENSE_RANGE1
Gain = 318.75
LP = 312kHz
Gain = 2
LP = 312kHz
DUT_VOLTAGE_SENSE
3.3V IN --> 2.245V OUT
GND
Title
Size
A3
Number
Revision
Date: 15.06.2016
Sheet of
File: D:\Prosjekter\..\TagMeasure_CurrentSense.SchDoc
Drawn By:

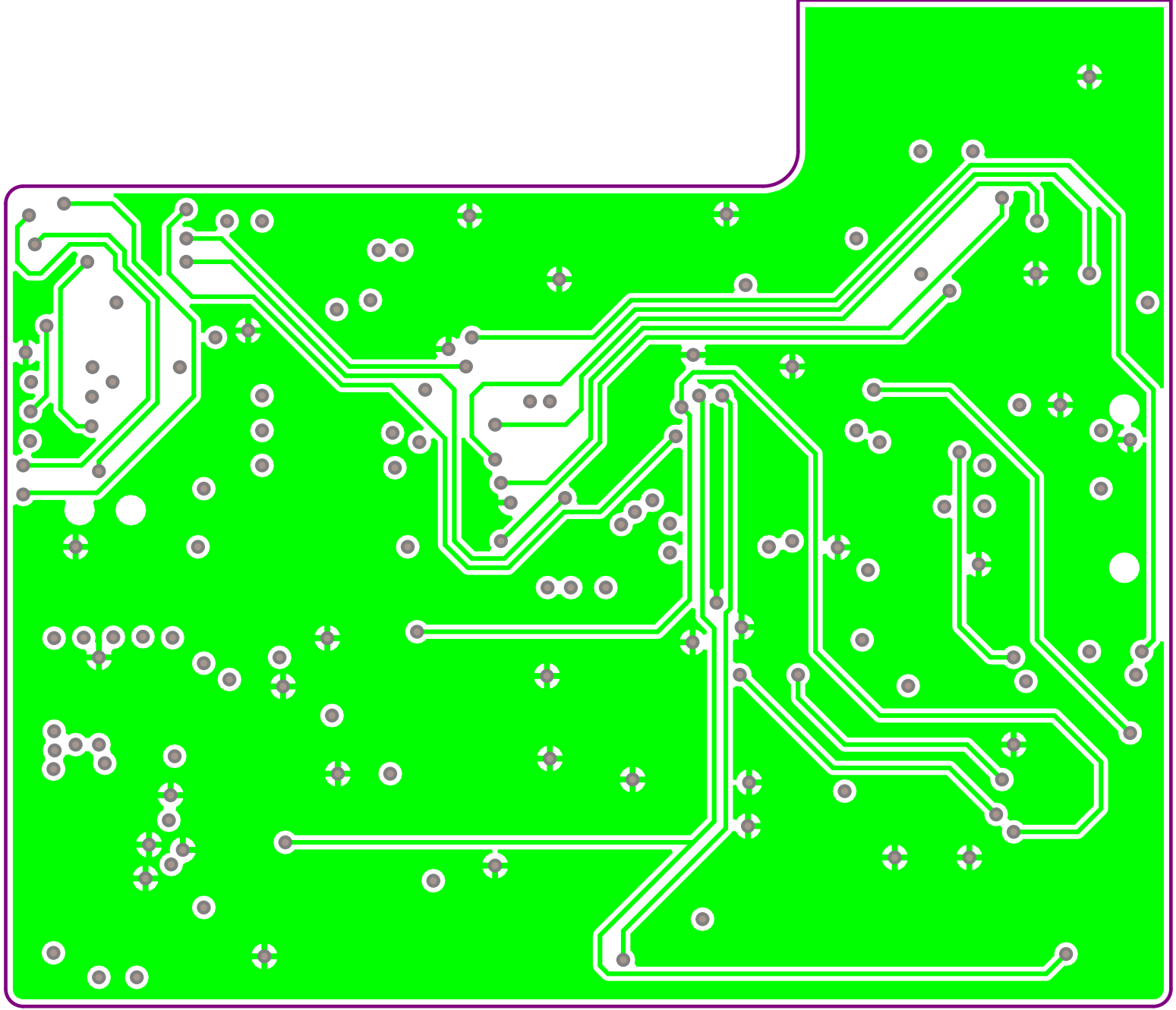
