Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Department of Information Technology and Electrical Engineering

Spring Semester 2018

Reliable 3rd Generation Data Collection

Master Thesis

Markus Wegmann
mwegmann@student.ethz.ch

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

Supervisors: Jan Beutel, janbeutel@ethz.ch
Roman Trüb, roman.trueb@tik.ee.ethz.ch
Reto Da Forno, rdaforno@ee.ethz.ch

Professor: Prof. Dr. Lothar Thiele, thiele@ethz.ch

31st August, 2018

Abstract

In this thesis a new wireless communication platform based on Semtech's next generation *LoRa* radio chip SX1262 is being introduced in the context of the *PermaSense* wireless sensor network project and the *dual-processor platform*. By systematic evaluation of the radio chip's intrinsics we set the foundation for a new data acquisition protocol, called *LWB/Gloria*, which promotes long-range, high-throughput and low-power scenarios. The protocol is based on a concurrent-transmission flooding-mechanism inspired by Glossy. We then show an event-driven implementation of the protocol in hardware and as simulation.

Acknowledgements

My great thanks go to my supervisors, Jan Beutel, Roman Trüb, Reto Da Forno and colleagues at TIK for their great support and provision of instrumentation, lab facilities and the great *SOLA-Stafette*, my dear friend Dan Mugioiu of HSR for his great insights into the radio industry, the tuning of the RF path and *LoRa* details, Marco Zahner for the review of the RF path's layout and filter components, my colleagues and business partners at Technokrat for their patience, as well as the Semtech community and driver maintainers for a working and well-documented product.

Contents

Chapter 1

Introduction

Climate change and global warming have been and are rapidily transforming earth's environment since the industrial revolution of the 19th century. Its large scale impact among others might cause wildfires, blizzards, dangerous storms, the expansion of deserts, rise of the sea level, drastic changes in the biological ecosystems, and an increase in migration flows and geopolitical conflicts [1].

For the purpose of understanding, predicting and planning mitigations for the future development, scientists of all kinds are contributing to the cause. Data gathered in international collaboration serves as the scientific basis for factual reasoning as well as for early-warning systems.

One aspect of climate change research and geology is considered with the alpine cryosphere, in particular melting of the permafrost and glaciers which had their last high 21'000 years ago. While glaciers might be considered as a natural water reserve and support of the underlying rock, permafrost can enhance the structural durability of rocks and cliffs. As the mean temperatures are increasing in central europe, severe hangslides and rock falls can be the result, threatening mountain villages, reservoir dams and walls.

The more frequent thawing and freezing cycles of the upper permafrost layers contribute strongly to erosion, leading in increased cracks and weak points in the rock structure. A swiss reasearch initiative and consortium called *PermaSense* was founded therefore, "developing, deploying and operating wireless sensing systems customized for long-term autonomous operation in high-mountain environments."[2]

This chapter will give basic insight into the *PermaSense* concepts, current state of the wireless sensing network, introduce a new long-range based RF modulation technique, called *LoRa*, have a look on the established wireless protocols and conclude with the motivation for this thesis.

