



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich



Institut für  
Technische Informatik und  
Kommunikationsnetze

Master Thesis  
at the Department of Information Technology  
and Electrical Engineering

# Distributed Time Pulse for FlockLab

Balz Maag

**Advisors:** Roman Lim  
**Professor:** Prof. Dr. Lothar Thiele

August 29, 2014



# Abstract

This thesis presents a wireless synchronization protocol, which is able to disseminate a pulse signal through a network with an accuracy in the sub- $\mu s$  range. The protocol is based on Glossy, a flooding architecture for wireless networks, which allows a packet to be transmitted through the whole network in a highly reliable and fast manner. Various techniques are used to enhance Glossy's implicit synchronization process and compensate for non-deterministic timing behaviours. The final design distributes a pulse-per-second (PPS) of a single GPS device and is thereby able to synchronize the observers in the FlockLab network with an average synchronization error below 100 *ns*.

A recently developed enhancement for FlockLab, the data acquisition unit (DAQ), improves the current limitations on event handling of the FlockLab services. Further, it features a timing module, which allows to apply accurate timestamps to events. The internal clock of the DAQ is synchronized to an external PPS. This thesis develops a hardware implementation of the DAQ, the FlockDAQ board, which is integrated into the current FlockLab setup. The board combines a Spartan-6 FPGA and an SRAM memory with a CC430 system-on-chip with integrated RF transceiver, which runs the synchronization protocol and provides the DAQ with an accurate PPS. The final design of the FlockDAQ in combination with the synchronization protocol is able to trace a highly accurate GPS PPS with an average synchronization error of 163 *ns*.



# Acknowledgements

I would like to thank my supervisor Roman Lim for his support and guidance during this thesis and Prof. Dr. Lothar Thiele and the Computer Engineering and Networks Laboratory TIK for giving me the opportunity to write this thesis and providing the facilities and equipment.

Special thanks also go to Felix Sutton for his helpful assistance during the hardware design task.

Further, I would like to thank my fellow student Benjamin Dissler for the time we spent in our office.



# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>i</b>
<b>1. Introduction</b>	<b>1</b>
<b>2. Related Work</b>	<b>5</b>
<b>3. Synchronization Protocol</b>	<b>7</b>
3.1. Glossy . . . . .	7
3.2. Synchronization Protocol . . . . .	11
3.3. GPS pulse dissemination in FlockLab . . . . .	17
<b>4. FlockDAQ Board</b>	<b>19</b>
4.1. Requirements . . . . .	19
4.2. Hardware Components . . . . .	22
4.3. Final Design . . . . .	26
<b>5. Evaluation</b>	<b>29</b>
5.1. Setup . . . . .	29
5.2. Synchronization Accuracy . . . . .	32
5.3. FlockDAQ Board Performance . . . . .	40
<b>6. Conclusion and Outlook</b>	<b>45</b>
6.1. Conclusion . . . . .	45
6.2. Outlook . . . . .	47
<b>A. Appendix</b>	<b>49</b>
A.1. Operation Manual . . . . .	49
A.2. Plots . . . . .	55
A.3. PCB Schematics . . . . .	59
<b>Bibliography</b>	<b>71</b>





# 1

## Introduction

Testbeds are a powerful tool in the development of wireless embedded systems. They are used for debugging, testing and evaluating applications by conducting controlled tests. Conventional debug tools like logic analysers, power analysers or serial data loggers are expensive and limited in their application for large-scale networks. A testbed overcomes these limitations and combines such services in a distributed context.

FlockLab [9] is a wireless sensor nodes (WSN) testbed at ETH Zürich. It offers the facilities and tools for elaborate testing in a sensor network. The FlockLab network employs 30 distributed nodes, each of them is represented as an observer-target pair. The observer is a hardware unit consisting of an embedded Gumstix computer and a FlockBoard, which can control up to four different target nodes. The target is the actual sensor node, currently employed are for instance the Tmote Sky platform from Moteiv or the MSP430-CCRF from Olimex. The observers are connected to a back-end server, which coordinates distributed tests and collects and stores the test results.

During a FlockLab test an observer can monitor one target per observer and offers data collection for various services:

- **GPIO tracing:** The observer can capture GPIO signal changes of up to five target pins.
- **GPIO actuation:** Three GPIO target pins can be controlled (set, clear, toggle) by the observer at either predefined times or in periodic time intervals.

- **Power profiling:** An analog-digital converter (ADC) on the FlockBoard measures the current consumption of the target. The service can be turned on during predefined time windows.
- **Serial I/O:** The observer can communicate (transmit and receive data) over the serial port of the target.

These four services can run simultaneously on all the observers.

### Motivation

In order to provide meaningful test results, the distributed measurements need to be synchronized to a global time. The current solution is the operation of a network time protocol (NTP) server. The observers synchronize their time to the server over Ethernet or Wi-Fi. The average and maximum pairwise error of timestamped events is shown in table 1.1 (according to [9, Table 2]). This error is mainly due to non-deterministic

	average	maximum
only Ethernet	36 $\mu s$	394 $\mu s$
Wi-Fi included	166 $\mu s$	1170 $\mu s$

Tab. 1.1.: Error of time-stamped FlockLab events

NTP packet delays, which are higher, when transmitted over Wi-Fi than Ethernet. Therefore FlockLab is not suited for tests, which require a synchronization accuracy in the small  $\mu s$  range or below.

For example Glossy [3], a flooding protocol for wireless networks, allows multiple nodes to concurrently transmit packets. The time when these packets are received at listening nodes varies in the sub- $\mu s$  range. To capture timing information about the distributed communication sequence, the FlockLab observers need to be synchronized with an accuracy in the sub- $\mu s$  range as well.

The work in [2] presents an enhancement for FlockLab, called the data acquisition unit (DAQ). The current setup limits the rate of events which can be detected by the Gumstix due to the interrupt handling. The DAQ overcomes this limitation and provides a higher detection rate (10 MHz). In addition it features a time calibration module, which is synchronized to a GPS pulse-per-second (PPS) and therefore provides accurate timestamps with high resolution. The prototype of the DAQ uses a LEA-6T GPS device as PPS input. These devices can generate a timepulse with an accuracy [20, Figure 2] below 15  $ns$ . Due to the limited GPS signal availability for the indoor observers and the high acquisition cost for the whole network, a wireless protocol is needed, which distributes a PPS from a single GPS device to all the observers in the network. Because the DAQ

requires a highly accurate timepulse to synchronize its internal clock, the distributed timepulse must be emitted on all observers with high accuracy and reliability.

Further, the existing prototype needs to be integrated into the current FlockLab setup. A hardware solution is needed, which combines all the necessary hardware modules for a successful implementation of the DAQ.

### **Challenges**

The goal is a synchronization protocol, which is able to synchronize all the observers in FlockLab with an accuracy in the sub- $\mu s$  range. It must be able to disseminate a time pulse through the whole network in a reliable and fast manner. The hardware implementation of the DAQ requires multiple interconnected integrated circuits (IC) combined on a compact printed circuit board (PCB), which can be integrated into the existing FlockLab setup.

### **Contribution**

The thesis designs a synchronization protocol which is able to disseminate a timepulse through a wireless network in a highly accurate and fast manner. The protocol enhances Glossy, a wireless network flooding architecture, with various adaptations of the implicit synchronization algorithm. The improvements consist of rounding divisions appropriately, considering packet transmission delays and suppressing significant reference time outliers. The final design is able to calculate a globally synchronized reference time and simultaneously trigger a pulse on all FlockLab observers. The protocol is finally used to disseminate a highly accurate PPS by a single GPS receiver and is able to synchronize the observers with an average synchronization error below 100 ns.

Further, the project designs a hardware implementation of the DAQ, the FlockDAQ board. The board features all required hardware modules of the DAQ and to run the synchronization protocol, but pays also special attention to a successful integration into the existing FlockLab setup. The FlockDAQ board is placed between the FlockLab observer and the Gumstix, where the DAQ enhances the current limitations on the event capturing and uses the synchronization protocol to apply synchronized timestamps to the events. The hardware units are controlled and programmed through a single USB to serial converter and use a specific power supply to power the FlockDAQ from the FlockBoard.

## Outline

The remainder of this thesis is structured as follows:

Chapter 3 describes the implementation of the synchronization protocol. The basis of Glossy and its limitations are described and the improvements by the protocol are elaborated as well as how a PPS by a single GPS device can be distributed through the FlockLab network.

In chapter 4 the hardware implementation of the DAQ, the FlockDAQ, is presented with emphasis on the requirements for an integration into the existing FlockLab setup.

The work concludes in chapter 5 with an extensive evaluation of the synchronization protocol for the FlockLab network by measuring the synchronization error of single observers as well as the performance of the FlockDAQ board.

# 2

## Related Work

Clock synchronization is an eminent task in networks consisting of multiple nodes, both wired and wireless. A global common time available on each individual node is required for a consistent operation of various applications like distributed sensing, communication or network controlling.

### **Wired Networks**

A popular and widely used protocol is the Network Time Protocol (NTP) [11]. Network nodes synchronize their time to an NTP server by exchanging timestamped packets. These timestamps are used to calculate the round-trip delay and offset. In order to filter outliers and improve the synchronization accuracy, NTP uses an external PPS signal to adjust the system clock.

### **Global Positioning System (GPS)**

GPS devices are highly accurate PPS signal generators. They periodically generate a 1 Hz signal, which is synchronized to the atomic clocks of the GPS satellites. The accuracy of that PPS is normally around a few nano-seconds and therefore provides an ideal synchronization source, e.g. the u-blox LEA-6T [21] [20]. Therefore GPS is often used for network synchronization, for instance in cellular networks. Base stations of the global system for mobile communications (GSM) are using PPS signals by GPS receivers to synchronize the communication data [5]. However, due to limited GPS signal availability for indoor deployments

and high acquisition costs for large networks, GPS PPS signals need to be distributed through the network in an efficient and accurate way.

### Wireless Networks

The Flooding Time Synchronization Protocol (FTSP) [10] is a synchronization protocol developed for wireless networks. It uses radio-driven message flooding by a dedicated root node to multiple clients. The packet contains the sender's timestamp at the start of its transmission and is locally timestamped at reception by each listener. In order to synchronize each node to the root's global time, the protocol uses multiple techniques and statistical tools to reduce non-deterministic timing metrics and to compensate for clock drift. FTSP achieves a synchronization accuracy within the  $\mu s$  range. Glossy [3] is a wireless flooding protocol, which is able to distribute a radio-packet through a network within a few milliseconds and with a reliability above 99.99%. This is done by exploiting constructive interference of overlapping packets, which enables their concurrent transmission. Due to a highly deterministic timing behaviour of the flooding sequence, Glossy is able to synchronize nodes with high accuracy. In fact, Glossy achieves a significant higher synchronization accuracy and a smaller flooding latency than FTSP.

This work presents a synchronization protocol based on Glossy, which disseminates a PPS signal, generated by a single GPS device, through a wireless network. By exploiting Glossy's timing behaviour, the presented protocol compensates for clock drift, packet propagation delay and measurement inaccuracies. Hence, the distributed PPS signal achieves a synchronization accuracy in the sub- $\mu s$  range.

# 3

## Synchronization Protocol

This chapter presents a synchronization protocol for FlockLab. Section 3.1 describes Glossy, the basis of the protocol, and section 3.2 three improvements to the synchronization algorithm. In section 3.3 an overview is given, how a GPS time pulse can be disseminated through the FlockLab network based on the details in the previous sections.

### 3.1. Glossy

The basis for the synchronization protocol is a flooding architecture for wireless sensor networks, called Glossy [3], which provides wireless communication for a multi-hop network in a highly reliable and fast manner. Glossy is able to achieve a flooding reliability over 99.99% by exploiting constructive interference of broadcast packets, i.e. concurrently transmitted signals can still be detected with high probability under certain conditions. This effect usually occurs, when multiple signals with the same frequency overlap in time and space and their temporal displacement does not exceed an upper bound. For example, the IEEE 802.15.4 [1] modulation scheme requires a temporal displacement of at most  $0.5 \mu s$ . More details on constructive interference can be found in [3].

#### Glossy Flood

In order to meet the requirement for the temporal displacement and therefore to achieve a high flooding reliability, Glossy follows a specific

communication pattern. A network usually consists of multiple nodes, where one is assigned to be the initiator (or root) node and the rest are receivers. Fig. 3.1 shows a typical flood through a network with 4 nodes. A flood is started by a first packet transmission (TX) from the initiator node. Any nearby receiver will capture the packet and immediately retransmits it after successful reception (RX). Receivers, which are 2 hops away from the initiator node, capture the relayed packet and follow the same procedure. This is done until all nodes transmitted a packet for a predefined number of times  $N$ . The immediate retransmission of the

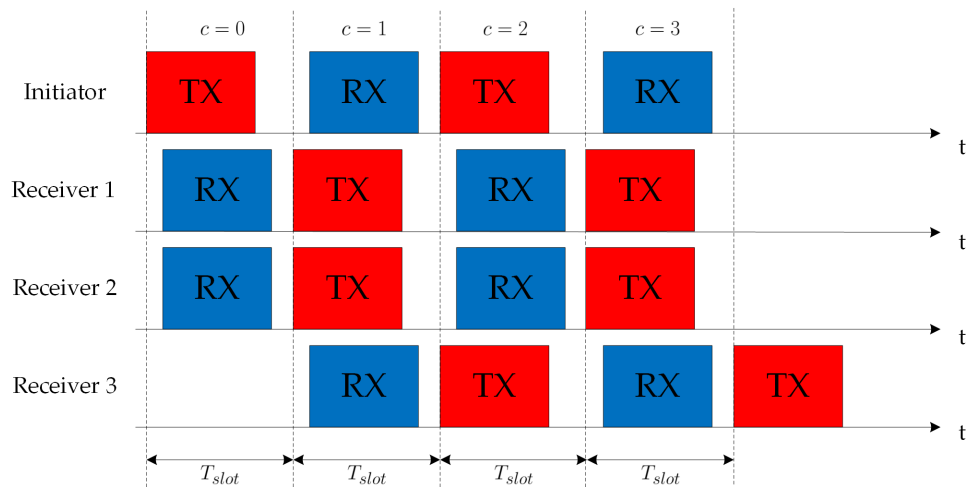


Fig. 3.1.: Example of a Glossy flood with 4 nodes and  $N = 2$

received packets allows, that even nodes multiple hops away from the initiator receive a packet within short time.

### Synchronization

Besides the fast propagation of a packet through a network, Glossy includes also the feature to synchronize nodes to a global reference time.

As shown in Fig. 3.1, each flood consists of multiple slots. A slot is defined as the time  $T_{slot}$  between the start of a packet transmission and the start of the consecutive one. All the slots during a flood are enumerated using a relay-counter  $c$ , i.e. the first transmission by the initiator is slot with  $c = 0$ . This is done by sending the current relay counter with every transmission. Nodes are therefore at any time aware how many times a packet has been relayed and most importantly how many slots have passed, before they received their first packet. By locally estimating  $T_{slot}$ , which is highly constant, and with the knowledge of the relay counter  $c$  it is possible for a node to trace back the time at which the initiator started its first transmission. Therefore, after each flood all the nodes can calculate a



common global reference time  $T_{ref} = t_{tx,initiator}(c = 0)$ . More details behind these calculations are given in the next section.

### Reference Time Calculation

The whole flooding process is controlled by radio-events, e.g. after the end of a packet is signalled by the corresponding radio interrupt, the CPU triggers the next transmission. Each radio event<sup>1</sup> is precisely timestamped at the occurrence of its interrupt. Fig. 3.2 shows the most important timing

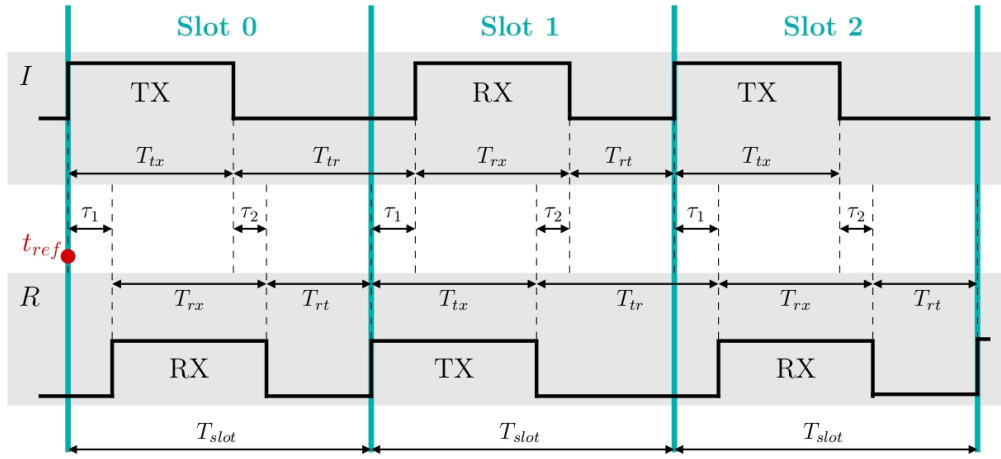


Fig. 3.2.: Timing relations between two nodes

metrics of a sequence of TX and RX packets between an initiator  $I$  and a receiver  $R$ :

- $T_{tx}$  and  $T_{rx}$  describe the time it takes to transmit or receive a packet, respectively.
- $T_{tr}$  and  $T_{rt}$  are the time distances between the start of a transmission and the start of a reception, and vice versa.
- $\tau_1$  defines the delay between the start of a transmission at the transmitter and the end of the reception at the receiver,  $\tau_2$  describes the same behaviour but relating to the end of a packet.

As mentioned in the previous section a node needs to calculate  $T_{slot}$  in order to calculate the reference time  $T_{ref}$  with the knowledge of  $c$ . Therefore, at the first reception each node saves the timestamp  $t_{rx,0}$  and the current relay counter  $c_0$ . According to Fig. 3.2,  $T_{ref}$  can be calculated as follows

$$T_{ref} = t_{rx,0} - \tau_1 - c_0 \cdot T_{slot} \quad (3.1)$$

<sup>1</sup>Start and end of RX/TX and RX/TX error

There are two ways to calculate  $T_{slot}$ :

1. Using only timestamped events

$$T_{slot} = \frac{T_{rx} + T_{rt} + T_{tx} + T_{tr}}{2} \quad (3.2)$$

This approach has the disadvantage that the calculations are only possible, if a successful  $RX \rightarrow TX \rightarrow RX$  consecutive sequence occurred, which might fail due to packet loss.

2. From Fig. 3.2, the relation between  $\tau_{1,2}$  and the timing of events can be described as following equations

$$\begin{cases} \tau_1 - \tau_2 = T_{tx} - T_{rx} \\ \tau_1 + \tau_2 = T_{tr} - T_{rt} \end{cases} \quad (3.3)$$

Combining (3.2) and (3.3) will give

$$\begin{cases} T_{slot} = T_{rx} + T_{rt} + \tau_1 \\ T_{slot} = T_{tx} + T_{tr} - \tau_1 \end{cases} \quad (3.4)$$

These calculations are possible after each  $RX \rightarrow TX$  or  $TX \rightarrow RX$  sequence, but on the other hand require the knowledge of  $\tau_1$ .

Due to the non-deterministic  $T_{rx}$  values, a node applies the second method to estimate  $T_{slot}$  as many times as possible and uses the average value over all estimations at the end of the flood to calculate  $T_{ref}$ . With a maximum number of transmission  $N$  a node can calculate up to  $2 \cdot N - 1$  estimated  $T_{slot,i}$  values, which leads to a final  $T_{slot}$ :

$$T_{slot} = \frac{1}{2 \cdot N - 1} \sum_{i=1}^{2 \cdot N - 1} T_{slot,i} \quad (3.5)$$

Therefore each node calculates its local reference time as follows:

$$T_{ref} = t_{rx,0} - \tau_1 - \frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n} \quad (3.6)$$

Both equation (3.4) and (3.6) depend on the variable  $\tau_1$ , elaborate details on this parameter are given later in section 3.2.2.

## 3.2. Synchronization Protocol

The current Glossy implementation simplifies its synchronization algorithm by assuming, that the timing behaviour is highly deterministic and does not consider any influence on the synchronization accuracy by non-constant time relations. In fact, all the previous presented timing metrics show a non-deterministic behaviour, which need to be considered for an in-depth synchronization protocol:

- The reception time  $T_{rx}$  is not constant, this is mainly caused by a non-deterministic packet transmission delay between two nodes. Fig. 3.3 shows the time between the start of a TX packet at one node and the start of the corresponding reception at a second node. Reasons for this behaviour is the limited resolution of the CC430's clock frequency, which was set to 13 MHz during the measurements, compared to the 868 MHz carrier frequency of the packet and the possible phase offset of the oscillators between the two nodes. In other words, it's impossible to capture the exact reception-time of a packet and as a result,  $T_{rx}$  shows a non-deterministic behaviour, see Fig. 3.4.