Appendix B

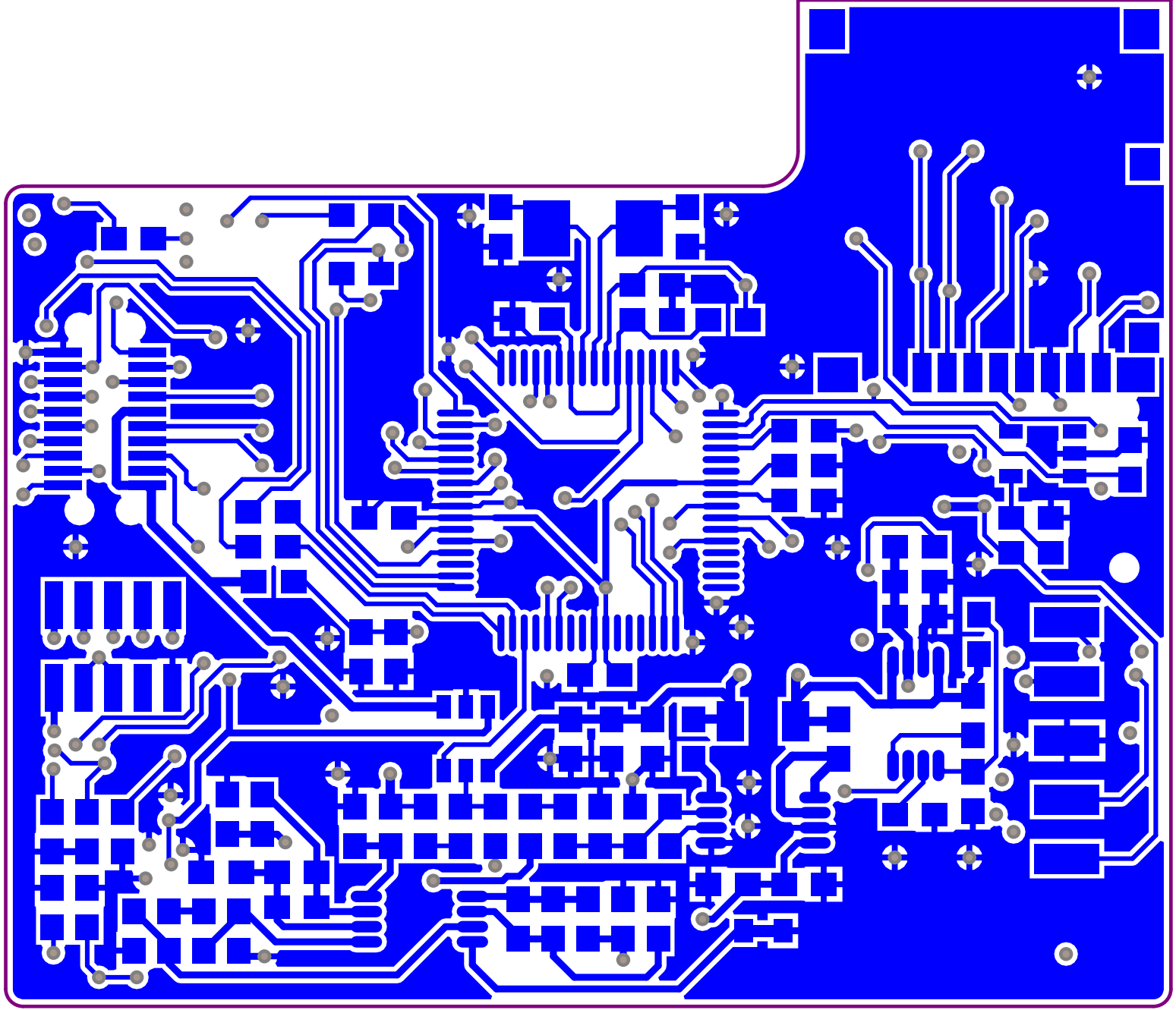
CMS PCB layout, 3D Model & Design files

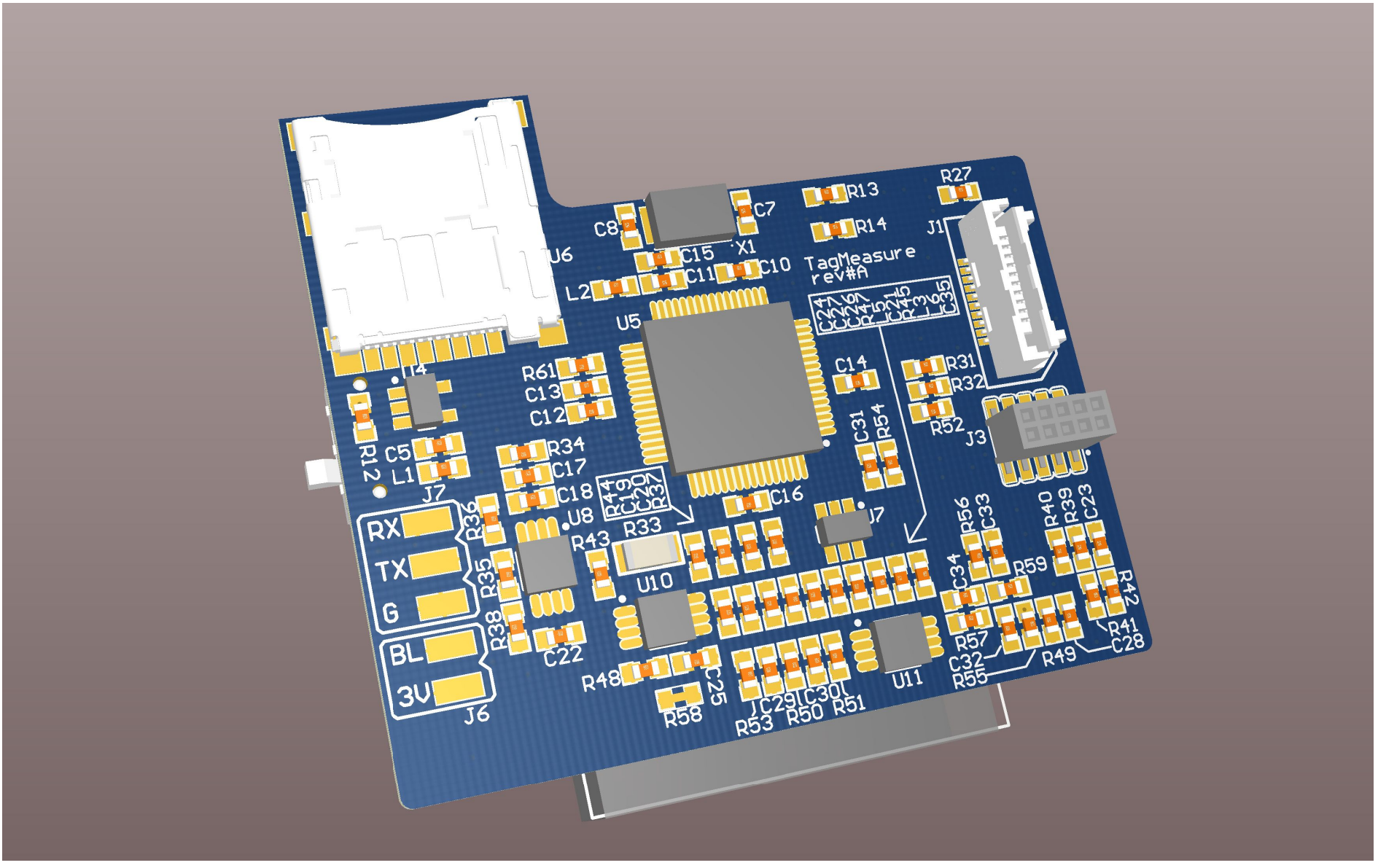
This appendix consists of the PCB layout, 3D Model render and the hardware design files. The PCB layout and 3D Model is included on the next five pages. The hardware design files is only found in the digital attachment folder. The complete appendix is available in the digital attachment folder, path within the folder: ”CMS Hardware/”