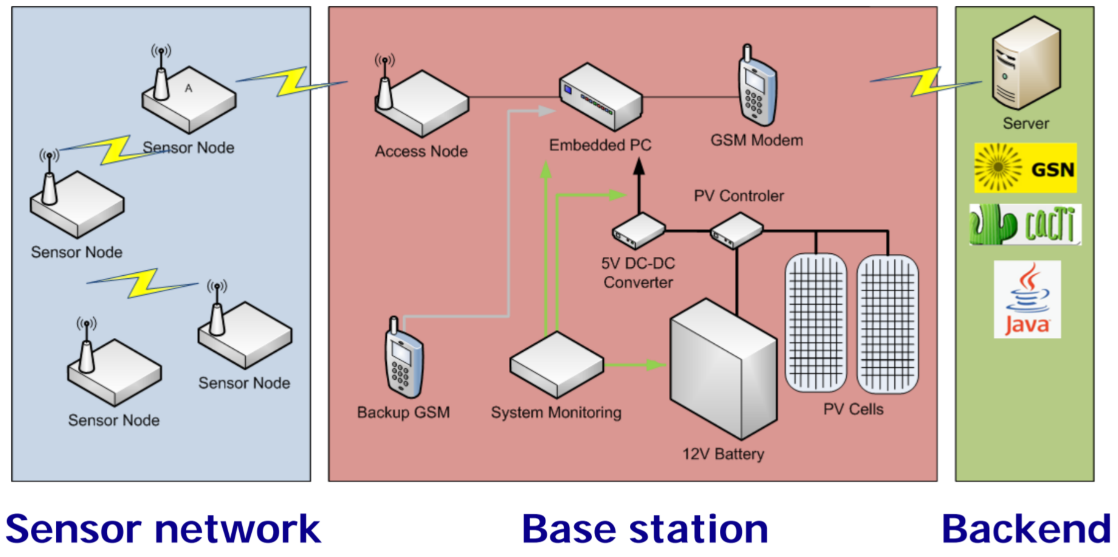


Figure 1.1: *PermaSense*'s network topology concept.[3]

1.1 PermaSense

The *PermaSense* consortium maintains several wireless sensing networks (WSN) in the swiss alps, including Matterhorn and Jungfraujoch region. Several dozen sensor nodes are distributed throughout areas of high interest and are transmitting their gathered data via a wireless link to a base station in or near this area. These areas can span some hundred meters up to some kilometers, with different distances between the sensor nodes and/or base station.

The base station is connected via a directed Wifi link to the Internet, and may upstream its gathered data to the *global sensory network* (GSN), which exposes a free and open database interface for researchers around the globe.

1.1.1 Sensor Node

The sensor nodes interface with different sensoric instruments as data source, depending on their location. An overview of available sensors is given by [4]. The latest generation sensor nodes also include geophones, which are recording the rock's accoustic response to geodynamic events up to several hundred hertz.

The default sensor node configuration is consisting of:

- Sensor Interface Board: Preamplifies and digitalizes the analog sensor probe signals itself or gathers the the data from digital sensor equipment via serial bus. The built-on non-volatile SDMMC Flash memory allows to store the data, mitigating network outages, and may serve for verification and data consistency checks. The application processor controls the sensor data flow and processing according to the nodes intended purpose.

- Communication Board: Serves as the RF MAC layer interface and backend for the wireless communication network. It includes its own processor and 868 MHz radio.
- Battery and power management unit: Delivers the power needed for all subsystems and includes the power fuel gauge. Optionally used for energy harvesting (solar power, Peltier elements, etc.).
- Ruggedized Enclosure: A robust and watertight sensor node metal housing ensures operability under extreme high-mountain conditions, such as rapid temperature changes, heavy rain, rock fall, lightning, rime, snow, ice, avalanches and sun exposure. An external *rubber ducky antenna* and conntectors for the peripherals are panel mounted directly on the box and are covered via a protective shoe. The protective shoe enclosure allows for a fast exchange of the inner sensor node box in case of maintenance by simply sliding it out of the fixture.

Frequent maintenance of the dispersed nodes is cumbersome and expensive due to transportation, equipment and staff costs. The sensor nodes are thus designed for three years unattended lifetime, in a $-40\,°C$ to $65\,°C$ ($\Delta T \leq 5\,°C/min$) environment. Data yield is required to be $\geq 99\,\%$. The battery has a capacity of 13 Ah which equals roughly a power budget of 300 µA. A robust and power efficient radio protocol is therefore crucial in fulfilling these requirements.[3]

1.1.2 Base Station

The base station has, in contrast to the sensor nodes, much more resources at its disposal: A full-fledged embedded Linux, a 100 Ah lead-based battery and 2×90 W solar panels, WiFi and GPRS link relax the needs for power efficient wireless protocol mechanisms on the base station side.

The communication board is yet the same as on the sensor nodes, which reduces the system complexity on the RF side as the specifications are identical.