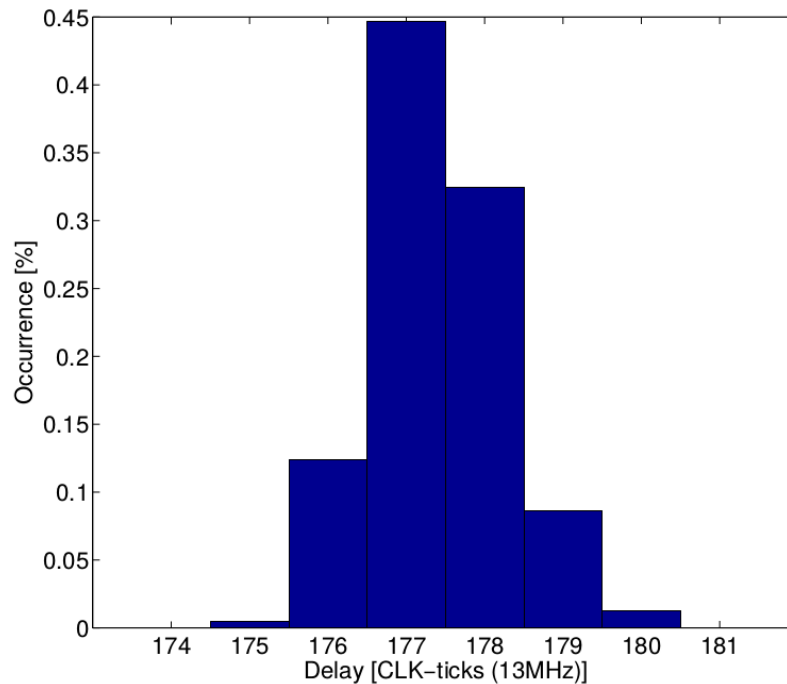


Fig. 3.3.: Packet delay between two nodes in clock ticks with a frequency of 13 MHz, standard deviation  $\approx 70$  ns

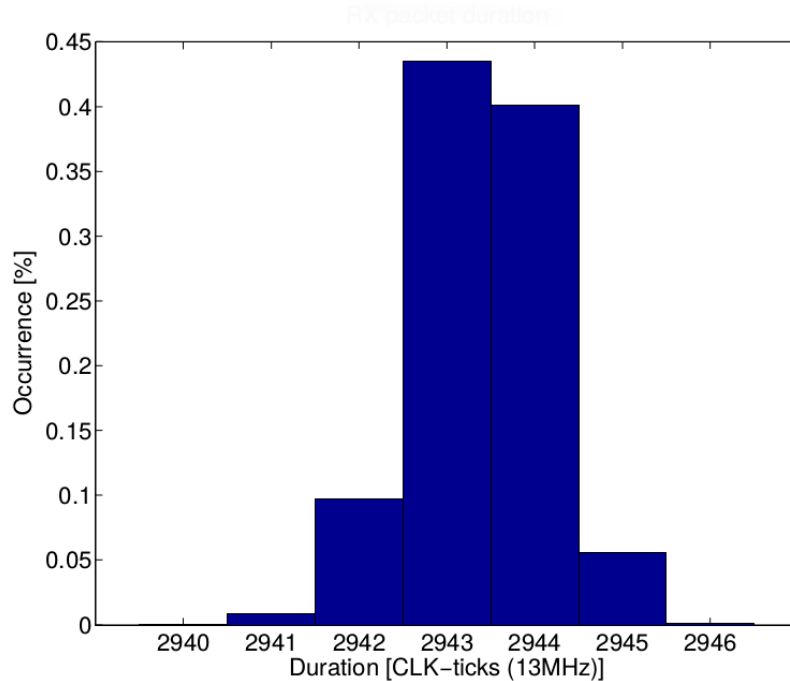


Fig. 3.4.: Duration of receiving a packet in clock ticks with a frequency of 13 MHz

- The transmission time  $T_{tx}$  is also not constant, even though the packet size is always the same. But unlike  $T_{rx}$ , it takes on only two sequent values (e.g. 2965 and 2966 CLK-ticks with a 13 MHz timer frequency). The main reason for this non-constant time is due to asynchronous clocks of the timer unit and the radio module. A frequency synthesizer re-samples the 26 MHz of the RF oscillator with various techniques like phase-locked loop (PLL) to generate the required radio frequency.
- On the CC430 the time to switch between the RX to TX mode and vice versa, is constant ([15, Chapter 25.3.3.7.4 *Timing*]), but due to the non-deterministic behaviours of  $T_{rx}$  and  $T_{tx}$ , the captured transition times are not constant either.  $T_{tr}$  shows the same behaviour as  $T_{rx}$  and  $T_{rt}$  as  $T_{tx}$ .

These timing uncertainties influence the synchronization performance and limit its accuracy. Therefore, in this section three improvements to the current Glossy synchronization process are presented, which compensate for following limitations:

- **Rounding errors:** The CC430 is always rounding the division in equation (3.6) downwards due to the representation of the

timestamps as integer variables. However, it is possible to detect, when a division can be rounded upwards and therefore adapt the calculated reference time  $T_{ref}$ .

- **Packet delay:** The transmission of a packet between two nodes is delayed, mostly due to the time it takes to demodulate the preamble of a packet. The current implementation assumes this value to be constant, namely as  $\tau_1$ . However, the transmission time is also affected by the propagation delay of the packet, which is not constant over all nodes. Because  $\tau_1$  is used to calculate  $T_{ref}$  with the use of the first reception timestamp, the propagation delay needs to be considered as well. Although the calculations are not straightforward,  $\tau_1$  can be estimated under certain limitations.
- **Outlier suppression:** Timestamp uncertainties or limited number of  $T_{slot,i}$  estimations due to packet loss have an direct influence on the synchronization and might result in a reference time, which is not as accurate as in most of the cases. These outliers can be detected by estimating the current time with the previous one plus one second and using the difference between the actual and the estimated one as a metric for synchronization accuracy. With the use of this metric, the current reference time can be adapted by applying a weighted sum.

Mathematical background and further details on these three adaptations are described in the following sections.

### 3.2.1. Rounding Correction

All the timestamps and other timing related values are defined as unsigned integer values on the CC430. Every time a division is done, the arithmetic logic unit is therefore not taking any decimal digits into account, in other words the result is always rounded downwards. An important division, which has a direct influence on the synchronization, is the  $\frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n}$  estimation. Representing this term as a float variable would improve its resolution and the one of the synchronization. However, a reference time which is not synchronous to the CPU clock is not helpful, because setting pins (e.g. generating the globally synchronized timepulse) is bound to the clock frequency. Further, the usage of float variables requires more arithmetic instructions and is limited in precision. Therefore, it's not beneficial to represent any division results as floating point variable.

However, it is possible to round upwards according to the common rounding rules instead of always rounding downwards. In order to do

so, it's not required to perform the actual calculations with float's, it's sufficient to check if the following condition is true:

$$\frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n} - \left\lfloor \frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n} \right\rfloor \geq \frac{1}{2} \quad (3.7)$$

The condition checks if the decimal digits of the division are equal or larger than  $\frac{1}{2}$ . Because only the integer part of the division is known, (3.7) is formulated as follows:

$$\Rightarrow c_0 \cdot \sum_{i=1}^n T_{slot,i} - n \cdot \left\lfloor \frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n} \right\rfloor \geq \frac{n}{2} \quad (3.8)$$

The CC430 can precisely calculate all terms of (3.8) except  $\frac{n}{2}$ , if  $n$  is odd. But because the left-hand side value of (3.8) is a natural number  $\mathbb{N}$ , it is sufficient to change the division on the right-hand side of (3.8) to

$$c_0 \cdot \sum_{i=1}^n T_{slot,i} - n \cdot \left\lfloor \frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n} \right\rfloor \geq \left\lfloor \frac{n}{2} \right\rfloor + (n \bmod 2) \quad (3.9)$$

Whenever this condition is true, a node can round  $T_{ref}$  upwards by adding one CLK tick.

This correction is helpful on nodes which are more than one hop away from the initiator, because  $c_0 \neq 0$ . The effect of rounding upwards is an improvement of the average synchronization error. Whenever the CC430 rounds downwards,  $T_{ref}$  will be too late by at most one CLK-tick, the rounding corrects this for all the cases where the decimal part indicates that  $\frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n}$  was more likely to be larger. In fact, numerical analysis on multiple synchronization results have shown, that in over 91 % rounding upwards was beneficial to the synchronization accuracy. Experimental results of the rounding correction are presented in chapter 5.2.1.

### 3.2.2. Packet Delay

The time between the start of a transmission at the sending node and the start of the corresponding reception at a receiving node is delayed by a short amount of time. This delay is mostly due to the time it takes to demodulate the package and detect the sync-word, which depends on the used data rate, modulation scheme and carrier frequency. Therefore this time is constant on all nodes.

However, the delay is also influenced by the propagation delay of the packet and depends on the distance between two nodes. Because the network may not consist of equidistantly distributed nodes, the

synchronization needs to compensate for this delay. E.g. for a distance of 10 m between two nodes, the propagation delay is approximately

$$\delta_p = \frac{10 \text{ m}}{c} = 33.4 \text{ ns} \quad (3.10)$$

with  $c$  = speed of light.

Revisiting equation (3.3) and Fig. 3.2, the relation between  $\tau_1$  and  $\tau_2$  is specified as follows:

$$\begin{cases} \tau_1 - \tau_2 = T_{tx} - T_{rx} \\ \tau_1 + \tau_2 = T_{tr} - T_{rt} \end{cases} \quad (3.11)$$

By solving the system of linear equations, it is also possible to estimate  $\tau_1$  and  $\tau_2$ :

$$\begin{cases} 2 \cdot \tau_1 = T_{tr} - T_{rt} + T_{tx} - T_{rx} \\ 2 \cdot \tau_2 = T_{tr} - T_{rt} - T_{tx} + T_{rx} \end{cases} \quad (3.12)$$

In other words, after a consecutive TX  $\rightarrow$  RX or RX  $\rightarrow$  TX sequence it is possible to calculate the sum of the two associated  $\tau_{1,2}$  values.

These estimated values can be used to improve the synchronization by adapting the timestamp of the first received package  $t_{rx,0}$ , which is used to calculate  $T_{ref}$ :

$$T_{ref} = t_{rx,0} - \tau_1 - c \cdot \frac{1}{n} \sum_{i=1}^n T_{slot,i} \quad (3.13)$$

where  $\tau_1 = \bar{\tau}_1 + \tilde{\tau}_1$  is the sum of the constant part  $\bar{\tau}_1$  and the variable propagation delay influence  $\tilde{\tau}_1$ .

$\tau_1$  is also needed to calculate the duration of  $T_{slot}$ :

$$\begin{cases} T_{slot,i} = T_{rx,i} + T_{rt,i} + \tau_1 \\ T_{slot,i} = T_{tx,i} + T_{tr,i} - \tau_1 \end{cases} \quad (3.14)$$

But because  $\tau_1$  is alternately added and subtracted to  $T_{slot,i}$  and the final value is the average over the sum of them,  $T_{slot}$  is not affected by  $\tau_1$ . Even if the number of all  $T_{slot,i}$  is odd, the variable part  $\tilde{\tau}_1$  is in general smaller than the number of  $T_{slot,i}$  and hence is not affecting the average.

$\tau_2$  is not applied in the reference time calculations.

A maximum number of transmission  $N$  provides  $2 \cdot N - 2$  different  $2 \cdot \tau_1$  values, which are averaged to get an estimation of  $\tau_1$ :

$$\tau_1 = \frac{1}{2} \cdot \frac{1}{2 \cdot N - 2} \sum_{i=1}^{2 \cdot N - 2} 2 \cdot \tau_{1,i} \quad (3.15)$$

The estimation of  $\tau_1$  is then applied in equation (3.6). The performance of this adjustment is limited by the fact, that these calculations are based on the communication between only two nodes, as seen in Fig. 3.2. However, in the FlockLab network multiple nodes can communicate with each other at the same time and due to the nature of Glossy it is not possible to determine the exact communication sequence. In other words, it is not clear from which node a packet was received. But in the end it is still beneficial to estimate  $\tau_1$  rather than using a constant value, which is evaluated in section 5.2.2, but also how this limitation produces a remaining error in the final implementation in section 5.2.4.

### 3.2.3. Outlier Suppression: Weighted Sum

Due to different error sources, like measurement errors, remarkable packet loss etc., it might happen, that the locally calculated synchronization pulse is far apart from the average accuracy. Under the assumption, that the local clock rate is sufficiently constant, such outliers can be detected when the difference between the calculated  $T_{ref,k}$  at time  $k$  and the old value at  $k - 1$  plus one second are remarkably far apart, illustrated in Fig. 3.5.

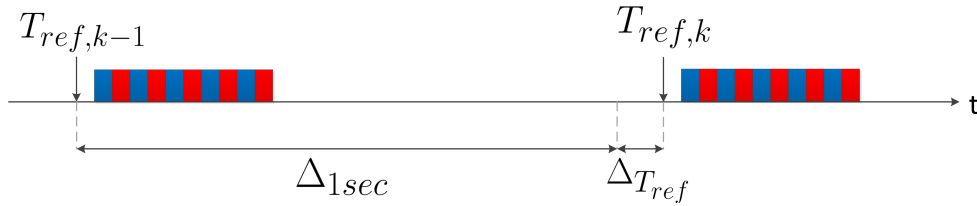


Fig. 3.5.: Outlier detection based on the difference between  $T_{ref,k}$  and  $T_{ref,k-1} + \Delta_{1sec}$

To counteract this,  $T_{ref}$  can be calculated as follows:

$$\Delta_{T_{ref}} = T_{ref,k} - (T_{ref,k-1} + \Delta_{1sec}) \quad (3.16)$$

$$\tilde{T}_{ref,k} = \frac{1}{|\Delta_{T_{ref}}|} \cdot T_{ref,k} + \left(1 - \frac{1}{|\Delta_{T_{ref}}|}\right) \cdot (\tilde{T}_{ref,k-1} + \Delta_{1sec}) \quad (3.17)$$

with

- $T_{ref,k}$ : reference time at time  $k$ .
- $|\Delta_{T_{ref}}|$ : absolute difference between  $k$ 'th reference time and  $(k-1)$ 'th + one second (which would be the perfect synchronization point at time  $k$  in case of a perfect one at time  $k - 1$ ).
- $\Delta_{1sec}$ : #CLK ticks during one second.



The weighted sum in (3.17) considers  $\Delta_{T_{ref}}$  as a local metric for synchronization accuracy at time  $k$ , i.e. for small  $\Delta_{T_{ref}}$  the calculated reference time is feasible. On the other side, a large  $\Delta_{T_{ref}}$  is caused by an inaccurate calculation of  $T_{ref,k}$ , assuming that  $T_{ref,k-1} + \Delta_{1sec}$  is accurate. Therefore the weighted sum makes sure in case of a large  $\Delta_{T_{ref}}$ , that the influence of  $T_{ref,k}$  is lessened with respect to the magnitude of  $\Delta_{T_{ref}}$ . In statistical terms this will demagnify the variance of the synchronization error.

In order to generate a feasible result, an accurate calculation of  $\Delta_{1sec}$  is required. This can be done by using a sliding window over a fixed number  $M$  of  $T_{ref}$  and average the difference between two consecutive  $T_{ref}$  values. Therefore the calculation on the CC430 looks as follows

$$\Delta_{1sec} = \frac{1}{M} \sum_{i=k-M+1}^k T_{ref,i} - T_{ref,i-1} \quad (3.18)$$

This sliding window is performed locally on each node and adapts, depending on the size of  $M$ , to any possible clock drift in short time.

### 3.3. GPS pulse dissemination in FlockLab

The above presented synchronization protocol is used to synchronize the 30 FlockLab nodes. The initiator node is equipped with a GPS device and captures the PPS pulse. As soon as a GPS pulse is captured, a new Glossy flood is prepared and after a constant delay initiated. After all nodes have relayed a packet  $N$  times, the flood is finished and each node calculates the globally synchronized  $T_{ref}$  according to the above presented algorithm. Based on this reference time, the function, which emits the pulse for the DAQ, and the next Glossy flood, which will happen one second later, are scheduled. The pulse for the DAQ is triggered with a predefined offset after  $T_{ref}$ . Because this offset might be affected by any clock drift, each node compensates for the drift by using the locally estimated second  $\Delta_{1sec}$ . For figurative illustration of the whole synchronization process see Fig. 3.6.

To ensure that the timestamps and the corresponding calculations aren't effected by clock drift, a flood should be as quick as possible. The duration of a flood can be controlled by setting different parameters, which are associated with trade-offs concerning synchronization accuracy:

- **Maximal number of transmissions:**  $N$  determines how many times a node relays a packet and therefore the more relays, the longer a

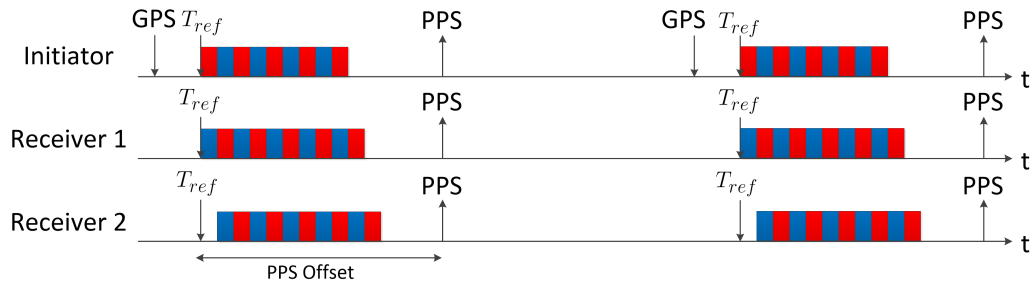


Fig. 3.6.: Synchronization process for FlockLab

flood takes. But a small  $N$  limits the number of times  $T_{slot,i}$  in (3.5) can be estimated and hence might affect the synchronization.

- **Packet size:** The slot length depends on how much data needs to be transmitted. Therefore, it is best to keep the packet size as small as possible.
- **Modulation scheme and data-rate:** The modulation scheme and the data rate have a direct influence on how fast a packet is transmitted, but also how reliable it can be received, since it might prevent from exploiting the constructive interference.
- **Initiator node location:** To shorten the flooding latency, a initiator node with a central position and multiple neighbours is recommended. A central node is able to spread a packet in all directions of the network and therefore minimize the maximum hop-distance between the initiator and any node.

Glossy uses a 2-GFSK modulation with a data rate of 250 kbps. Because only a header with no additional data needs to be flooded through the network, the payload of the data packet is empty. The header is a 4-byte field holding the initiator ID, a header type identifier and the current relay count. Tests have shown that a reasonable value for  $N$  is 5. In the end with these settings, the maximal duration for a node to execute a whole flood is approximately 5 ms and over all nodes below 8 ms.

How the synchronization process performs for the FlockLab network, is evaluated in chapter 5.

# 4

## FlockDAQ Board

The FlockDAQ board is an extension for FlockLab, which improves the current limitations on event tracing frequency and time synchronization. This chapter presents an overview of the hardware design and its requirements (section 4.1) as well as the most important components and their purpose (section 4.2).

### 4.1. Requirements

The current FlockLab observer consists of an embedded Gumstix computer and the FlockBoard, which connects the Gumstix to the target nodes. This solution limits the rate of events, which can be detected and their time-stamp accuracy. The data acquisition unit DAQ presented in [2] proposes an improved model to handle the event tracing and time-stamping. The DAQ is integrated into FlockLab by placing it between the Gumstix and the FlockBoard, where it enhances the current data acquisition process. Because the hardware implementation is integrated into the existing setup, certain requirements and limitations are given.

The Gumstix is connected to the FlockBoard over a 60 pin Verdex connector, which holds besides the target's tracing and actuation lines also an USB bus, an SPI bus to the ADC on the FlockBoard and various other data, control and power lines. The FlockDAQ board, which is placed between the connectors, needs to route the required signals from the FlockBoard to the DAQ and from the FlockDAQ's internal interfaces to the Gumstix. The signals, which are not required, are traced

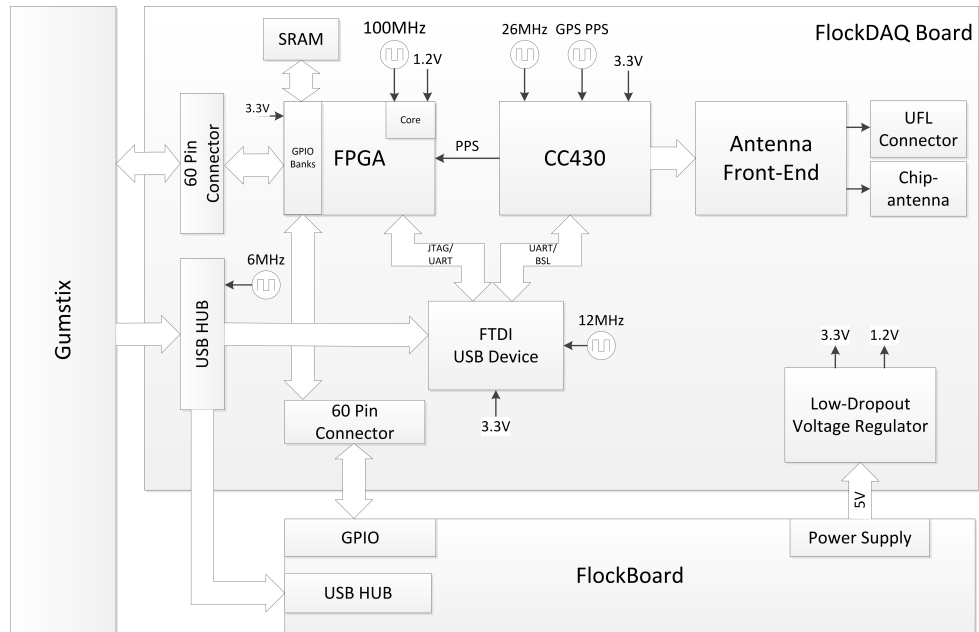


Fig. 4.1.: FlockDAQ board block diagram showing the most important parts, data and power paths

directly through, so the existing configuration is not disrupted. Because the hardware implementation of the DAQ needs multiple different interrelated components and these must not interfere with the existing setup, certain implementation aspects need special attention.

The FlockDAQ board combines all these components, the most important ones are shown in Fig. 4.1. In broad terms, the board can be parted into four specific modules:

1. **Data acquisition unit:** The DAQ is an FPGA based design, which handles the three common FlockLab services: event tracing, pin actuation and power profiling. To do so, the corresponding signals are traced to GPIO pins, but also from further FPGA outputs to the Gumstix. It is able to trace the mentioned signals with a high rate by using an external memory, which serves as a buffer for data packages. The DAQ transmits the packages over SPI to the Gumstix or, if the DAQ's services are not needed, routes all the signals directly through.