TagMeasure
rev#A
RX
TX
G
BL
3V
J6
J7
J1
J3
U5
U6
U8
U10
U11
X1
L1
L2
C5
C7
C8
C10
C11
C12
C13
C14
C15
C16
C17
C18
C22
C25
C28
C31
C32
C33
C34
R12
R13
R14
R27
R31
R32
R33
R34
R35
R36
R38
R39
R40
R41
R42
R43
R44
R48
R49
R50
R51
R52
R53
R54
R55
R56
R57
R58
R59
R61

Appendix C

CMS Firmware source code

This appendix consists of the source code for the firmware developed for the CMS. The appendix is available in the digital attachment folder, path within the folder: ”CMS Firmware/”

Appendix D

CMS Analyzer software

This appendix consists of the source code for the the analyzer software. The appendix is available in the digital attachment folder, path within the folder: ”CMS Analyzer Software/”

Appendix E

SensorTag Firmware source code

This appendix consists of the complete firmware used on the SensorTag. This includes full version of Contiki together with the developed application firmware. The appendix is available in the digital attachment folder, path within the folder: ”SensorTag Firmware/”

Appendix F

Border Router configuration & Slip-Radio Firmware

This appendix consists of the configuration file used for the 6LBR Border Router and the firmware binary flashed to the CC2531 Slip-radio. The appendix is available in the digital attachment folder, path within the folder: "Border Router/"

Appendix G

RAW Measurement data

This appendix consists of the RAW measurement data from all of the conducted tests. The appendix is available in the digital attachment folder, path within the folder: ”RAW Result Data/”

Appendix H

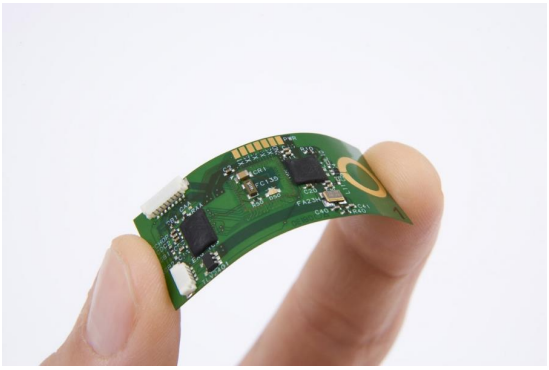
Original assignment

The given assignment paper for this report is included on the next page. This document is also available in the digital attachment folder, path within the folder: ”Original Assignment/”

Adaptive Store and Forward

A possible solution to the increasing resource consumption in wireless sensor networks?

Project Background



Emerging use of sensors on new areas and connectivity to the internet introduces new challenges.

With an increasing amount of sensors coupled with a rapid development of factors such as sensor speed and resolution, rising resource consumption is an unavoidable side effect.

Therefore, solutions to handle this increasing resource demand is important. This resource demand is both relevant when looking at power consumption, but can also be a challenge for the total network capacity.

One possible solution to solve the resource demands is implementation of new data handling methods. The traditional way is to send all the gathered data upwards to a processing server. This scheme is static and does not reflect the flexible nature of a wireless sensor network.

By implementing more adaptive solutions to decide whether data should be sent upwards or stored locally for later use, savings could possibly be achieved, in both power consumption as well as network usage.

Topic descriptions

First, it should be examined whether and how an “adaptive store and forward” solution could improve a sensor network.

Thereafter, several factors should take place in the thesis:

- **Creation of a test system including appropriate hardware to survey the topics of this thesis**
- **Data-gathering from a traditional data-harvesting-method to establish a baseline for the challenge being looked at**
- **Development of “store and forward” –filter**
- **Benchmark the store and forward method with respect to:**
  - **Energy consumption**
  - **Network usage**
- **Comparison and analysis of the test results from before and after the implementation of developed methods**
- **Investigate opportunities for improvements**