1.2 The Communication Board

The communication board has seen several overhauls and designs circulating since its original conception. An overview over some of the different RF platforms (*motes*) is given in table 1.1. *DPP2 LoRa* is the latest development at TIK (Institut für Technische

Informatik und Kommunikationsnetze, Computer Engineering and Networks Laboratory) and the main topic of this thesis. When writing in this thesis about *DPP2 Lora* (SX1262 radio) and its predecessor *DPP2* (CC430 radio) the author is relating to the communication board part rather than the whole dual-platform system including the application MCU.

1.2.1 MCU

Most of the different power aspects have been subject to technical progress due to the shrinking of the semiconductor process sizes and agressive clock & power gating of the systems' subcomponentes, resulting in finer configuration options regarding low power or sleep modes. Concerning the resources and feature set of new-generation microcontrollers, 32-bit architectures, built-in full-fledged DSP instructions, single-precision floating point units, 2D-DMA for e-ink displays, built in analog operational amplifiers, USB, and many more timers, low-power peripheral extensions (e.g. low-power UART among others) are state of the art, and not neccessarily anymore in conflict with low-power requirements. Future products might even include neural network accelerators for smart IoT and medical appliances (see *Poseidon*, a new RISC-V ASIC by IIS, ETH Zurich [5]).

As the speed of microcontrollers (MCU) has been increasing, semiconductor processes became smaller, yet low-power applications do not require much more bulk processing power and therefore are idle most of their time. Leakage current has therefore become more and more important in low-power system design. Techniques such as power gating but also physical ones (PVT), including decreasing of the supply voltage.

Current Limitations

DPP2's communication firmware has to be fitted into 8 KiB of SRAM and 32 KiB Flash memory. The consequence is that firmware stack and protocol have to adjust to the MCU's intrinsics, resulting in more complex, platform dependent code. Extensions for logging, debugging, interfacing, and operating system facilities have to be kept low.

The minimal spacing between the end of a reception and start of a transmission is partly defined by the time needed for the MCU to process the state transition. This might include the data-transfer from and to the Radio as well. With the current implementation of Glossy, 2 ms process time (or gap time) have been shown to work.

Another problem is regarding the data transfer between application processor, communication processor, and radio PHY: While having a time critical task running on the

Platform		**TinyNode584 [7]**	TinyNode184	MSP430-CCRF [8]	**DPP2 LoRa**
Year		2005	2005-2008	2011	2018
Model		TinyNode584	TinyNode184	OLIMEX MSP430-CCRF	Custom built
Nom. Voltage		3.0 V	3.0 V	3.0 V	3.0 V
Min. Voltage		2.4 V	2.1 V	1.8 V	1.8 V
CPU	Model	TI MSP430F1611	TI MSP430F2417	TI CC430F5137IRGZ (MSP430 CPUXV2) [9]	ST STM32L433CC [10]
	Architecture	16-bit MSP430	16-bit MSP430	16-bit MSP430	32-bit ARM Cortex-M4
	Clock	16 MHz	16 MHz	20 MHz	80 MHz
	Drystone 2.1	0.288 DMIPS/MHz [11]	-	-	1.25 DMIPS/MHz
	RAM	10 KiB SRAM	8 KiB SRAM	4 KiB SRAM	64 KiB SRAM
	Flash	48 KiB Flash	92 KiB Flash	32 KiB Flash	256 KiB Flash
	Min. power	0.2 µA (LPM4)	0.1 µA (LPM4)	1.0 µA (LPM4)	8 nA (Shutdown without RTC) or 0.28 µA (Standby with RTC) [12]
	Wakeup time	6 µs (from low-power)	1 µs (from low-power)	6 µs (from low-power)	7 µs (from Stop mode)
	Specialties	-	-	Radio SoC	RAM2 (16 KiB) has option for retention
Radio	Model	Semtech XE1205	Semtech SX1211	TI CC430F5137IRGZ (CC1101) [9]	Semtech SX1262 [13]
	Nom. Frequency	868 MHz	868 MHz	868 MHz	868 MHz
	Min. sleep current	4.1 µA	0.1 µA	-	160 nA
	Rx current	16 mA	3 mA	16 mA	4.5 mA (5.5 mA in *boost* Rx)
	Max. sensitivity (1 % PER)	−121 dB at 1.2 kBaud	−107 dB at 25 kBaud	−111 dB at 1.2 kBaud	−148 dB at 293 kb/s (SF12, 125 kHz)
	Max. Tx power	12 dBm	12.5 dBm	12 dBm	22 dBm
	Max. power @ max. Tx power	186 mW	75 mW @ 10 dBm	108 mW	389 mW
	Modems	2-FSK	FSK, OOK	2-FSK, 2-GFSK, MSK, OOK, ASK	LoRa, GFSK
	Specialties	On-board chip antenna	On-board chip antenna, Packet Engine	Radio SoC, Packet Engine	Packet Engine
Specialties	-	External 512 KiB Flash	External 512 KiB Flash	-	Separate BOLT Processor (TI MSP430FR5969) with 50 nA sleep current