Further it provides a highly accurate timestamp to each event, which is achieved by the use of a synchronized internal clock. The synchronization is based on a pulse-per-second (PPS) signal and is received by the synchronization unit, which is described later.

The DAQ is configured and controlled over an UART connection and a reset line.

The FPGA needs to be reprogrammed after every power-cycle, which is possible with various techniques such as JTAG, an external flash memory or over an SPI bus.

To perform all the mentioned features, the FPGA must provide at least 78 available GPIO pins.

2. **Synchronization module:** The FlockDAQ runs a synchronization protocol to synchronize all nodes to a common global time. The software is implemented on a CC430 [16] chip, a System-on-Chip module, which combines an MSP430 micro-controller and a CC1101 radio core. The synchronization module is able to generate a PPS signal with an accuracy in the sub- $\mu$ s range as input for the DAQ, but also to capture a GPS generated pulse and disseminate it over the network. The radio module provides balanced radio frequency signals and needs an external balun circuit and an antenna. For optimal communication performance these two components should be optimized for an application with the CC430.

The chip needs to be flashed with an image of the synchronization algorithm, which is achieved by either using dedicated JTAG pins or a bootstrap loader (BSL) over UART.

3. **USB device:** In order to simplify the communication with the FPGA and the CC430, an USB to quad serial port chip is employed. It allows the communication over USB to four individual ports, which each of them can emulate different communication architectures.

However, only one USB bus is available from the Gumstix to the FlockBoard. Therefore an USB hub needs to connect both the serial converter and the existing connection with the Gumstix.

4. **Power supply:** The Gumstix is powered by a switching voltage regulator on the FlockBoard over supply pins on the Verdex connector. The regulator provides 5 V DC at a maximal output current of 2 A. This supply can also be used to power the FlockDAQ, but the operation of all these different modules requires a specific power management solution.

On one hand the FPGA requires two different voltage levels (1.2 V and 3.3 V) to power up its core and the I/O banks. On the other hand, the CC430 and USB device are also powered by 3.3 V and therefore the available output current must be large enough. The individual components need to be powered up in a specific sequence to ensure a successful booting process.

## 4.2. Hardware Components

### 4.2.1. FPGA

The FPGA is a Spartan-6 XC6SLX9 device [23] by Xilinx. The chip offers 112 available I/O pins, which are distributed over four I/O banks. The remaining 30 pins are used for power supply and various services like JTAG or internal pull-up resistor configuration.

Reasons for applying the LX9 FPGA are its sufficient GPIO pins and logic cells to implement the DAQ design. The final design will roughly use 50% of the available logical units and 78 of the 112 GPIO pins. A second reason is the available *Flat Pack TQG144* package type, which features extending leads. This allows a manual and in-house soldering of the chip onto the circuit board.

#### External SRAM

Data packets, which are generated by burst of events and therefore can not be transmitted to the Gumstix fast enough, are buffered in an external SRAM memory [13]. The SRAM offers a 512k long and 16bit wide memory array (equals a total of 1MB memory) and is also available as a flat pack package type.

### 4.2.2. CC430

The synchronization protocol runs on a CC430F5137 [16], an ultra low-power micro-controller of the CC430 family. The chip combines an MSP430 architecture with a CC1101 radio core. The MSP430 16-bit CPU offers up to 32 kB integrated programmable flash memory, 4 kB of RAM, two timers, 30 available I/O pins and various other features. The radio module supports three different frequency bands, a wide range of data rate, output power up to 10 dB, various modulation schemes and on-chip support for packet-oriented applications. The two architectures are merged in a small and compact IC package, which makes it a suitable tool for the synchronization algorithm in a space-saving way.

#### Clock management

The CC430 offers three internal clock signals, provided by the unified clock system (UCS). The UCS is sourced by a 26 MHz external oscillator, which is required for the radio functionalities. The internal clock signals are derived from the oscillator by dividing the frequency by  $2^x, x = \{0, 1, \dots, 5\}$ . However the CPU and the peripheral driver support only frequencies up to 20 MHz and therefore the maximal frequency for

the internal clocks is  $\frac{26\text{MHz}}{2^1} = 13\text{MHz}$ .

It is also possible to use an additional oscillator to source the clock signals, but due to limited space and a maximal CPU frequency of 13 MHz is sufficient for the synchronization process, no additional oscillator is applied.

### Timing-Related Functions

The emitted PPS by the CC430 to the DAQ needs to be carried out in a deterministic way in order to provide high accuracy. This can be done by using a compare block of the timer. The basic mechanism is to control output signals when a user-defined value in a dedicated register, the compare block, is identical to the counter register. Therefore the PPS signal line is connected to a specific pin, which is connected to a compare block, and consequently the pulse can be set at an exact time.

A similar feature is used to detect the GPS time pulse. A capture register records the current counter value, whenever a rising edge occurs and hence the GPS pulse output used by the initiator is connected to such a dedicated capture pin.

### 4.2.3. Radio Frequency Front End

#### Balun

The conversion of the unbalanced RF output of the radio module is done by a balun from Johanson Technology [7], a single chip solution specifically designed for the use in combination with the CC430, or the CC1101 radio core in general. On the unbalanced output side a simple filter circuit is placed between the balun and the antenna port. All the circuit tracks are designed to meet 50  $\Omega$  impedance characteristics.

#### Antennas

In order to offer flexibility between a compact hardware solution and RF performance, two different antennas can be used:

- **PCB chip antenna:** A 868 MHz ceramic chip antenna by Johanson Technology [6]. For optimal performance the antenna needs a pi-network tuner and a specific PCB trace layout.
- **External antenna:** An external antenna can be connected using an UFL connector.

Indoor range test have shown, that the chip antenna is able to communicate over a 60 m long hallway with high reliability. The MSP430-

CCRF Olimex board, which serves as a reference, achieves the same reachability, as well as an external antenna.

However during tests within the FlockLab, the chip antenna is not able to reach a similar performance to the MSP430-CCRF Olimex board. Reasons for this behaviour might be the polarity [6, Page 3] and the position of the chip antenna. It is attached on top of the FlockBoard where various power and ground planes can interfere with the RF communication. The external antenna, which can be placed away from the FlockBoard, achieves therefore a better performance.

The desired antenna can be selected by soldering capacitor C65 appropriately. Fig. 4.2 shows which pads need to be used to switch between the antennas.

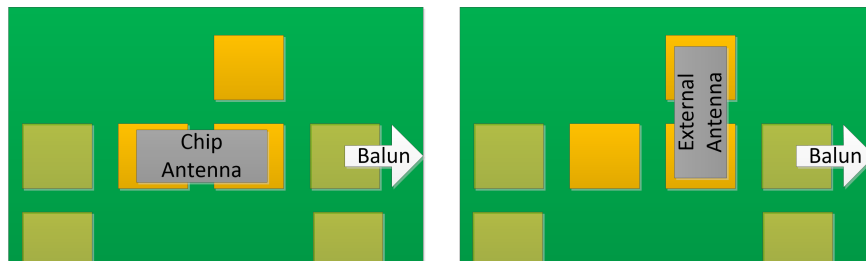


Fig. 4.2.: Left: Horizontal soldering for chip antenna. Right: Vertical soldering for external antenna

#### 4.2.4. USB

An USB hub on the FlockBoard connects the Gumstix with three external USB ports and with a second hub in series to the four target slots. The FlockDAQ adds another USB hub [18] between the Gumstix and the FlockBoard, which makes it possible to control the FlockDAQ via the Gumstix without interfering with the existing USB connections.

The FPGA has a volatile configuration memory, in other words after every power-cycle the FPGA needs to be reprogrammed. This is achieved by using the dedicated JTAG pins. Programming the FPGA over an external flash memory or SPI would require additional hardware parts and software modifications. The DAQ needs an UART connection for configuration and to write out timing related parameters. Further the CC430 needs to be flashed as well and offers debug information over UART.

To manage all these tasks, an FTDI FT4232H USB to quad UART converter [4] is applied. This single chip handles the USB protocol and



offers various serial protocols on four individual ports. In this design only three ports are used:

- **Port 0: FPGA JTAG:** The FTDI chip emulates the standard JTAG protocol and therefore is able to reprogram the FPGA.
- **Port 1: CC430 UART/BSL:** On one side this port manages the UART communication with the CC430 and on the other side exploits also the bootstrap loader (BSL) functionality to flash the CC430 .
- **Port 2: FPGA UART:** The third port manages the UART communication with the FPGA, which is needed to configure the DAQ.
- **Port 3: not used**

How the FTDI chip can be used to perform all this actions is described in the appendix A.1.

#### 4.2.5. Power Supply

The power supply is managed by a dual-output, low-dropout voltage regulator (LDO) [17]. It converts 5 V input voltage from the FlockBoard down to two output channels:

- $V_{OUT1}$ : Fixed 3.3 V with up to 1A output current
- $V_{OUT2}$ : Fixed 1.2 V with up to 2A output current

The FPGA is powering its core unit, which is used for logical operations, with 1.2 V and all its other functions with 3.3 V. All other components are also powered by 3.3 V.

#### Decoupling Circuits

The power supply is a crucial factor for an FPGA-based PCB design in order to provide high-speed I/O service. Each FPGA I/O bank has multiple power supply pins, as well as the internal core unit and auxiliary services like the clock management unit or JTAG. Each of these pins needs a decoupling network to filter any noise in the voltage level.

In order to fulfil the recommended specifications for covering both low and high frequency noise ranges, each power supply pin is decoupled by using a  $4.7 \mu\text{F}$  ceramic capacitor with package type 0402. These capacitors feature frequency characteristics (impedance  $|Z|$  and equivalent series resistance  $ESR$  vs. frequency  $f$ , [14, Fig. 1]), which are suitable for an appropriate decoupling.

Similar to the FPGA, decoupling of the power supply pins of the CC430, the FTDI chip and the LDO itself is an important aspect for optimal functionality. Therefore the design follows the recommendations by Texas Instruments [16, Figure 28][17, Page 34] and FTDI [4, Chapter 6.2 ].

### Startup Sequence

The design follows a specific start-up sequence, see Fig. 4.3, which ensures that all components boot correctly without interfering with each other. In

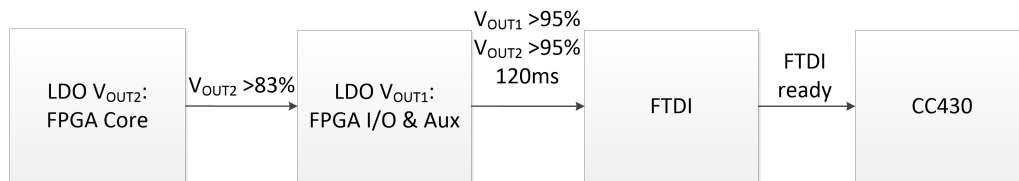


Fig. 4.3.: FlockDAQ startup sequence of the ICs

order to meet the requirements of the FPGA to power up the core first, the LDO enables  $V_{OUT1}$  after  $V_{OUT2}$  reached 83 % of its output voltage.

Further the LDO features a reset indicator pin, which is connected to the FTDI's reset pin. It holds the FTDI chip in a reset state as long as the two output voltages have not reached at least 95 % of their designated output level and a 120 *ms* delay has passed.

The reset line of the CC430 is driven by an FTDI port, and consequently the CC430 is also hold in a reset state, as long as the FTDI chip is not ready.

## 4.3. Final Design

Fig. 4.4 shows the top layer of the FlockBoard. On the left side is the CC430 with the chip antenna and UFL connector assembled, on the top right-hand side the LDO and on the bottom right-hand side the FTDI USB converter. On the far right side is the Verdex connector (female), where the Gumstix is plugged in. The bottom layer of the FlockDAQ is displayed in Fig. 4.5, in the center are the Spartan-6 and the external SRAM memory and on the bottom the hub. On the far left side is again a Verdex connector (male), which connects the FlockDAQ with the FlockBoard. In Fig. 4.6 the placement of the FlockDAQ board between the FlockBoard and the Gumstix is shown.

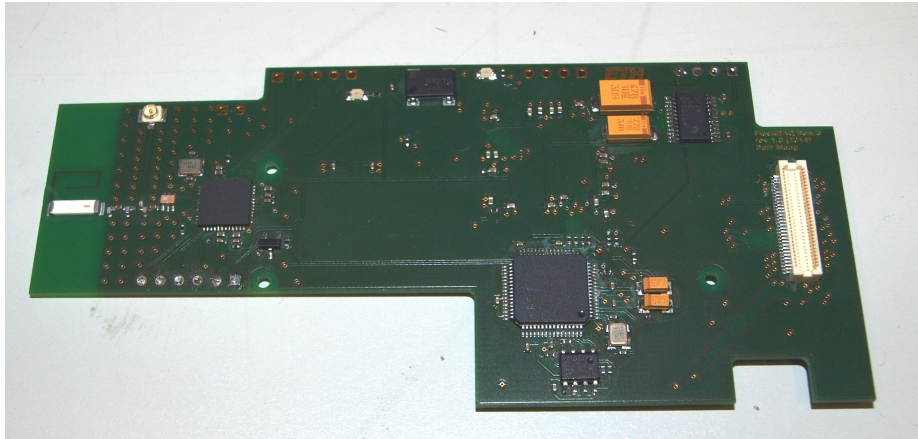


Fig. 4.4.: Top layer of the FlockDAQ board

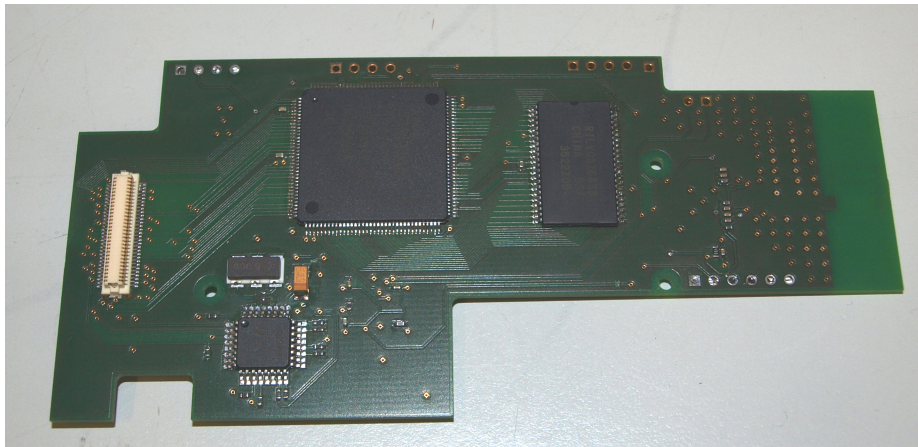


Fig. 4.5.: Bottom layer of the FlockDAQ board

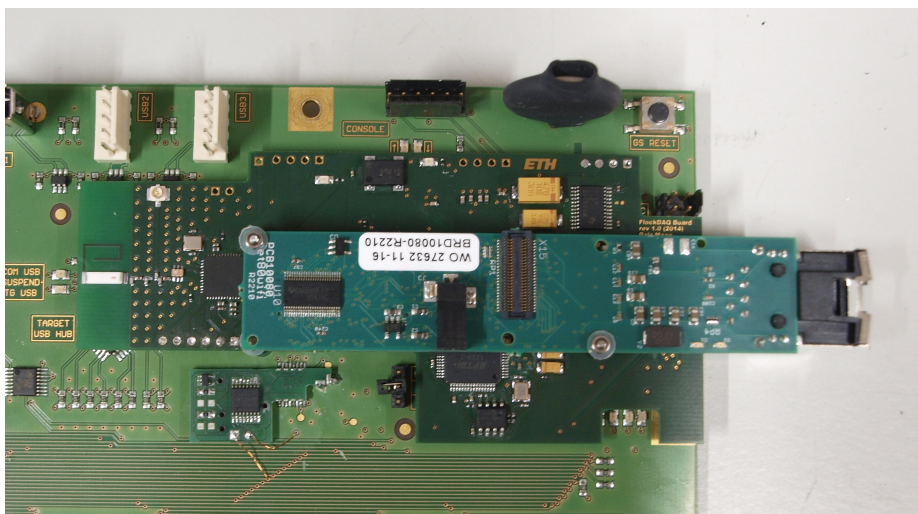


Fig. 4.6.: FlockDAQ board assembled between FlockBoard and Gumstix

### Power traces

For current measurements a power analyser is powering a FlockLab observer with 12 V DC and tracing the current draw.

The FlockDAQ draws in idle mode 40 mA, and the whole FlockBoard equipped with the FlockDAQ in idle mode 230 mA.

Fig. 4.7 shows the power trace during a test with the FlockDAQ activated. The employed target executes a simple "blink" application, where the three LEDs visualize a 3-bit counter. While the test is running small changes due to the number of flashing LEDs are recognizable, but in average the board draws 290 mA. When the test stops, the FTDI communicates to the FPGA and the DAQ resets all its modules and clears the FIFOs, which leads to a temporary rise in current up to 360 mA. After the test is completely finished, the DAQ is in idle mode, the CC430 and the target are still running and the current drops to an average of 265 mA.

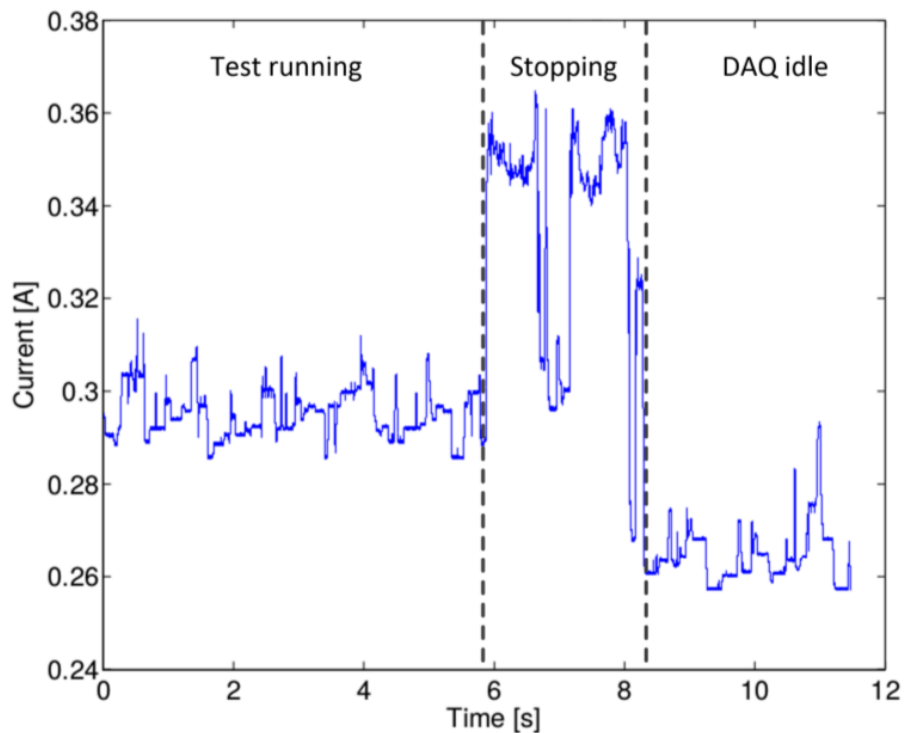


Fig. 4.7.: Current draw at 12 V DC during a running test, stopping it and with the DAQ in idle mode

# 5

## Evaluation

In this chapter the presented synchronization protocol is evaluated based on how accurate a pulse can be distributed through the network and how the FlockDAQ board performs, using this pulse to synchronize its timing module. Section 5.1 describes the testing setup and the metric for benchmarking the synchronization. In section 5.2 the synchronization accuracy is evaluated with emphasis on the three respective Glossy improvements and the remaining errors in the final implementation. The performance of the FlockDAQ board, in particular the synchronization of the internal clock and a GPS pulse tracing test, is presented in chapter 5.3.

### 5.1. Setup

The evaluation of the synchronization protocol performance is conducted by using all the available FlockLab observers. The protocol is running on the MSP430-CCRF targets [12], which uses an on-board PCB antenna. The targets on the four outdoor observer are equipped with an external antenna.

For the test with the FlockDAQ board, observer 1 and 202 were equipped with a PCB and used the external antenna for communication.

#### Synchronization Error

An important metric for the synchronization accuracy is the node-to-node error of the generated reference time, Fig. 5.1 elaborates its definition. To measure this error, a GPS device needs to provide a node with its PPS

signal. Therefore, six u-blox GPS devices are distributed in the FlockLab network according to the map in Fig. 5.2. These devices are able to simultaneously emit a time pulse with an accuracy below 15 ns [20]. The time of the PPS occurrence and the calculated  $T_{ref}$  are used to calculate the error as follows:

$$d_1 = T_{ref}^{(R1)} - T_{GPS}^{(R1)} \quad (5.1)$$

$$d_2 = T_{ref}^{(R2)} - T_{GPS}^{(R2)} \quad (5.2)$$

$$\Rightarrow error_{(R1) \leftrightarrow (R2)} = d_2 - d_1 \quad (5.3)$$

The error defines how accurate two nodes (R1 and R2) are synchronized to each other. For the evaluation, the synchronization error of a node is calculated with respect to the initiator.

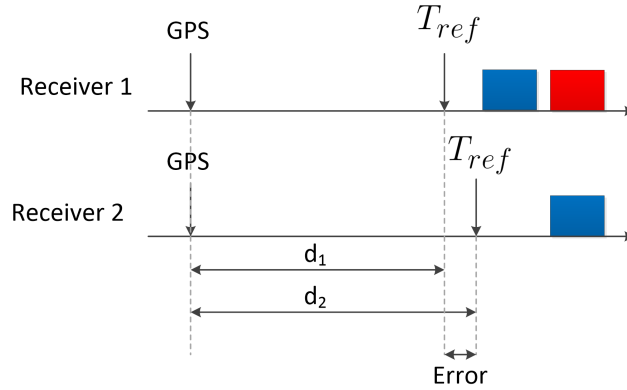


Fig. 5.1.: Definition of synchronization error using GPS reference times

### Tests

The following evaluations are based on test results over a total of six hours, conducted at various times during a day and different weekdays. This ensures, that influences of environmental conditions, like closed office-doors, working people or temperature variations, are broadly covered.