Table 1.1: Different radio platforms subject to reasearch at TIK over the years. Values extracted from supplied datasheets. Note the shift towards smaller minimum suply voltages, lesser power consumption, higher speeds and Tx power levels: They might be the cause for the long-range IoT boom we have seen the last years.

communication processor, the application processor might have to write sensor data to the communication processor. As processing and memory bus resources are sparse, means to mitigate the problem were developed, resulting in an additional, dedicated and timing safe FIFO Buffer between both processors called *BOLT* [6]. Its main advantage is the built in persistent FRAM which maintains even under power off a valid and consistent FIFO state. State of the art MCUs have support for autonomous DMA in both ways and fine grained IRQ priorization. A DMA-based ring-buffer might therefore pose an alternative to *BOLT*.

1.2.2 Radio

At TIK, several concepts of single-radio and multi-radio topologies have been evaluated. Some specialities include highly regenerative wakeup radios or custom OOK based low-power listening techniques.

Yet the default method of wireless data communication consists of an 868 MHz radio with 2-FSK (or 2-GFSK) modulation at 12 dBm and 250 kb/s. The integrated package engine for managing of preamble, syncword, addressing, payload management and CRC checksum is used by default, allowing for accelerated interfacing as well as reliable and error free reception handling (meaning only valid data is being processed).

Current Limitations

The current platform's radio configuration is staticly configured, meaning the wireless network's bandwidth, bitrate and power have been predefined. This lead to problems in the field regarding wireless channels and their corresponding path loss being near the radio's sensitivity level. Thus, a packet error rate (PER) over 40 % has been no rarity, degrading overall network performance.

The CC430's maximum Tx power level (12.5 dBm) regarding long-range IoT applicances is quite low. Line-of-sight distances above five to ten kilometers are common in the *PermaSense* regions. For reliable communication, especially for orchestration purposes from the base station, the maximum power should keep up with modern *LoRa*-based nodes, with transmission powers up to 22 dBm.

The current Glossy firmware implementation includes hand-tuned assembly code for precise radio timings and time exchange. Refactoring the existing firmware code therefore requires to precisely analyze the impact on the defined timings.

1.3 FlockLab

"FlockLab is a wireless sensor network (WSN) testbed, developed and run by the Computer Engineering and Networks Laboratory at the Swiss Federal Institute of Technology Zurich in Switzerland." [14]

As the evaluation and testing of wireless systems is complex and very time consuming due to the distributed nature of WSN's (i.e. longer-distances between the nodes), automation of deployment, measurement and analysis of the distributed nodes is a must.

FlockLab allows to attach different types of communication boards (called targets or *nodes*) on over 25 scattered *observers*. The observers can monitor the communication boards's power consumption, GPIO activity, can actuate the GPIOs on their own, and might expose a serial interface via TCP/IP for remote scripting.

By registering *tests* on the FlockLab webpage, one can schedule a time slot where the given nodes (one per observer) are setup exclusively with his firmware and test configuration. After the user's time slot, the results are from the different observers are bundled together, and can then be analyzed offline by the user.

The complex nature of the ETH Zurich's ETZ building in which FlockLab is located for the most part, guarantees a high variation of different RF channels between nodes.