In all the tests the CC430 CPU frequency was set to 13 MHz and the maximal number of transmission  $N = 5$ .

### Initiator Node

The choice of the initiator node has an influence on the synchronization performance, mostly how fast and reliable a flood can be conducted. The best node is one with a central position in the network and preferably multiple direct neighbours. Therefore, the initiator for all tests was node 24.

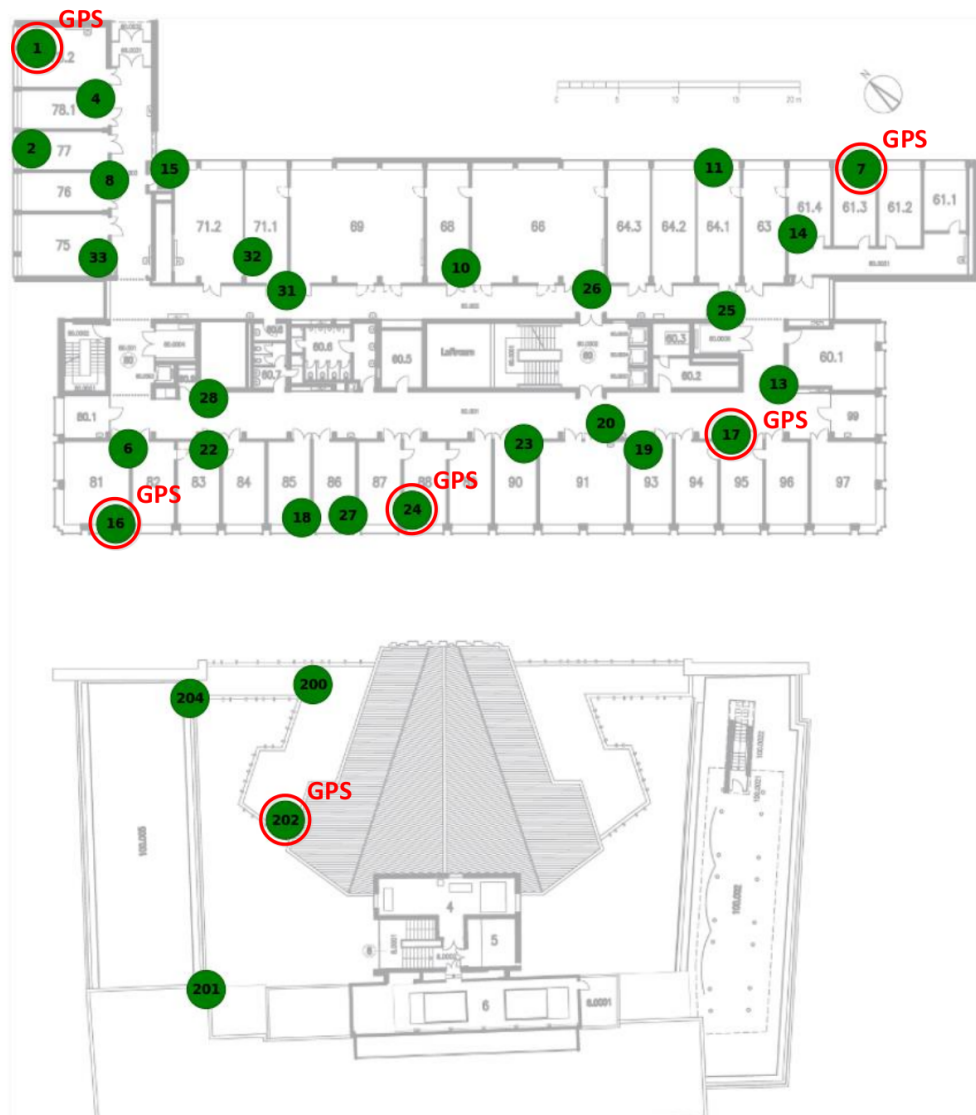


Fig. 5.2.: FlockLab map: 6 nodes equipped with a GPS device, node 202 is located outside and node 24 is the initiator

## 5.2. Synchronization Accuracy

In this section the three improvements to the basic Glossy synchronization process, the rounding correction (5.2.1), the packet delay (5.2.2) and the weighted sum (5.2.3), as well as the final design and its remaining errors (5.2.4) are evaluated. To show the benefits of the individual improvements, the contemplated feature is turned off and the test results are compared to the final design.

### 5.2.1. Rounding Correction

As mentioned in chapter 3.2.1, the synchronization protocol rounds the division  $\frac{c_0 \cdot \sum_{i=1}^n T_{slot,i}}{n}$  according to the common rounding rules. The term is used to calculate the delay of the first reception after the global  $T_{ref}$ . Assuming that the rounding correction is not used, the locally calculated  $T_{ref}$  will be too late. If the rounding is adapting the reference time, this lateness is counteracted and hence the accuracy of the synchronization is improved, see Fig. 5.3. The histograms in 5.3 shows the error of node 1

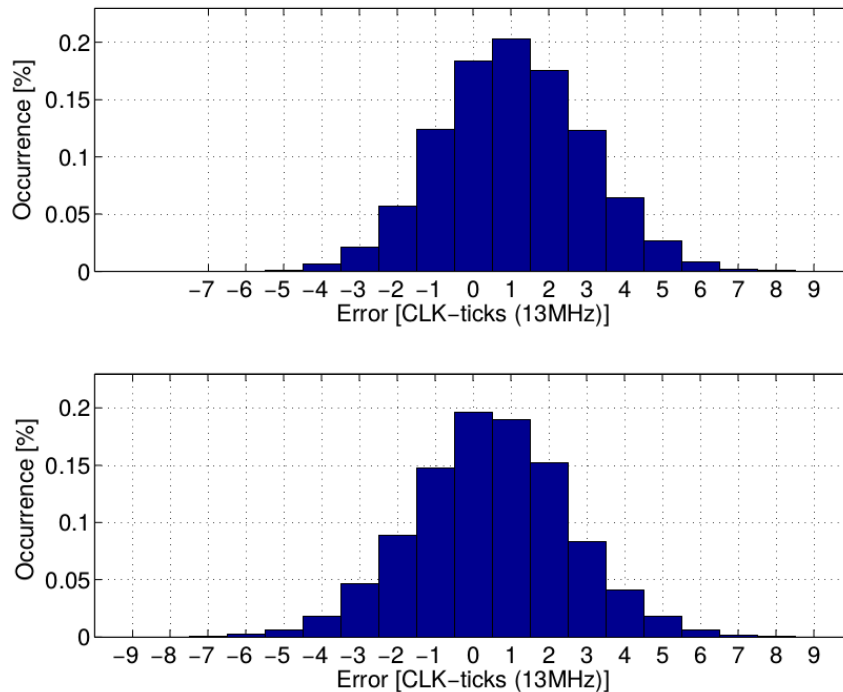


Fig. 5.3.: Synchronization error of node 1, top: without rounding correction, bottom: with rounding correction

with and without the correction. In mathematical terms the effect is an improvement of the average synchronization error, table 5.1 shows the average error for all evaluated nodes.



Node	Without rounding	With rounding
1	1.18	0.46
7	1.33	0.56
16	1.86	1.07
17	0.9	0.71
202	0.43	0.12

Tab. 5.1.: Average error [CLK-ticks (13 MHz)] of all nodes without and with the rounding correction, Initiator node = 24

For all evaluated nodes the rounding was beneficial to the average synchronization error. Also for node 202, which can directly communicate with node 24 and therefore most of the time does not apply the rounding correction. Yet in some cases it may happen, that  $c_0 \neq 0$ , for instance because of a communication error, and hence the rounding is also applied.

## 5.2.2. Packet Delay

In the current Glossy implementation it is assumed that the delay  $\tau_1$  of a packet between two nodes is constant over the whole network. In fact  $\tau_1$  consists of a constant part  $\bar{\tau}_1$  and a variable part  $\tilde{\tau}_1$ . The constant part  $\bar{\tau}_1$  depends on the data-rate and modulation scheme and is approximately  $13.54 \mu s$ . This equals 176 CLK-ticks at a frequency of 13 MHz.

The variable part  $\tilde{\tau}_1$  is influenced by the propagation delay and hence by the distance between two nodes. For instance the distance between node 24 and 202 is approximately 30 meters and would roughly induce a propagation time of  $100 ns$ . But as already described in chapter 3.2.2, calculating the actual  $\tilde{\tau}_1$  with the given timing informations is not straightforward and only an estimation. In fact, the calculated values for  $\tilde{\tau}_1$  vary, see Fig. 5.4.

The variations can be traced back to the uncertainty of the timestamps, multiple nodes sending at the same time and possible multi-path effects. Yet in 40% of the cases  $\tilde{\tau}_1 = 2$ , which is the delay that a propagation time of  $100 ns$  induces. And therefore using a non-constant  $\tau_1 = \bar{\tau}_1 + \tilde{\tau}_1$  is beneficial to the synchronization accuracy. Fig. 5.5 shows the error of node 202 when the measured  $\tilde{\tau}_1$  are used to correct the reference time.

Similar to the rounding correction, the effect is an improvement of the average synchronization accuracy. Again  $T_{ref}$  is too late without the correction and the effect is observable on all nodes, see table 5.2

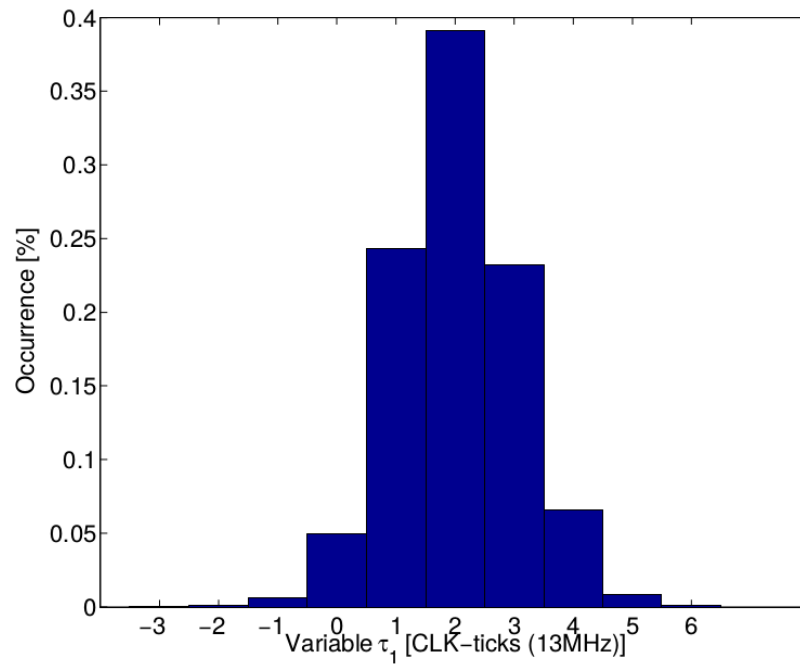


Fig. 5.4.: Measured variable part  $\tilde{\tau}_1$  of node 202

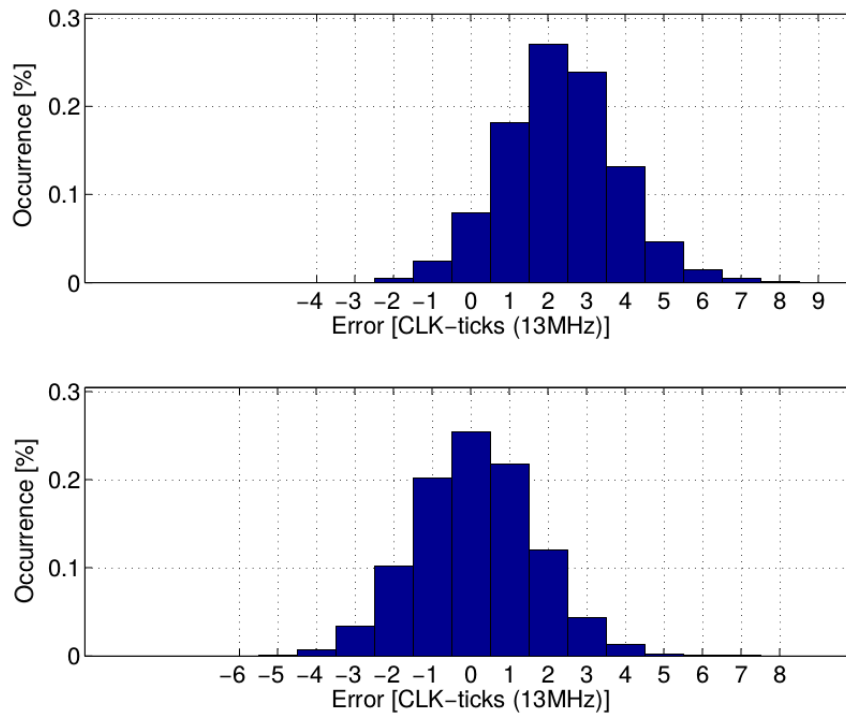


Fig. 5.5.: Synchronization error of node 202, top: without  $\tilde{\tau}_1$  correction, bottom: with  $\tilde{\tau}_1$  correction

Node	Without $\tilde{\tau}_1$	With $\tilde{\tau}_1$
1	2.81	0.46
7	2.32	0.56
16	2.97	1.07
17	2.24	0.71
202	2.30	0.12

Tab. 5.2.: Average error [CLK-ticks (13 MHz)] of all nodes without and with the  $\tilde{\tau}_1$  correction

### 5.2.3. Outlier Suppression: Weighted Sum

Various error sources like significant packet loss or the uncertainty of the timestamps can have a remarkable effect on the synchronization accuracy. The concept of a weighted sum in chapter 3.2.3 is removing outliers, i.e. reference times which have a notable worse error than in most of the cases. Fig. 5.6 shows the effect on the synchronization error for node 17, if the weighted sum is used. The weighted sum is not only removing

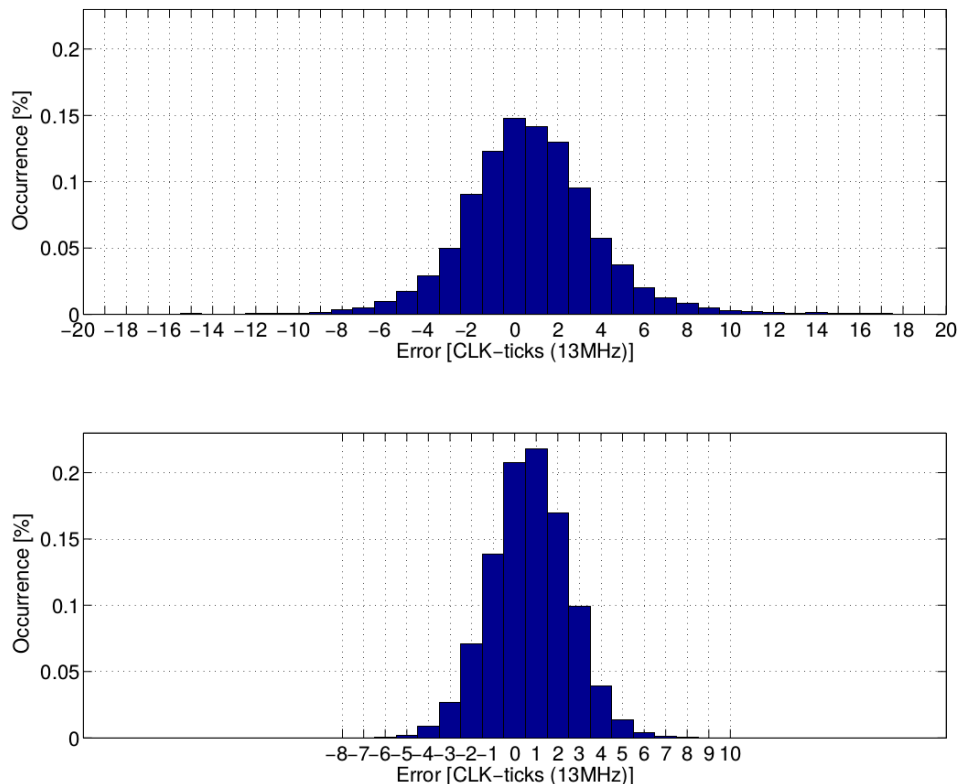


Fig. 5.6.: Synchronization error of node 17, top: without weighted sum, bottom: with weighted sum

single outliers but also reducing the standard deviation of the error. Table 5.3 shows the standard deviation and the range of the error for all nodes. The benefit on the error range for node 16 and 202 is not as extensive as

Node	Without weighted sum	With weighted sum
1	4.05, [-25,82]	2.07, [-9,9]
7	3.62, [-22,32]	1.83, [-7,9]
16	2.15, [-11,17]	1.66, [-6,9]
17	3.25, [-23,23]	1.84, [-8,10]
202	1.94, [-7,7]	1.57, [-6,8]

Tab. 5.3.: Standard deviation and range of the error [CLK-ticks (13 MHz)] for all nodes without and with the weighted sum

for the other nodes. These two are both close to the initiator node and are less affected by any packet loss, timestamp uncertainty or accumulating errors.

#### 5.2.4. Final Design: Remaining Errors

With the effect of the three improvements it was able to improve the synchronization accuracy of the basic Glossy protocol. Table 5.4 shows the metrics of the synchronization with all the three improvements turned off. With a timer frequency of 13 MHz a clock cycle takes approximately

Node	Average	Standard deviation	[Min.,Max.]
1	2.99	4.61	[-28,40]
7	2.18	3.14	[-31,24]
16	3.16	2.39	[-26,21]
17	3.39	1.96	[-11,11]
202	2.39	1.84	[-6,10]

Tab. 5.4.: Average error, standard deviation, error range of the basic Glossy design

$\frac{1}{13 \cdot 10^6 \text{ Hz}} = 77 \text{ ns}$ . Therefore the average synchronization error of the basic Glossy process is below  $260 \text{ ns}$  and achieves a maximal absolute error of approximately  $3 \mu\text{s}$ . Table 5.5 summarizes the most important metrics for the synchronization accuracy of the final protocol implementation. The accuracy is remarkably improved, the average synchronization accuracy for all nodes is below  $100 \text{ ns}$  with an absolute worst-case error of  $770 \text{ ns}$ . In other words, compared to the basic Glossy synchronization the final protocol achieves more than halve the average error and a four times smaller worst-case error.

Node	Average	Standard deviation	[Min.,Max.]	Hop distance
1	0.47	2.07	[-9,9]	3
7	0.56	1.83	[-7,9]	3
16	1.07	1.66	[-6,9]	1
17	0.71	1.84	[-8,10]	1
202	0.12	1.57	[-6,8]	0

Tab. 5.5.: Average error, standard deviation, error range and hop distance to initiator for the final design

The performance between the single nodes differs in various and non-intuitive aspects. With respect to the average error, the direct neighbour 202 of the initiator 24 performs best. However node 1 and 7, which are both three hops away, have a notably better average error than node 16, which is only one hop away. It seems, that the hop distance has no deteriorating effect on the average synchronization error. Concerning the standard deviation, the more distant nodes perform worse than the close ones. Their synchronization can be influenced by packet loss or uncertainty in timestamps, but also accumulating errors. For instance, an error in the first reception timestamp will be passed on to more distant nodes and influence their first reception timestamp. Hence, such an error can be accumulated per hop through the network.

Overall, various remaining error sources influence the final synchronization behaviour and are discussed in the following section.

### GPS Inaccuracy

Although the GPS devices provide an accuracy of their time pulse below  $15\text{ ns}$ , the conditions for this performance are not clear. Measurements with three LEA-6T devices have shown, that the difference between the provided PPS signals can be up to  $80\text{ ns}$ .

This inaccuracy is not an error in the synchronization protocol but in the evaluation. A worst-case device-to-device delay of  $80\text{ ns}$  can cause an error of 2 CLK-ticks, which will worsen the standard deviation of the evaluated synchronization error.

### Timing Uncertainty

The non-deterministic packet transmission delay between two nodes and the associated timestamp uncertainty have a remarkable impact on the synchronization accuracy. Especially the timestamp of the first reception, which is used as a starting point for the  $T_{ref}$  calculations, contributes a direct error. Fig. 3.3 shows a Gaussian-like distribution

of the transmission delay, which limits the precision of the first reception and hence a Gaussian can also be recognized in the final synchronization error. This non-determinism is not only affecting the first reception, but also all the taken timestamps and hence the calculations in the algorithm and can not be completely compensated for with the synchronization protocol presented in this work.

Also other wireless synchronization protocols [8] mention this Gaussian-like packet transmission delay as a fundamental limitation for synchronization accuracy.

### Inaccurate $\tilde{\tau}_1$ Estimations

Besides the timestamp uncertainty, another reason for inaccurate  $\tilde{\tau}_1$  estimations is the fact, that the calculations are based on the communication between only two nodes. However, the FlockLab network consists of 30 nodes, each of them having multiple neighbours. Due to the nature of a Glossy flood, the exact communication sequence is unknown and the assumption that always only the same two nodes are exchanging packets may not be true. This fact produces an error in the  $\tilde{\tau}_1$  calculations and is illustrated with an experiment:

Three nodes are connected to a signal splitter, one root node, one close receiver over a 10 m cable and one far receiver over a 50 m receiver, see Fig. 5.7. The splitter circuit connects the antenna outputs directly to each other and makes sure that the RF signals are not interfering with each other or are influenced by multi-path effects.

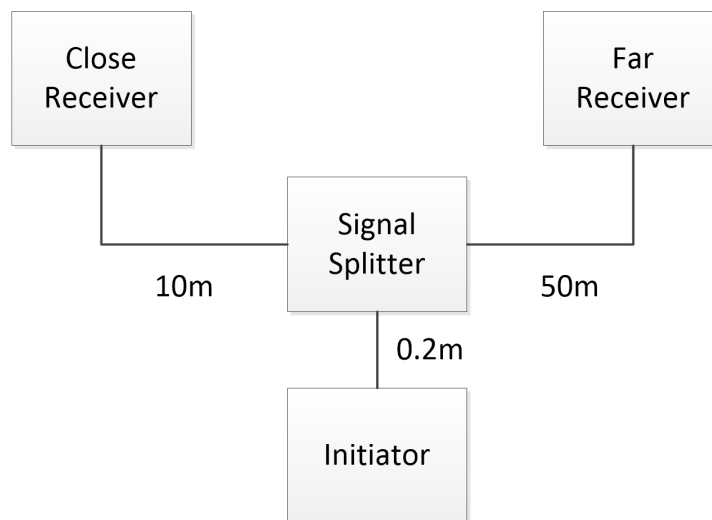


Fig. 5.7.: Signal splitter experiment, setup with three nodes

Table 5.6 shows the average synchronization error and the average  $\tau_1$  value. Because  $\tau_1$  is influenced by the propagation delay, the actual  $\tau_1$

Node	Average error	Average $\tau_1$
Close	-0.51	177
Far	2.16	177

Tab. 5.6.: Average error and average  $\tau_1$  [CLK-ticks(13 MHz)] for the close and far node

value for the far node should be larger, e.g. for a distance of 50 m roughly 2 CLK-ticks. The problem is, that the packets transmitted by the far node are not received by the initiator. The initial packet will first arrive at the close node and the first relayed packet received by the initiator is the one from the close node, because it takes notably longer to receive and relay the packet by the far node. This process repeats for all following packets, see Fig. 5.8. Hence the communication sequence is only influenced by the close and the initiator node. The far node will always receive packets, which are relayed based on the close node. In other words, it receives packets too early compared to the case, when only the far node is communicating with the initiator.

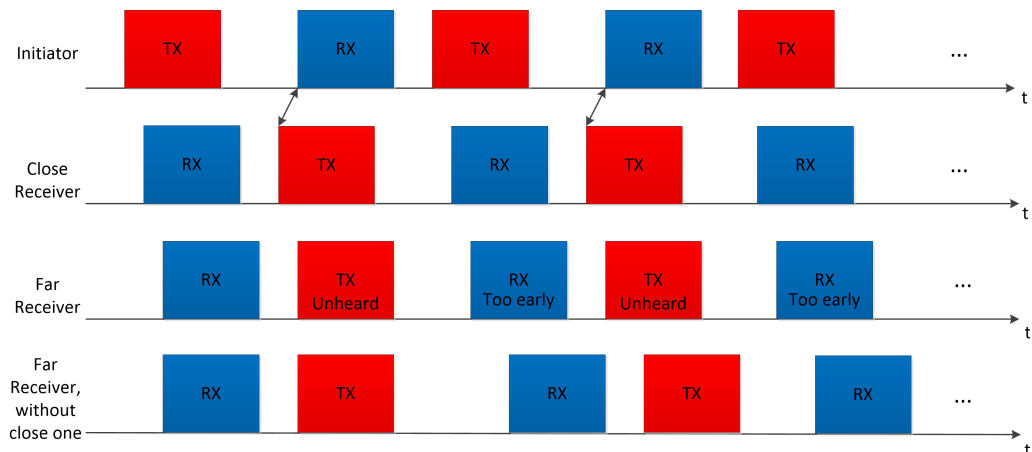


Fig. 5.8.: Packages of the far node are not received by the initiator and therefore the communication is only based on the close node and the initiator

In fact, if the close node is turned off, the average  $\tau_1$  increases to 179.3 CLK-ticks and consequently the average error decreases as well.

With respect to the FlockLab network, where multiple nodes at the same time can be affected by the communication of two close nodes and it is unclear whose packet has been detected, the situation gets more complex and induces a remaining error. If one node is relaying its packets too early, all its neighbours, which receive these packages will also be influenced

and the communication can be affected through the whole network.

This effect might be an explanation, why node 16 has a worse average error than node 7. Node 16 has remarkably more direct neighbours with unequal distances than node 7.

## 5.3. FlockDAQ Board Performance

The previous evaluation of the synchronization protocol shows, that it is possible to synchronize nodes within sub- $\mu s$ . However, the distributed PPS has not the same precision as the one of a GPS device, which was used by the prototype of the DAQ to synchronize its internal clock. This section evaluates the DAQ performance, when the synchronization protocol is used to provide the PPS.

### 5.3.1. Internal Clock Synchronization

The DAQ features an internal time calibration module, which is synchronized to the distributed PPS. The internal time is based on the 100 MHz oscillator and derives a 2M wide counter to represent the duration of a second. After this counter reaches its maximum, the DAQ provides some timing related information over UART.

One metric is the number of clocks between two consecutive PPS. Fig. 5.9 shows the histogram of this clock-count for node 202. The four major peaks in the histogram are due the deviation of the  $T_{ref}$  calculation. The peaks are roughly 80 ns apart, which is because of the smaller frequency of the CC430 (13 MHz vs. 100 MHz) and equals approximately one 13 MHz clock cycle.

Another important parameter is the offset between the internal full second counter and the PPS occurrence. It defines how accurate the internal time is synchronized to the global time. Fig. 5.10 shows the cumulative distribution function (CDF) of this absolute offset for node 202. In 90% of the cases the DAQ is able to synchronize its internal second within an error of  $\pm 200$  ns to the PPS.

### 5.3.2. GPS Tracing

To test the DAQ's functionality to trace and precisely timestamp a targets tracing signals, a GPS PPS is connected to a target slot on the FlockBoard. The DAQ traces the occurring pulses and records their time. An optimal behaviour would be, that the timestamps are always one second apart. Fig. 5.11 shows the difference of two consecutive PPS timestamps to the optimal interval of 1 s. In over 80 % of all cases the time between two



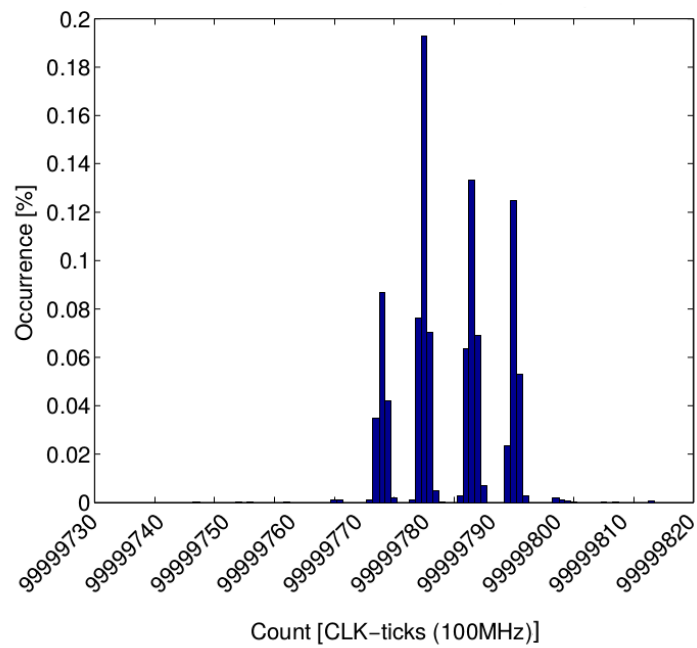


Fig. 5.9.: Difference between two consecutive PPS captures in CLK-ticks, Node 202

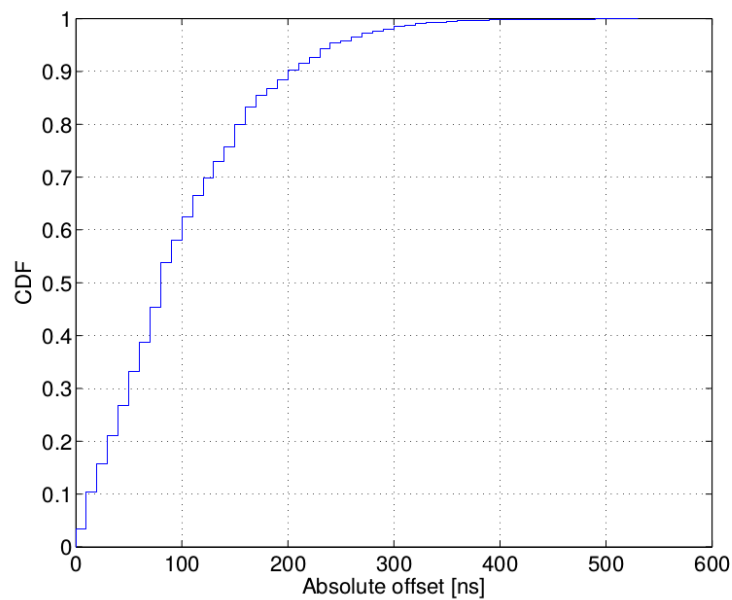


Fig. 5.10.: Empirical cumulative distribution function of the absolute offset between PPS occurrence and internal full second counter in  $ns$ , Node 202

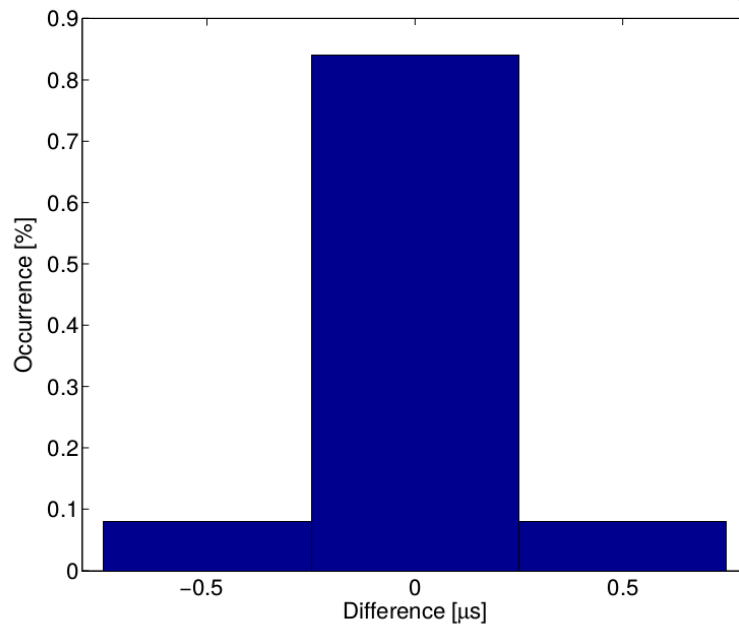


Fig. 5.11.: Difference of GPS PPS capture times to 1 second in  $\mu\text{s}$ , Node 202

pulses is exactly one second, in the remaining ones  $\pm 0.5 \mu\text{s}$ . These small differences are due to the limited accuracy of the synchronization protocol but also the inaccurate GPS pulse.

The same test without using the FlockDAQ achieved differences in a range of  $[-331 \mu\text{s}, 254 \mu\text{s}]$  with a standard deviation of  $37.6 \mu\text{s}$ .

The results show, that even with the more imprecise PPS of the synchronization protocol compared to a GPS device, the DAQ is able to provide accurate timestamps to the FlockLab events.

### Synchronization

The rising edge of the traced GPS pulse occurred  $60 \mu\text{s}$  after a full second of the internal DAQ time and the falling edge  $0.8 \text{ s}$  later. Fig 5.12 shows the synchronization error of tracing both edges between the two nodes. The

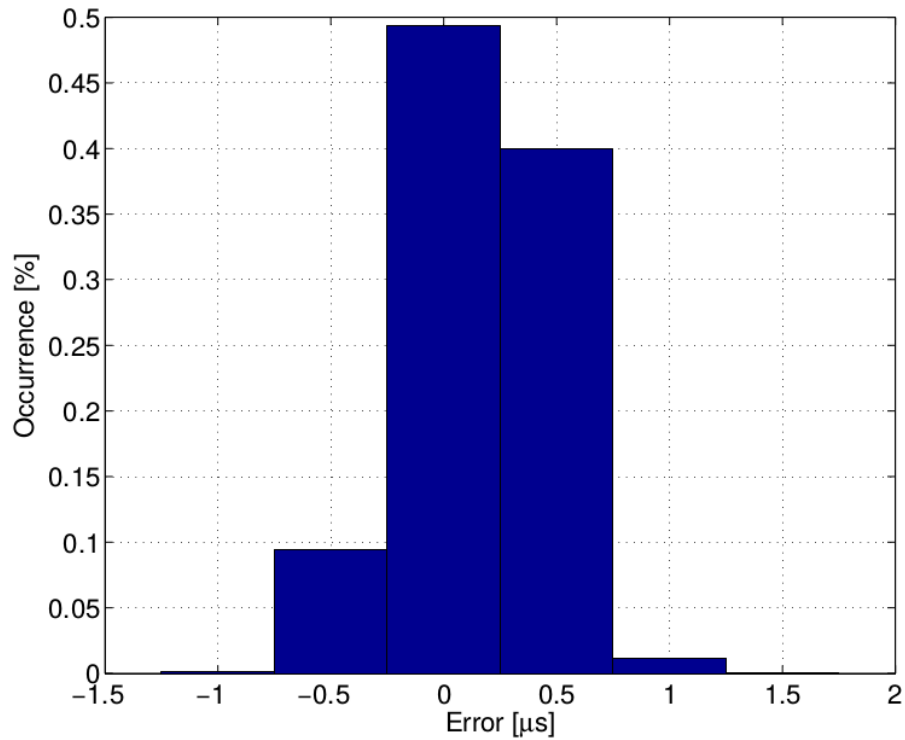


Fig. 5.12.: Synchronization error of the traced GPS pulse between node 1 and 202

average synchronization error between the two nodes is  $163 \text{ ns}$ . There is no notable difference between the two edges. This error is influenced by the synchronization protocol error, the inaccuracy of the two GPS devices and the limited precision of  $500 \text{ ns}$  of the DAQ's internal timing module.



# 6

## Conclusion and Outlook

### 6.1. Conclusion

This work presented a synchronization protocol for wireless networks. The basis for the protocol was Glossy, a flooding protocol for wireless networks, which is able to distribute a packet through a network in a highly reliable and fast manner. This is done by exploiting the constructive interference of wireless signals and applying a distinctive communication sequence. The highly deterministic timing behaviour of a Glossy flood provides an implicit technique for synchronizing nodes in a network. The implementation of Glossy on a CC430 system-on-chip is completely radio event driven and capturing the timestamps of these events is limited in precision. Especially due to a non-deterministic packet delay between two nodes, which is for instance induced by quantization errors between the local CPU and the re-sampled carrier frequency of the signal. This non-determinism limits the synchronization accuracy, therefore this work presented three improvements to the basic Glossy synchronization algorithm. The first adaptation is a rounding detection, which rounds a division to the common rounding rules, because divisions on a CC430 with integer value based timestamps do not consider decimal numbers and hence are always rounded downwards. A further improvement is the consideration of propagation delays between two nodes, the basic Glossy implementation assumes the time between the start of a packet at a node and its reception at a second to be constant. However, in a network with non-equidistantly nodes this delay is not constant and can be estimated with the given timing behaviour. The

last improvement is suppressing outliers, which might be induced by remarkable packet loss, timestamp uncertainty or accumulating errors. A weighted sum is adapting the current reference time, where the weight is based on the difference between the current reference time and the last one added by one second. The basic idea is to lessen the impact of any non-feasible reference time based on the last one. The final design of the synchronization protocol is used to disseminate a pulse-per-second of a highly accurate GPS device through the FlockLab network. This synchronizes each observer to a time pulse with an accuracy in the sub- $\mu$ s range.

In the second part of this work a hardware design for a recently developed FlockLab enhancement, the data acquisition unit DAQ, was designed. The DAQ improves the current limitations on event detection rate and timestamping of these events. It uses an FPGA with high-speed I/O to capture events and, if necessary, an external SRAM memory to store data packages. For providing accurate timestamps, the design features an internal timing unit, which is synchronized to a PPS signal. The prototype of the DAQ used a GPS device as input for the synchronization pulse. The hardware solution, the FlockDAQ board, presented in this work, combined all necessary hardware components for a successful implementation of the DAQ. Instead of a GPS device on each single observer, the presented synchronization protocol is used to provide the DAQ with a PPS, which is synchronized to one single GPS device. The synchronization algorithm is running on a CC430 unit. The RF communication can be executed with either a chip antenna or an external antenna. The integration of the FlockDAQ board into the FlockLab setup was bound to certain limitations and requirements for a successful implementation. For a compact communication and programming solution of the hardware units, the existing USB line between the FlockBoard and the Gumstix is extended with a USB hub to the FlockDAQ, where it is controlling an USB to quad port serial converter. This chip allows to program the FPGA over JTAG and the CC430 over BSL and to communicate with both modules over UART in a compact single-chip solution and without interfering with the existing USB line. The FlockDAQ was integrated into the FlockLab on two observers and evaluated based on the tracing of a GPS PPS.

The work concludes with an evaluation of the synchronization protocol with respect to the synchronization error of each node to the initiator and the performance of the FlockDAQ board. Multiple tests with five distributed nodes and an initiator have shown the benefits of the three Glossy adaptations. With the rounding detection and the packet delay consideration it is possible to improve the average synchronization error and the weighted sum removes outlier and lessens the standard deviation

of the error. In the end the final design is able to synchronize the nodes with an average error below  $100\text{ ns}$  and an absolute maximum error of  $770\text{ ns}$ . In fact, it was possible to achieve more than halve the average error and a four times smaller worst-case error compared to the basic Glossy synchronization without the three improvements. The remaining errors of the protocol are mostly due to the non-deterministic packet delay between two nodes and the associated uncertainty of the timestamps, but also the fact that the communication between two close nodes might influence the timing behaviour of any node, which is more distant and its transmitted packages are not affecting the communication sequence. These errors can not completely be removed with the presented synchronization protocol.

The evaluation has shown, that the DAQ was able to synchronize its internal clock within an error of  $\pm 200\text{ ns}$  in 90 % of the time and accurately trace the PPS signal. Consequently, the final design of the FlockDAQ was able to provide timestamps on two nodes with an average synchronization error of  $163\text{ ns}$ .

## 6.2. Outlook

### Synchronization Protocol

The uncertainty of the timestamps has still a notable effect on the synchronization error. To improve the protocol further, these uncertainties need to be compensated for. For instance, with the use of statistical tools or modelling the synchronization process as a optimization problem.

### FlockDAQ

To fully integrate the FlockDAQ board into the existing FlockLab setup, a software adaptation is needed. The back-end server needs to be able to run tests with the FlockDAQ and collect and handle the data.





# A

## Appendix

### A.1. Operation Manual

#### A.1.1. Programming CC430 and FPGA

The four FTDI ports can be found under `/dev/flocklab/usb/daqX`, where  $X = \{1, 2, 3, 4\}$ .

##### CC430

As mentioned in section 4.2.4, port 1 (daq2) is connected to the CC430's UART, RESET and TEST pins. To flash a new image, the python-based `msp430-python-tools` can be used, in particular the module `msp430.bs15.uart`.

For instance:

```
01 python -m msp430.bs15.uart -p /dev/flocklab/usb/daq2 -e -S -V -P image.hex
```

##### FPGA

The JTAG pins of the FPGA are connected to port 0 (daq1). The application `xc3sprog` is a possible tool to configure a bitstream. It's a suite of utilities especially designed for programming Xilinx FPGAs using an FTDI based JTAG connection.

The following command configures the FPGA with the DAQ bitstream file:

```
01 xc3sprog -c ft4232h -v -p 0 DAQ_1_3.bit
```

For installing `xc3sprog` on the Gumstix, the library `libftdi` is required. Both can be added by installing following packages:

```
01 opkg install libftdi_0.18-r0.9_armv5te.ipk
02 opkg install xc3sprog_768-r0.9_armv5te.ipk
```

### A.1.2. FPGA UART

The DAQ can be configured and provides timing information over UART.

#### Configuration

The following sequence presents all steps to start a FlockDAQ test.

- Remove FlockLab and ADC modules

```
01 mmmod flocklab_gpio_monitor;
02 mmmod flocklab_ostimer;
03 mmmod flocklab_powerprof;
04 mmmod ads1271;
```

- Reset DAQ

```
01 echo "out set" > /proc/gpio/GPIO76
02 echo "out clear" > /proc/gpio/GPIO76
```

- Setting routing through off: An overview of all commands and data packets for configuring the DAQ can be found in [2, chapter 4.2.2]. The python script `fpga_daq_uart.py` is a simple serial handler, which writes the commands to the DAQ.

```
01 python fpga_daq_uart.py route off
```

- Configure the DAQ

```
01 python fpga_daq_uart.py configure <8 Tracing bits> <SampleDivider>
```

- Load the adapted ADC module

```
01 modprobe ads1271
```

- Start the test

```
01 python fpga_daq_uart.py start on
```

- After a test is done, the DAQ needs to be turned off and the direct routing of the signals on

```
01 python fpga_daq_uart.py start off
02 python fpga_daq_uart.py route on
```

Parameter	Header	Description
hccnt	0000'0001	CLK count between last and current PPS
offset	0000'0010	Difference between and PPS and full second
new diff	0000'0011	Current difference between PPS and 100MHz
avg diff	0000'0100	Average difference between PPS and 100MHz over 8 s
diff off cnt	0000'0101	Number of cycles to compensate
s_factor	0000'0111	Spread factor: number of cycles after a cycle is compensated
full_s	0000'0110	Full second counter

Tab. A.1.: FPGA timing parameters and corresponding UART package headers

### Timing Debug Output

Everytime the DAQ receives a PPS, it writes some information about the timing performance to the UART output. Each parameter consist of a 1 byte long header field and a 4 byte long data field. Table A.1 describes all the parameters. After every header follow the data bytes, which are indicated with a '1' at the MSB, the following 7 bits hold the data.