1.4 Motivation

To keep pace regarding the state of art in the Sub-Gigahertz area and enhancing *PermaSense*'s perfomance for the latest geophone equipment, TEC (Computer Engineering Group, a subsidiary of TIK) concluded that the development of a new communication board and major adaptions at the MAC layer, including new concepts for a reliable, scalable, long-range and low-power WSN are required. With the introduction of a new next-gen LoRa chip by Semtech in March 2018, called SX1262, featuring a higher maximum Tx power level, lower Rx power consumption, a new faster spreading factor (SF5) and a less complex interface, this master thesis can build uppon the latest cutting-edge technology available.

Utilizing a new radio chip also meant to analyze and understand the inner workings and interactions of the new components precisely, entailing a thorough analytical system and network model of timings, sensitivities and power consumption.

Chapter 2

Related Work

This chapter gives insight into similar reasearch topics in academia.

2.1 The MAC Protocols

In a extreme environment, such as the *PermaSense* regions, links between nodes and base are not guaranteed to withstand changing meteorological and topological conditions. Snow might fall on some of the nodes, increasing path loss. Nodes can – and might be even designed to – move due to avalanches and geomorphological events.

To keep connectivity between sensor node and base station, a chain of node-to-node links, relaying the original message up to the base station, can be used. This concept is called multi-hopping. Due to the nature of wireless networks having many redundant node-to-node links, multi-hopping can increase resilience against environmental changes and facilitate mobility requirements (e.g. nodes which are positioned on vehicles or people) by extending the networks range.

Furthermore, long periods of disconnection, the high energy costs involving frequent synchronization or discovery beacons, high system complexity concerns (e.g. handshaking schemes, distributed state, channel assignment), call for a custom protocol solution.

PermaSense's multi-hopping based protocols are currently based on two quite different approaches called *LWB/Glossy* and *Dozer*.

2.1.1 LWB/Glossy

"LWB (for *low-power wireless bus*) lets nodes communicate as if they were connected to a shared bus, where all nodes can receive all packets, although the underlying multi-hop wireless topology may be very complex and continuously changing". [15]

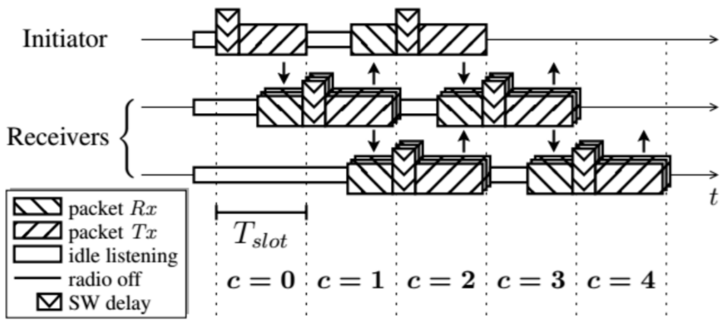


Figure 2.1: Example of a Glossy flood. By precisely timing the retransmissions at the receivers, Glossy might benefit of constructive interferences for higher reception ranges. [16]

LWB's key aspect is the scheduling and assignment of communication resources on the base station, without knowledge about the underlying network's topology. A periodic synchronization beacons announces *rounds* to the nodes. The rounds contain slots, which have different purposes defined by the round's schedule. These include non-contended up- and down-stream, one-to-one or one-to-many transmissions for a certain node. *Contention* based LWB slots are intended for the registration of data streams, which allow a node to signal the base station that it has data available.

eLWB (for *event-based* LWB) introduced an additional mechanism for more real-time notifications, with lesser overhead.

Glossy

The underlying low-level mechanism LWB interfaces with is called *Glossy.*[16] Glossy uses a message flooding mechanism. A initiating node (incl. the base station) will trigger the transmission of a message, and all other nodes will be listening. Every node that receives the message will retransmit it simultaneously with a fixed offet (the slot time) in the next slot. The nodes continue retransmitting within every second slot since their first transmission, until there are no retransmissions left. After a given slot count the flood is considered finished.