### A.1.3. u-blox GPS Configuration

The GPS devices by u-blox, in particular LEA-6T and LEA-6P, offer various options to configure their PPS output. There are two ways to change the configurations of an u-blox device, either with the u-blox software *u-center* or via writing commands to the device over a serial port.

#### u-center

*u-center* is an evaluation software, which can be used to write configuration to a u-blox device. Of particular interest for this work are the time-pulse settings, Fig. A.1 shows a screenshot of the configuration window. Under the tab Timepulse5 (TP5) following parameters can be set:

- **Period:** Specifies the period of the time-pulse in  $\mu s$ . Can also be specified as frequency in *Hz*.
- **Length:** Defines the length of the active phase of the time-pulse.
- **GNSS lock:** If the GPS has no locked connections to any satellites, it still outputs a time-pulse signal. Frequency and length for this

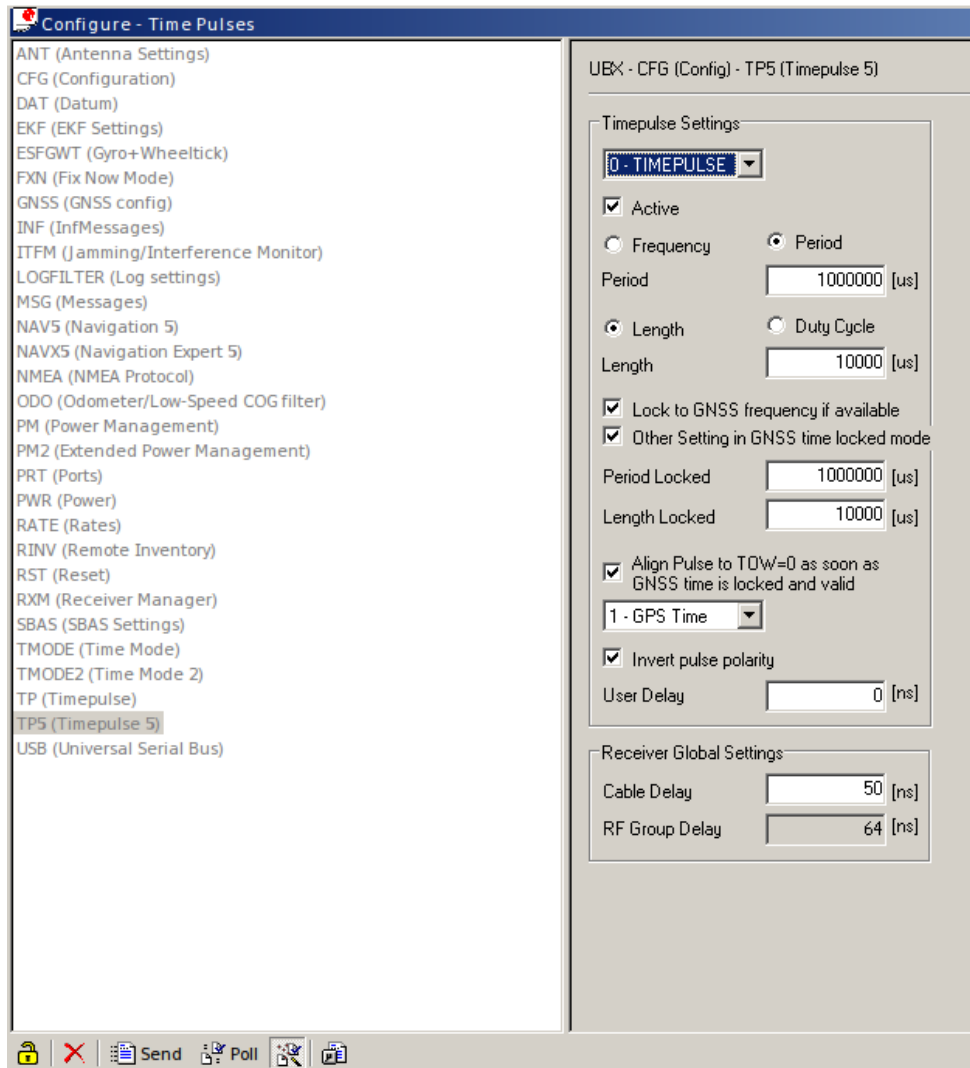


Fig. A.1.: TP5 (Timepulse 5) configuration window

case can be set separately.

- **Time source:** Selects the source on which the time-pulse depends, GPS time is recommended (GPS time and UTC are not the same and UTC provides a lower accuracy than the GPS time [19]).
- **Pulse polarity:** If "Invert pulse polarity" is set, the time-pulse is indicated by a rising edge and is set back to GND after the time set in "Length".
- **User delay:** A user-defined delay which will be added to the time-pulse. Not needed, therefore set to 0.
- **Cable delay:** The device compensates for the delay induced by the length of the cable. All the devices use a 5 m cable, which approximately add a 50 ns delay to the time-pulse.

The LEA-6T and LEA-6P devices feature also a service called "SBAS (Satellite Based Augmentation Systems)", a complex system to calculate integrity and correction data for GPS devices. This service should always be turned off to ensure a stable time-pulse, Fig.A.2 shows the corresponding configuration window. To improve the performance and

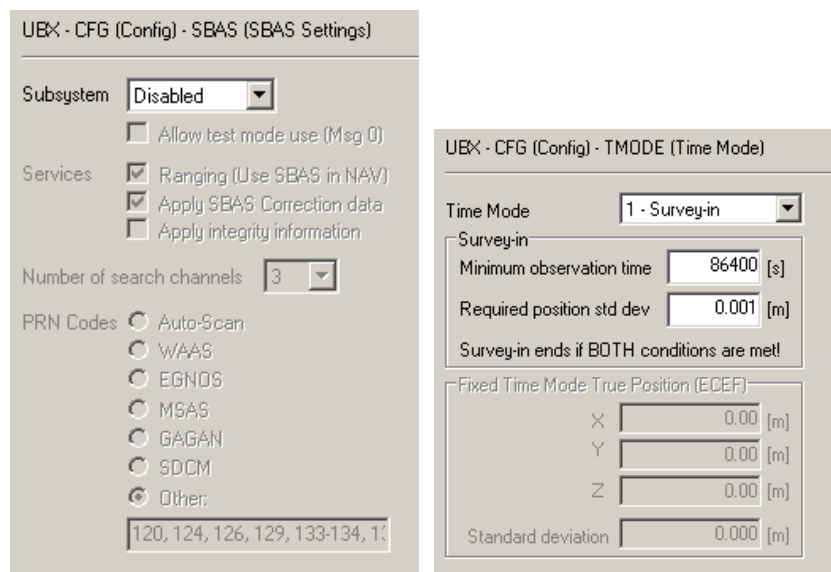


Fig. A.2.: Left: SBAS (SBAS settings) configuration window. Right: TMODE (Time Mode) configuration window

time-pulse accuracy even more, a service called "Survey-In" can be turned on. Stationary GPS devices are able to calculate their position with high accuracy, however this takes some time. Fig.A.2 shows the two parameters for the survey-in mode:

- **Minimum observation time:** This is the time window for calibrating the GPS device. After this time window ends, the GPS receiver uses the calculated position for an accurate time-pulse generation. According to the manufacturer the survey-in calibration should take at least 24 hours.
- **Required position std dev:** The survey-in is only completed if the standard deviation of the position measurements is smaller than this upper bound. A recommended value is 1 mm.

The problem with using the survey-in is the long time it takes to guarantee an accurate time-pulse, because after every power-cycle the calibration is reset. But in general the GPS is not turned off.

After all parameters are set and the GPS device is connected to a computer, the configuration is transmitted by pressing "Send" in the bottom left corner of the window, as in Fig. A.1.

### **Commands over serial port**

Instead of using u-center, the configurations can also be transmitted to the device by sending the corresponding commands over a serial port. A description of the data structures and the individual commands can be found in [22, chapter 22]. A much easier way is to use u-center and display the binary console (press F7). The parameters can now be set under the corresponding tabs and by sending the configuration, the packet will be displayed in the binary console in hexadecimal format. This information can now be send to the device with any serial port handler, e.g. a simple python script.

## A.2. Plots

### Synchronization Protocol

For completeness: Histograms of final synchronization error for all nodes.

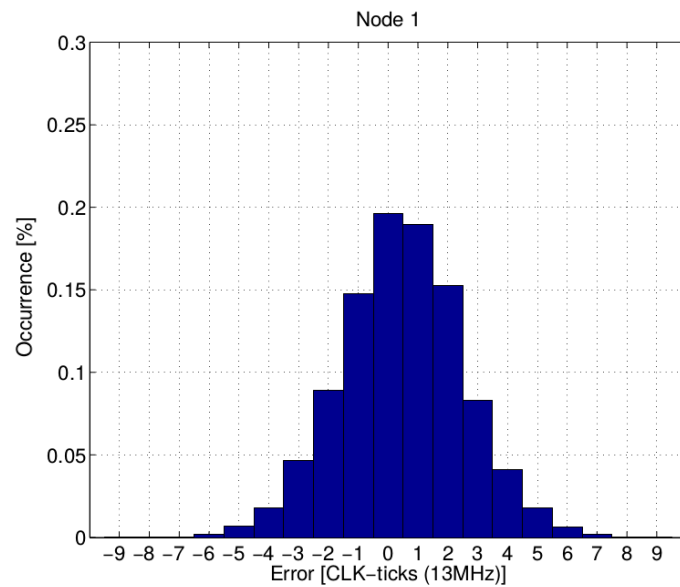


Fig. A.3.: Synchronization error, Node 1

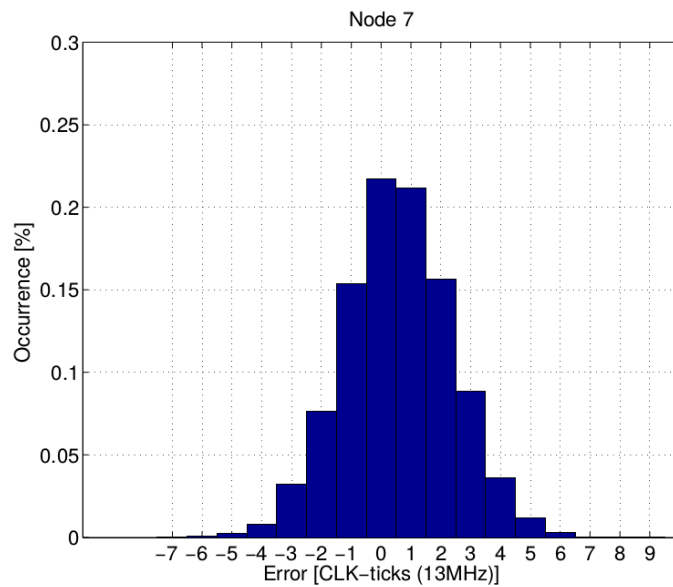


Fig. A.4.: Synchronization error, Node 7

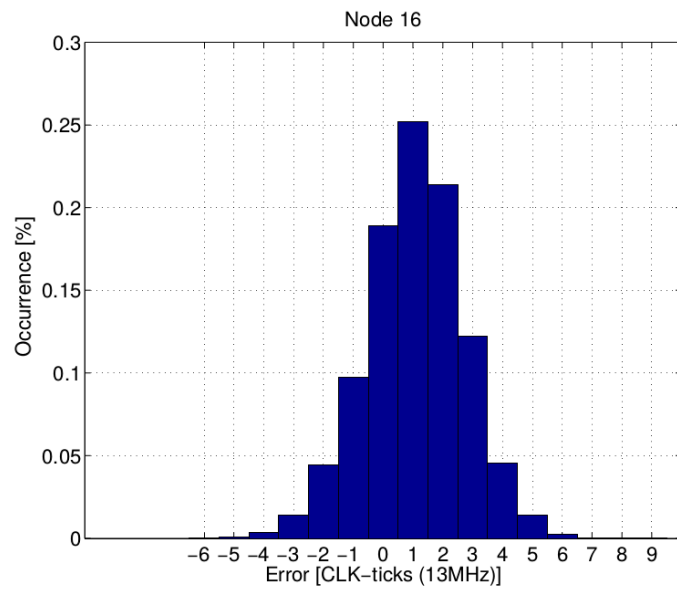


Fig. A.5.: Synchronization error, Node 16

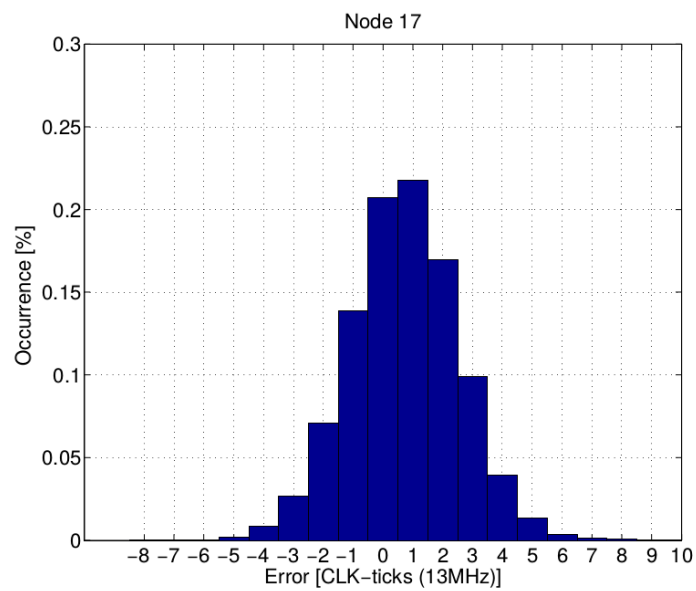


Fig. A.6.: Synchronization error, Node 17



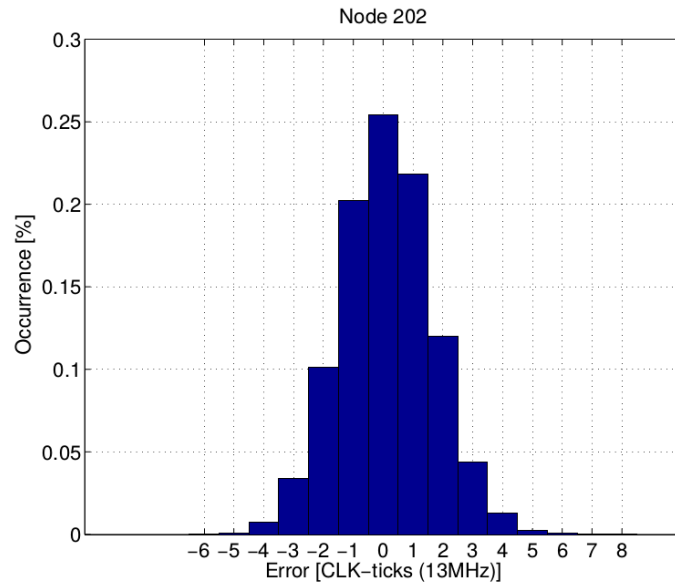


Fig. A.7.: Synchronization error, Node 202

### FlockDAQ Performance

Histograms of the FlockDAQ performance for node 1.

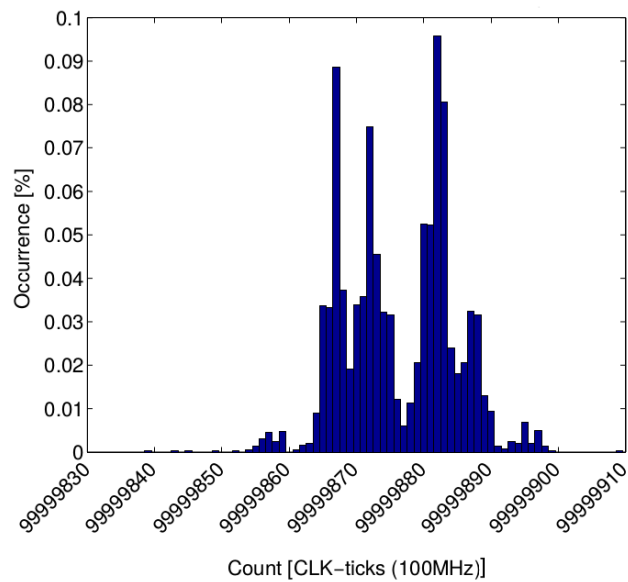


Fig. A.8.: Difference between two consecutive PPS captures in CLK-ticks, Node 1

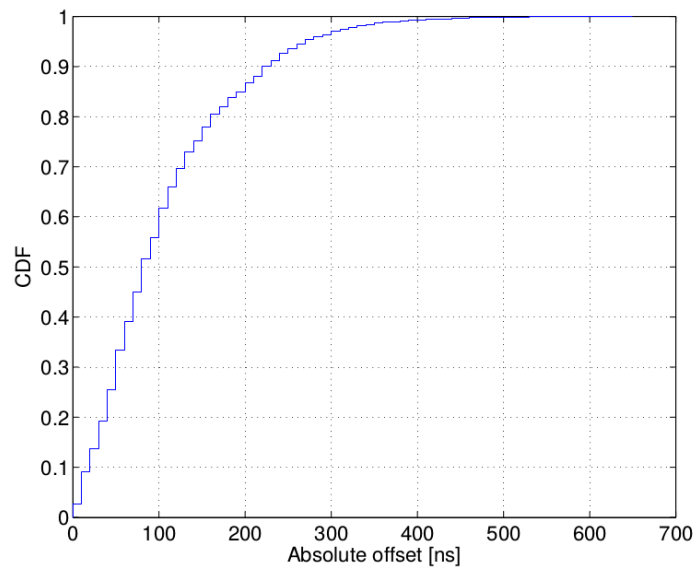


Fig. A.9.: CDF of the absolute offset between PPS occurrence and internal full second counter in ns, Node 1

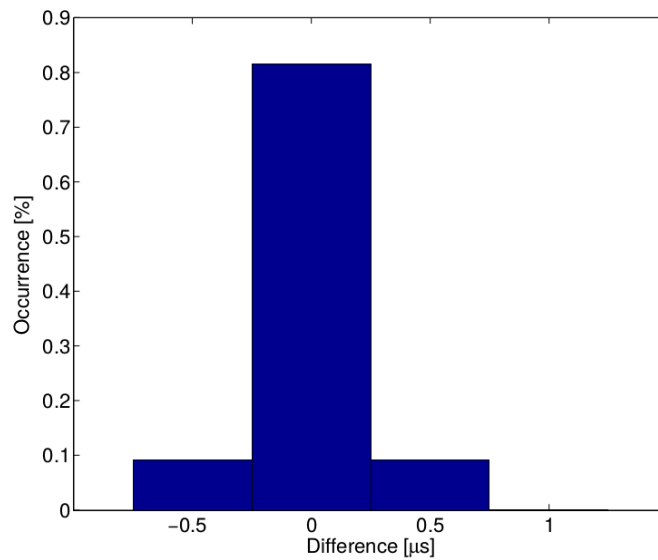
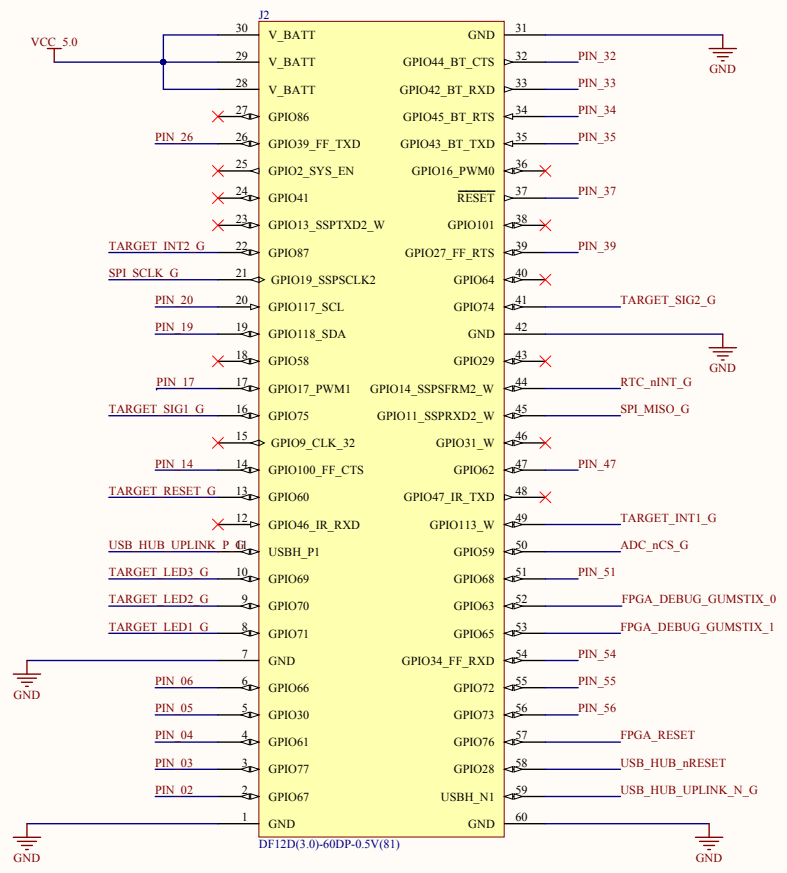


Fig. A.10.: Difference of GPS PPS capture times to 1 second in  $\mu s$ , Node 1

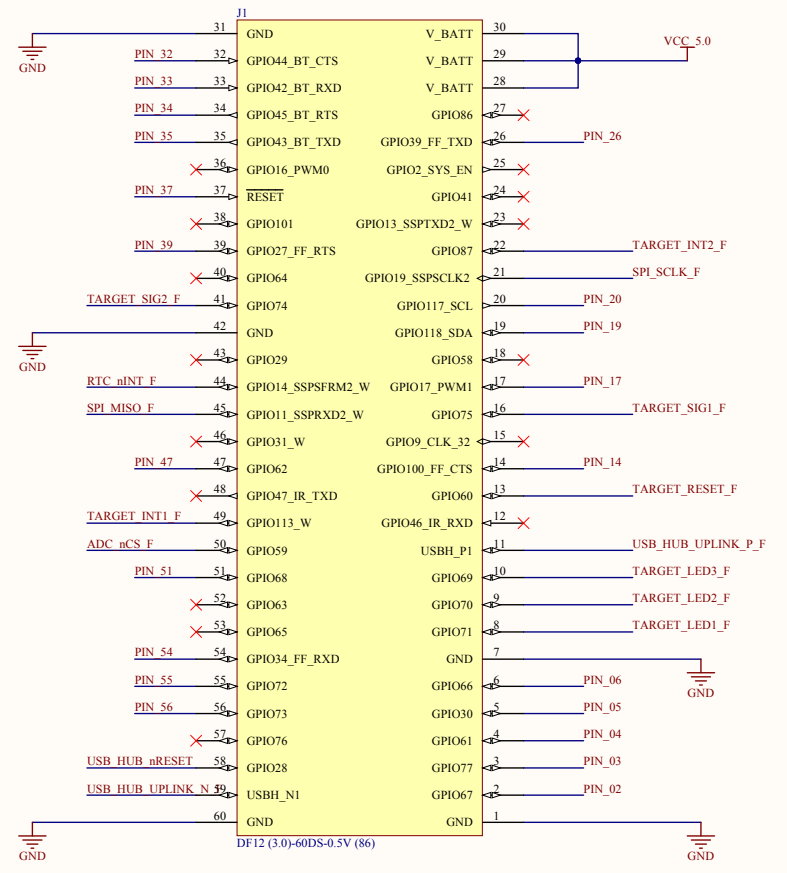
## **A.3. PCB Schematics**

A

Connecting to Gumstix, FEMALE



Connecting to Flockboard, MALE




C

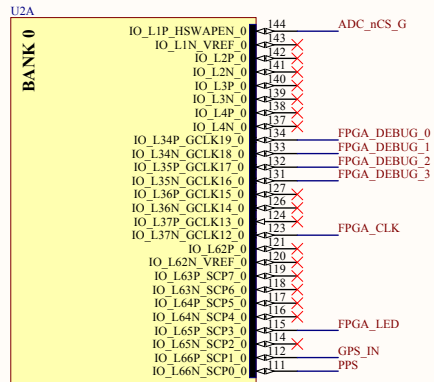
C

D

D

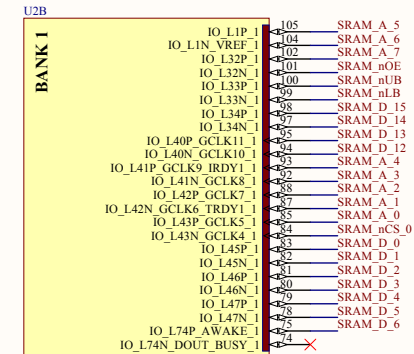
 <p>Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich</p>		Zeichnungs Titel:		
		<p><b>DAQ Board - Verdex Connectors</b></p>		
Zeichnungsnummer: 1	Rev: 1.0	Format: A3	Institut: TIK, ETH Zürich	Projekt: verdex_connector.SchDoc
Datum: 02.07.2014 11:11:45		Ersteller: Balz Maag		Seite 1 von 8
File: C:/Users/btbnode/SVN/permasense/engineering/pcb/projects/0009_flocklab/0004_daq_board/rev1.0/verdex_connector.SchDoc				

Bank 0, Debug and free pins



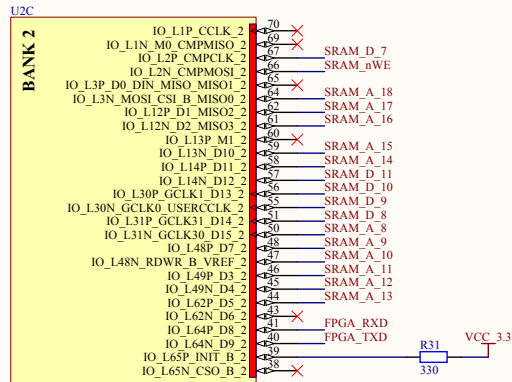
XC6SLX9-3TQG144C

Bank 1, SRAM



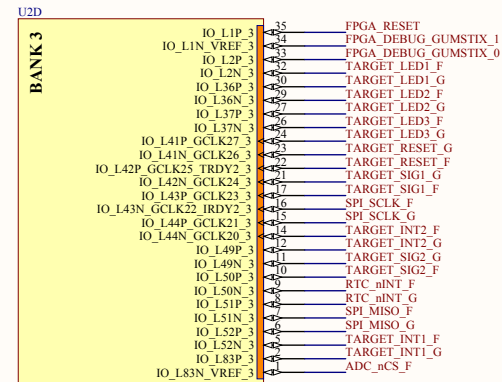
XC6SLX9-3TQG144C

Bank 2, SRAM

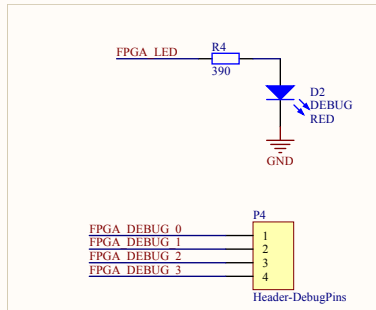
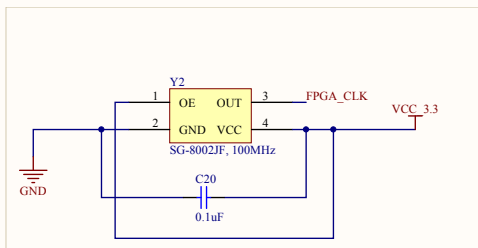


XC6SLX9-3TQG144C

Bank 3, Actuation and Tracing I/O



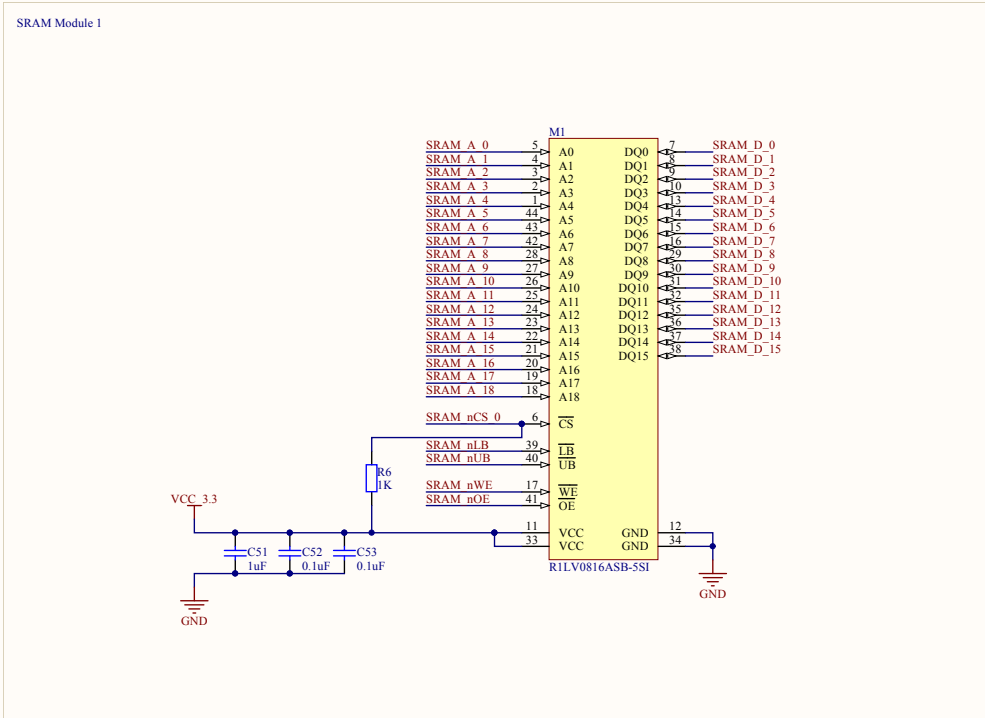
XC6SLX9-3TQG144C



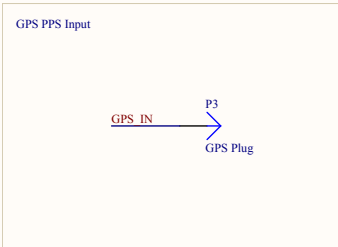
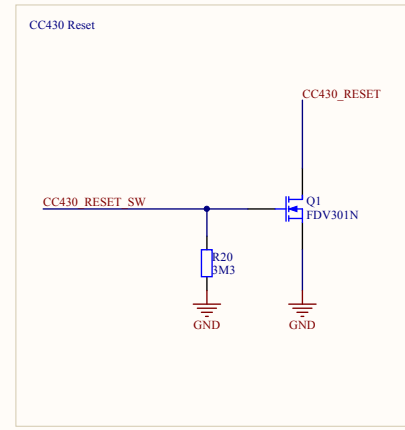
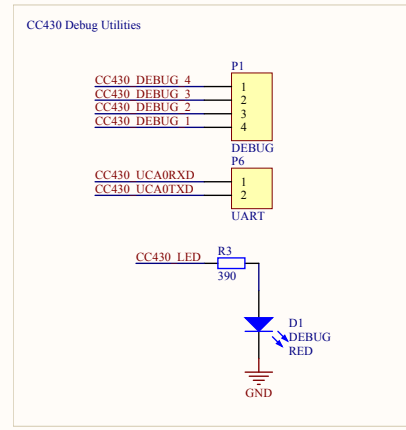
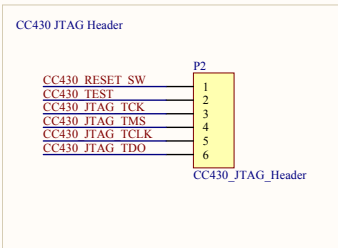
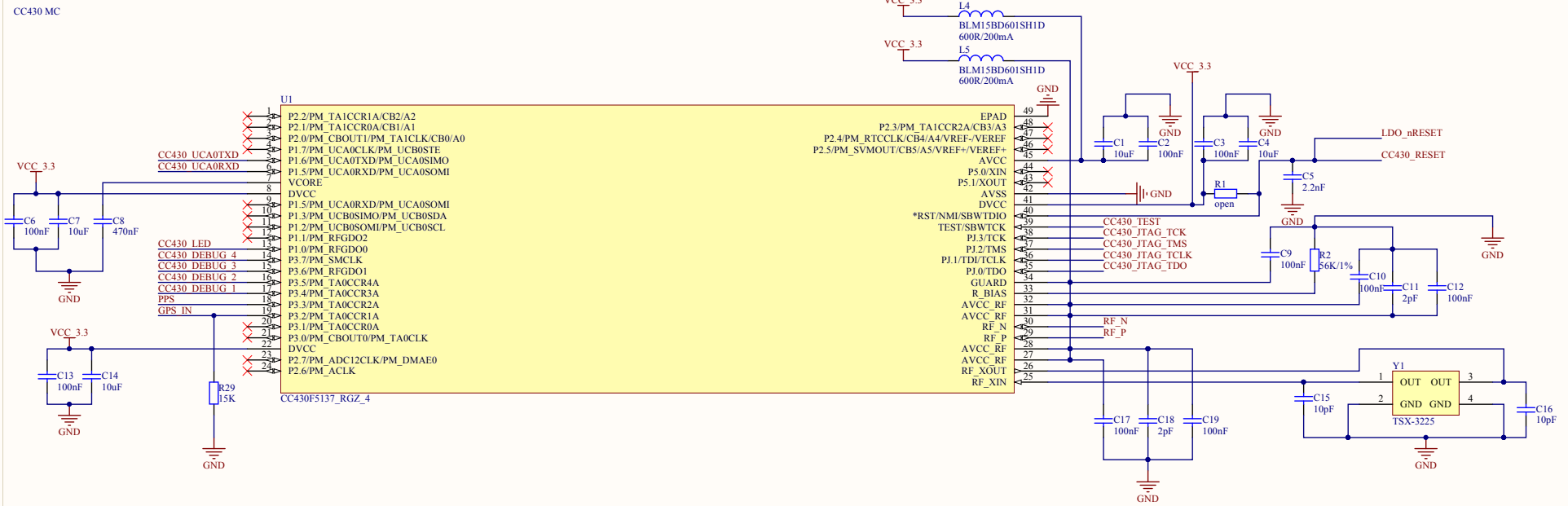
**ETH**  
Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

Zeichnungstitel:  
**DAQ Board - FPGA Banks**

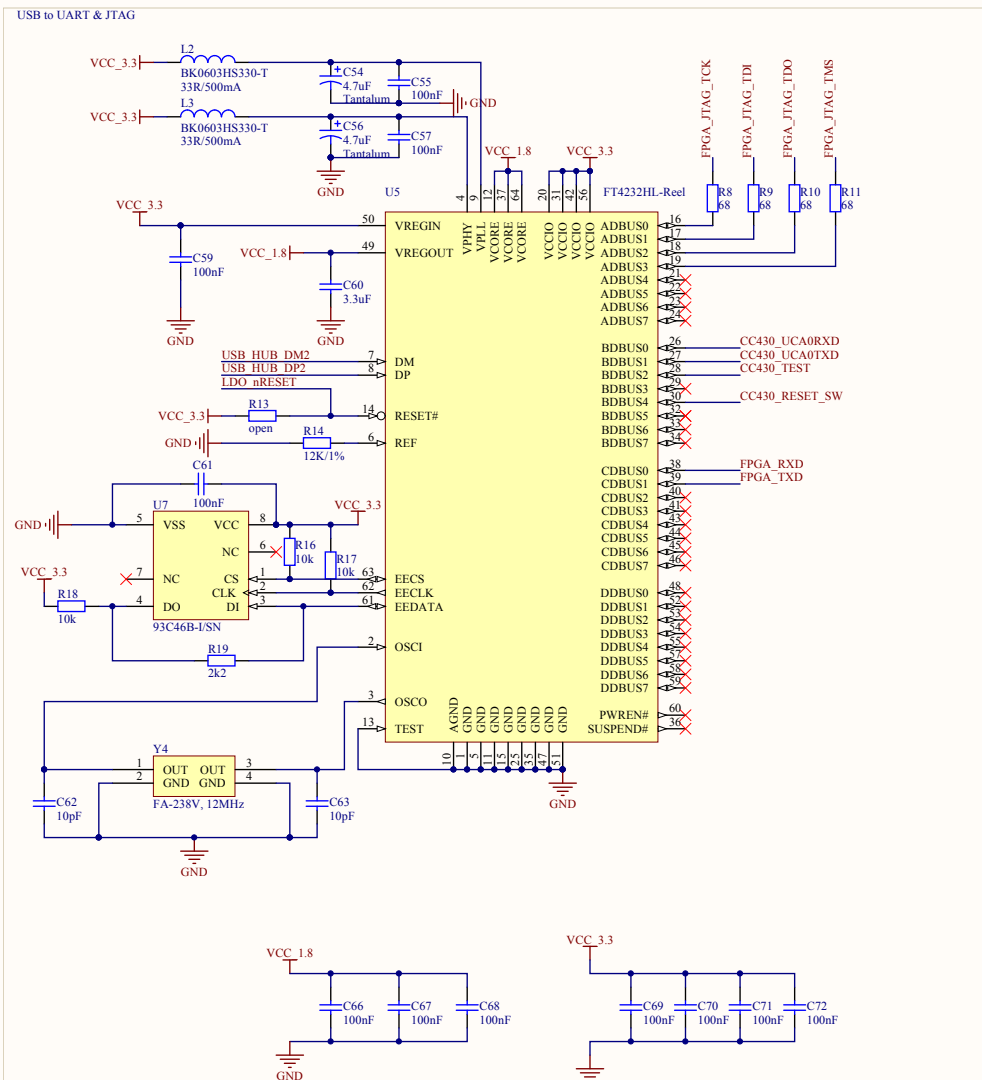
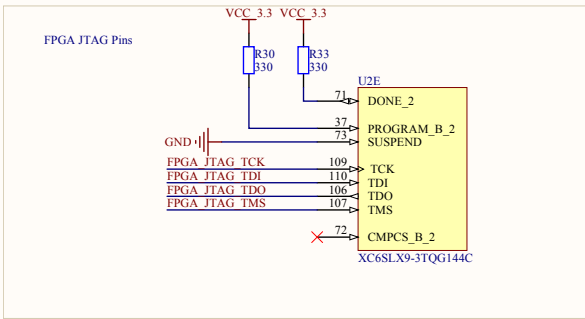
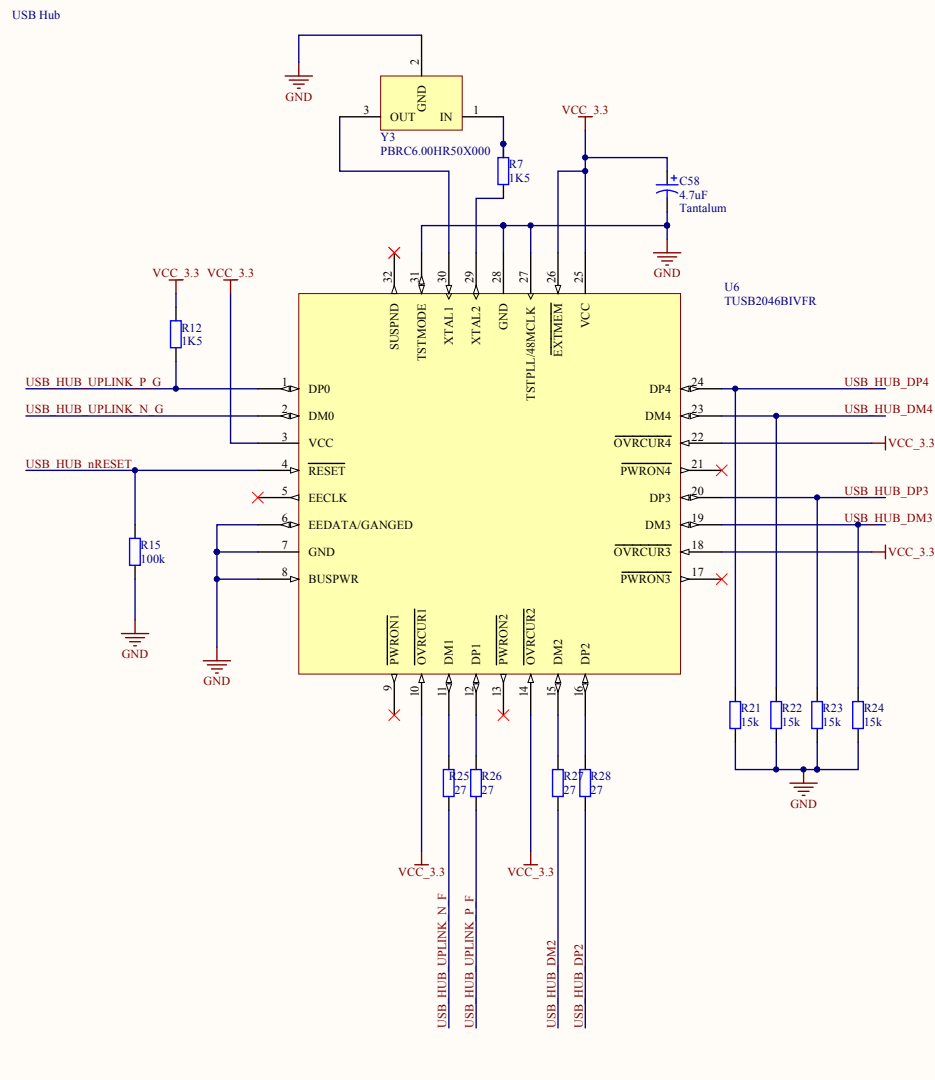
Zeichnungsnummer: 2	Rev: *	Format: A3	Institut: *	Projekt: fpga_banks.SchDoc
Datum: 02.07.2014 11:11:45		Ersteller: Balz Maag	Seite 2 von 8	
File: C:/Users/bnode/SVN/permasense/engineering/pcb/projects/0009_flocklab/0004_daq_board/rev1.0/fpga_banks.SchDoc				



 <b>Eidgenössische Technische Hochschule Zürich</b> <b>Swiss Federal Institute of Technology Zurich</b>		Zeichnungs-Titel:	
		<b>DAQ Board - SRAM</b>	
Zeichnungsnummer: 4	Rev: *	Format: A3	Institut: *
Datum: 02.07.2014 11:11:45		Ersteller: Balz Maag	Projekt: sram.SchDoc
Seite 4 von 8		File: C:\Users\btnode\SVN\permasense\engineering\pcb\projects\0009_flocklab\0004_daq_board\rev1.0\sram.SchDoc	



		Zeichnungstitel:	
		<b>DAQ Board - CC430</b>	
Zeichnungsnummer: 5	Rev: 1.0	Format: A3	Institut: *
Datum: 02.07.2014 11:11:45	Ersteller: Balz Maag	Projekt: cc430.SchDoc	
File: C:\Users\btbnod\SVN\permasense\engineering\pcb\projects\0009_flocklab\0004_daq_board\rev.1.0\cc430.SchDoc		Seite 5	von 8

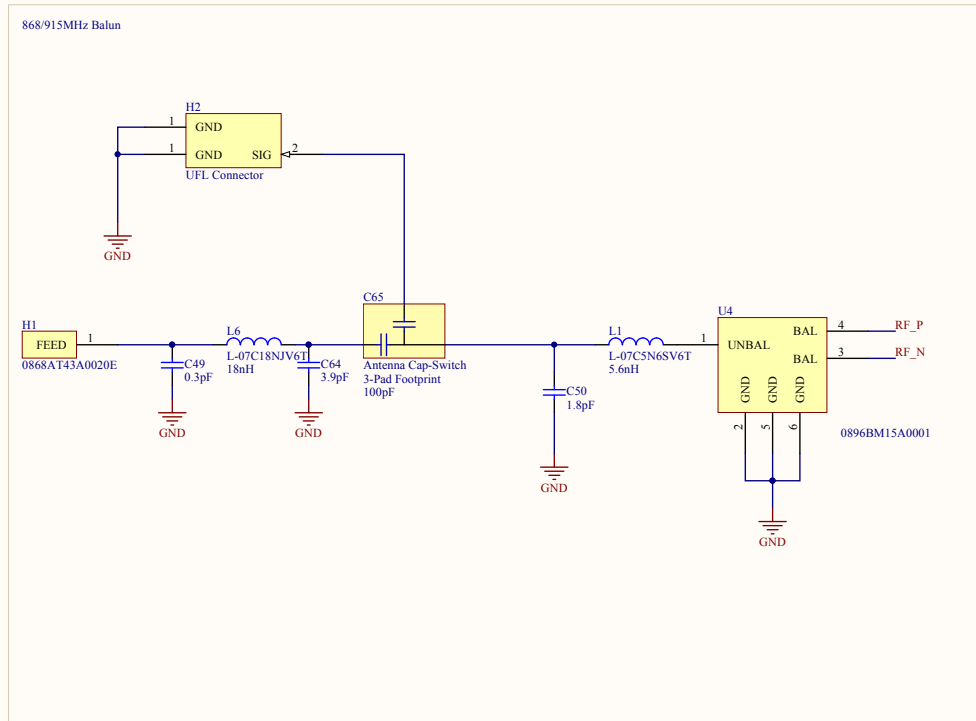



**ETH**  
Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

Zeichnungstitel:  
**DAQ Board - USB**

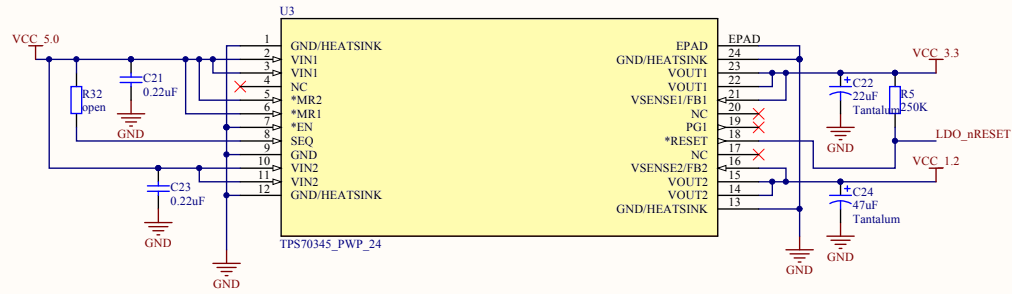
Zeichnungsnummer: 6	Rev: *	Format: A3	Institut: *	Projekt: usb.SchDoc
Datum: 02.07.2014 11:11:46			Ersteller: *	Seite 6 von 8
File: C:/Users/btnode/SVN/permasense/engineering/pcb/projects/0009_flocklab/0004_daq_board/rev1.0/usb.SchDoc				



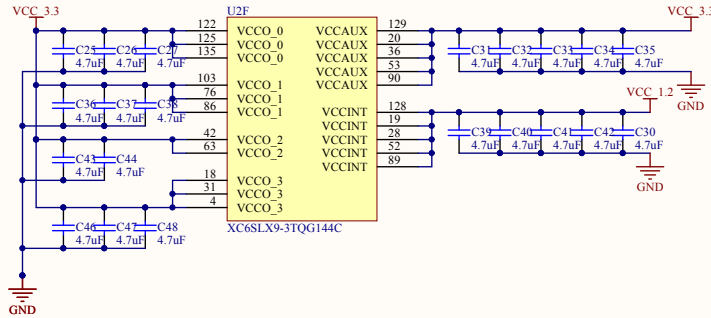


 <b>Eidgenössische Technische Hochschule Zürich</b> <b>Swiss Federal Institute of Technology Zurich</b>		Zeichnungs Titel:		
		*		
Zeichnungsnummer: 7	Rev: *	Format: A3	Institut: *	Projekt: rf_frontend.SchDoc
Datum: 02.07.2014 11:11:46			Ersteller: *	Seite 7 von 8
File: C:/Users/btmode/SVN/permasense/engineering/pcb/projects/0009_flocklab/0004_daq_board/rev1.0/rf_frontend.SchDoc				

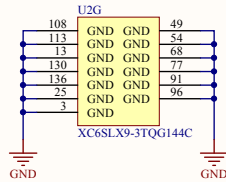
Voltage Regulator, 5V -> 3.3V and 1.2V



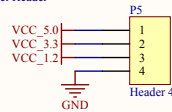
FPGA Power Inputs with decoupling cap-rails



FPGA Power GND Pins



Voltage Level Header



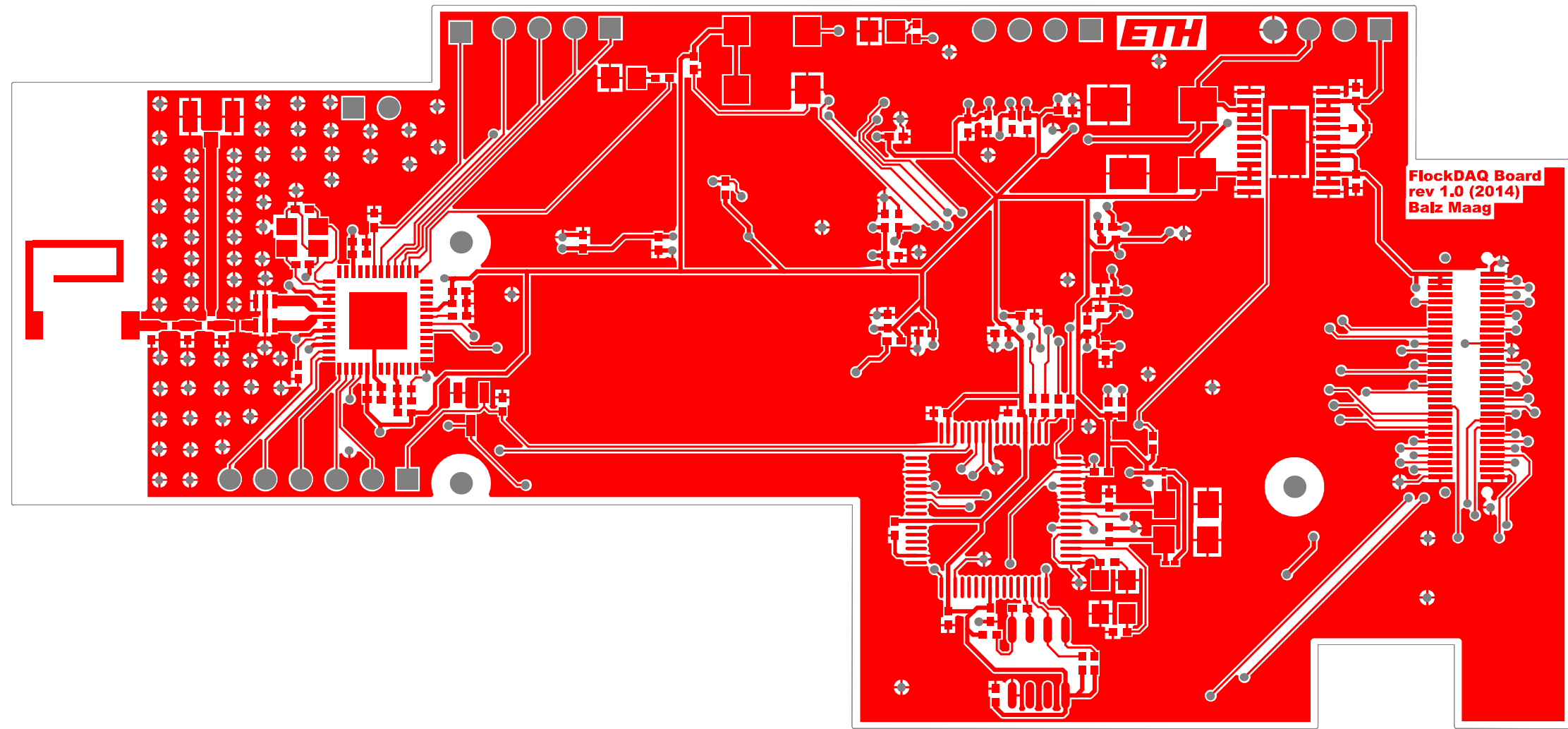
Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

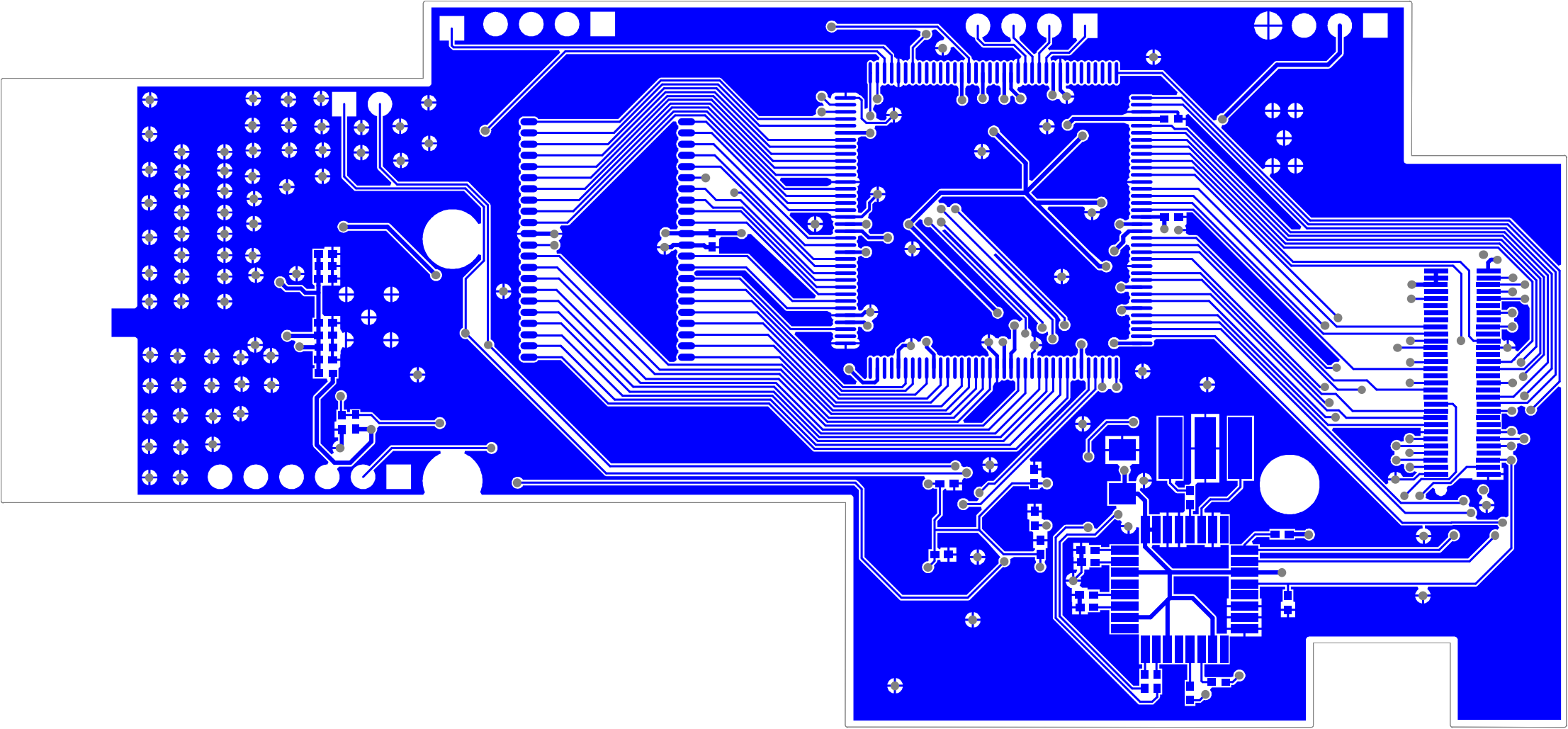
Zeichnungs Titel:  
**DAQ Board - Power**

Zeichnungsnummer: 8	Rev: 1.0	Format: A3	Institut: TIK	Projekt: power.SchDoc
Datum: 02.07.2014 11:11:46		Ersteller: Balz Maag	Seite 8 von 8	
File: C:/Users/btmode/SVN/permasense/engineering/pcb/projects/0009_flocklab/0004_daq_board/rev1.0/power.SchDoc				

**ETH**

**FlockDAQ Board  
rev 1.0 (2014)  
Balz Maag**





# Bill of Materials

DAQ Board - CC430

Source Data From: daq\_board.PrjPCB  
 Project: daq\_board.PrjPCB  
 Variant: None

Creation Date: 02.07.2014 11:12:05  
 Print Date: 02-Jul-14 11:12:12 AM

Footprint	Comment	LibRef	Designator	Description	Quantity
0402	10uF	Cap	C1, C4, C7, C14	Capacitor, Ceramic	4
0402	100nF	Cap	C2, C3, C6, C9, C10, C12, C13, C17, C19, C55, C57, C59, C61, C66, C67, C68, C69, C70, C71, C72	Capacitor, Ceramic	20
0402	2.2nF	Cap	C5	Capacitor, Ceramic	1
0402	470nF	Cap	C8	Capacitor, Ceramic	1
0402	2pF	Cap	C11, C18	Capacitor, Ceramic	2
0402	10pF	Cap	C15, C16, C62, C63	Capacitor, Ceramic	4
0402	0.1uF	Cap	C20, C52, C53	Capacitor, Ceramic	3
0402	0.22uF	Cap	C21, C23	Capacitor, Ceramic	2
CAP-TANTAL	22uF	Cap Pol1	C22	Tantalum Capacitor	1
C					
CAP-TANTAL	47uF	Cap Pol1	C24	Tantalum Capacitor	1
D					
0402	4.7uF	Cap	C25, C26, C27, C30, C31, C32, C33, C34, C35, C36, C37, C38, C39, C40, C41, C42, C43, C44, C46, C47, C48	Capacitor, Ceramic	21
0402	0.3pF	Cap	C49	Capacitor, Ceramic	1
0402	1.8pF	Cap	C50	Capacitor, Ceramic	1
0402	1uF	Cap	C51	Capacitor, Ceramic	1
CAP-C3216M	4.7uF	Cap Pol1	C54, C56, C58	Tantalum Capacitor	3
P					
0402	3.3uF	Cap	C60	Capacitor, Ceramic	1
0402	3.9pF	Cap	C64	Capacitor, Ceramic	1
Antenna Cap	100pF	Antenna Cap-Switch	C65	Antenna Cap-Switch, selection by soldering a Capacitor	1
Switch					
GD2012-0805	RED	LED2	D1, D2	Led with a specific color	2
0868A143A0	0868A143A0020	H1	H1	Chip Antenna, 868MHz	1
020E	E	E			
UFL_SMD	UFL Connector	UFL Connector	H2		1
Hirose60Verd	DF12 (3.0)-60DS-0.5V (86)	_R	J1	Verdex Connector Receptacle	1
ex_R					
Hirose60Verd	DF12(3.0)-60DP-0.5V(81)	_H	J2	Verdex Connector Header	1
ex_H3					
MURATA_IN	L-07CSN6SV6T	Inductor	L1	Inductor	1
D_0402					
0603	BK0603HS330-T	Inductor	L2, L3	Inductor	2
0402-A	BLM15BD601S	Inductor	L4, L5	Ferrite Bead	2
H1D					
0402-A	L-07C18NUV6T	Inductor	L6	Inductor	1
PTSB0044G	R1LV0816ASB-	M1	M1	R1LV0816ASB Series 8Mb Advanced LPSRAM (512k word x 16bit)	1
D-B_L	SSI	SSI			
HDR1X4	DEBUG	Header 4	P1	Header, 4-Pin	1
HDR1X6	Header 6	Header 6	P2	Header, 6-Pin, Socket for small signals	1
PIN1	SmallPlug	SmallPlug			
HDR1X4	GPS Plug	Plug	P3	Plug	1
HDR1X4	Header-DebugPins	Header 4	P4	Header, 4-Pin	1
HDR1X4	Header 4	Header 4	P5	Header, 4-Pin	1
HDR1X2	Header 2	Header 2	P6	Header, 2-Pin	1
SOT95P230-3M	FDV301N	FDV301N	Q1	MOSFET, N, Digital, 0.22A	1
0402	open	Res2	R1, R13, R32	Resistor	3
0402	56K1%	Res2	R2	Resistor	1
0402	390	Res2	R3, R4	Resistor	2
0402	250K	Res2	R5	Resistor	1
0402	1K	Res2	R6	Resistor	1
0402	1K5	Res2	R7, R12	Resistor	2
0402	68	Res2	R8, R9, R10, R11	Resistor	4
0402	12K1%	Res2	R14	Resistor	1
0402	100k	Res2	R15	Resistor	1
0402	10k	Res2	R16, R17, R18	Resistor	3
0402	2k2	Res2	R19	Resistor	1
0402	3M3	Res2	R20	Resistor	1
0402	15k	Res2	R21, R22, R23, R24, R29	Resistor	5
0402	27	Res2	R25, R26, R27, R28	Resistor	4
0402	330	Res2	R30, R31, R33	Resistor	3
RC248_4P1X	CC430F5137_R	CC430F5137_R	U1	CC430	1
4P1	GZ_4	GZ_4			
TQG144_L	XC6SLX0-3TQG144C	XC6SLX0-3TQG144C	U2	Spartan-6 LX 1.2V FPGA, 102 User I/Os, 144-Pin TQFP (0.5mm Pitch), Speed Grade 3, Commercial Grade, Pb-Free	1
PWP24_4P6	TPS70345_PW	TPS70345_PW	U3	Dual Channel LDO (3.3V/1.2V)	1
8XP4	P_24	P_24			
CC110L_BAL	0896BM15A000	0433BM15A000	U4	Johanson Technology 433MHz Balun for TI CC110L	1
UN	1	1			
LQFP-64_L	FT4232HL-Reel	FT4232HL-Reel	U5	Quad High Speed USB To Multipurpose UART/MPSSSE IC, LQFP-64, Tape and Reel	1
QFP80P900X	TUSB2046BIV	TUSB2046B	U6	4 Port USB Hub	1
900-32M	R				
SOIC-SN8_L	93C46B-VSN	93C46B-VSN	U7	1K, 64x16-bit, 5.0V Microwire Serial EEPROM, 8-Pin SOIC 150mil, Industrial Temperature	1
TSX-3225	TSX-3225	TSX-3225	Y1	Oscillator, Epson, 26MHz for CC430	1
SG-8002	SG-8002JF	SG-8002JF	Y2	Oscillator, Epson, Programmable 100MHz	1
PBRC-H	PBRC6_00HR50	PBRC-H	Y3	Crystal Oscillator, 6MHz, Internal Caps	1
FA-238V	FA-238V_X000	FA-238V	Y4	Crystal, Epson	1
12MHz					

Approved	Notes



# Bibliography

- [1] *IEEE standard 802.15.4-2003*, 2003.
- [2] B. Dissler. High-speed data-acquisition for flocklab. Master's thesis, ETH Zürich, 2014.
- [3] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with glossy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 73–84, April 2011.
- [4] Future Technology Devices International Ltd. *FT2232H Quad High Speed USB to Multipurpose UART/MPSSE IC*.
- [5] F. Grilli and A. Jain. *Base station synchronization for handover in a hybrid GSM/CDMA network*. Patent. US 20030002525 A1, 2003.
- [6] Johanson Technology. *P/N 0868AT43A0020, 868MHz Antenna*.
- [7] Johanson Technology. *P/N 0896BM15A0001, 868/915 MHz Impedance Matched/Balun/LPF Integrated Component for T.I. CC110X, CC111X, CC113X and CC115X, CC110L, CC113L, CC115L and CC430*.
- [8] C. Lenzen, P. Sommer, and R. Wattenhofer. Pulsesync: An efficient and scalable clock synchronization protocol. *Networking, IEEE/ACM Transactions on*, 2014.
- [9] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks, IPSN '13*, pages 153–166, New York, NY, USA, 2013. ACM.
- [10] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems, SenSys '04*, pages 39–49, New York, NY, USA, 2004. ACM.
- [11] D. Mills. Internet time synchronization: the network time protocol. *Communications, IEEE Transactions on*, 39(10):1482–1493, Oct 1991.

- [12] Olimex. *MSP430-CCRF development board, User's manual.*
- [13] Renesas. *R1LV0816ASB – 5SI, 7SI, 8Mb Advanced LPSRAM (512k word x 16bit).*
- [14] Samsung Electro-Mechanics. *Multi Layer Ceramic Capacitor (MLCC), CL05A475MQ5NRNC, Characteristics.*
- [15] Texas Instruments. *CC430 Family User's Guide.*
- [16] Texas Instruments. *MSP430 SoC With RF Core.*
- [17] Texas Instruments. *TPS70345, Dual-Output, Low Dropout Voltage Regulators With Integrated SVS For Split Voltage Systems.*
- [18] Texas Instruments. *TUSB2046B, 4-Port Hub For The Universal Serial Bus With Optional Serial EEPROM Interface.*
- [19] u-blox. <http://www.u-blox.ch/en/timing.html>. u-blox Webpage, September 2014.
- [20] u-blox. *GPS-based Timing, Considerations with u-blox 6 GPS receivers, Application Note.*
- [21] u-blox. *LEA-6, u-blox 6 GPS Modules Data Sheet.*
- [22] u-blox. *u-blox 6 Receiver Description, Including Protocol Specification.*
- [23] Xilinx. *Spartan-6 Family Overview.*