Adding timestamp and hop-count information to the message, Glossy can even serve as a base for sub-microsecond time-synchronization [17]. Floods can be initiated by multiple nodes in contention, relying on the capture effect for a message to reach its destination. The statistics on successful Glossy floods are highly-independent [15]. The topology agnostic protocol design has many other benefits as predictable timing characteristics (i.e. an upper bound on latency).

LWB/Gloria vs. LWB/Glossy

This thesis introduces a new variation of LWB/Glossy that tries to handle different communication ranges incorporating the flexibility of the new SX1262 and its different modes.

While LWB/Glossy only knows one datarate, and has therefore only one specified slot layout, LWB/Gloria has to cope with very different slot times. The key concepts of LWB extended with topology-agnostic link management lead to a more complex protocol.

Several other extensions that cope with the overall power consumption are also being introduced.

2.1.2 Dozer

Dozer is a heavily distributed message-hopping and listener initiated transmission protocol. Each node manages its own schedule regarding a chosen parent-node and several child nodes [18].

A node will send out a beacon periodically asking for other nodes (child nodes) to indicate their intent for sending data. The contention mechanism uses exponential backoff to establish then a local schedule for the node-assigned data slots. Messages can propagate this way through the whole network, eventually reaching their designated target.

Using different bands, Dozer becomes more resilient towards other nodes or noise sources. As messages are only transmitted where neccessary (in contrast to Glossy), Dozer has more potential in making the race for ultra-low-power WSNs.

The downside is the requirement for temporarly storing the messages on the intermediate nodes, making additional built-on persistent memory neccessary. Real-time transmissions and precise time-synchronization schemes are more hard to realize due to the chaotic nature of the hopping-chain and accumulated clock drift by the longer intermediate gap times.

During this master thesis' reasearch, another thesis implementing Dozer on top of the *DPP2 LoRa* platform and author's driver stack has been published [19].

2.2 Concurrent *LoRa* Transmissions (CT)

LoRa has already been under research regarding concurrent transmissions in a paper by Chun-Hao Liao et al. of University of Tokyo [20].

Key findings are that due to the longer symbol times, *LoRa* is not very suceptible to *beating*, “a fading-like effect caused by CFO” (carrier frequency offset). The paper assumes CFO to be in the range of several kHz. The *Capture effect* prevails and is the major aspect for working CT. A variation of CT called *offset-CT* is introduced, which backoffs transmissions by a random amount of *LoRa* symbols in each slot (CT is very similar to Glossy), allowing for a better power margin and separation between captured and wanted sub-carrier vs. interfering sub-carriers.

The concept of constructive interference (as promised by Glossy) is not considered, i.e. non-existent in the paper.

Chapter 3

Theory

3.1 LoRa

Introduced by Cycleo, and acquired by Semtech in 2012, the *LoRa* (for ***Lo**ng **Ra**nge*) PHY and modulation technology became a big player in the IoT business (beside *Sigfox*, *Zigbee*, or other standard *IEEE 802.15.4* solutions among others).

The modulation is based on *chirp spread spectrum* (CSS) where digital symbols are being modulated onto periodically reoccuring, linear *chirp*s (short for **C**ompressed **H**igh **I**ntensity **R**adar **P**ulse, *radar* short for **Ra**dio **D**etection **A**nd **R**anging).

A typical LoRa paket can be seen in fig. 3.1. To further understand the principles and timing characteristics of *LoRa*, let us introduce the most basic variables:

- **chirp rate** R_{chirp}: The speed of increase in frequency of the chirps in Hz/s. Can be interpreted as the slope of the chirps.
- **chip rate** R_{chip}: The periodicity at which the chips, the smallest information unit gets processed in the decoder or encoder unit in Hz. It equals the set bandwidth by design (see [13, p. 37]), and should not be confused with the *chirp rate*.
- **bandwidth** B: The frequency range at which a chirp starts and ends in Hz. Configurable options include 125 kHz, 250 kHz and 500 kHz.
- **symbol rate** R_S: The periodicity at which the spread information, the LoRa symbols, get sent in Hz.
- **spreading factor** SF: "The ratio between the nominal symbol rate and chip rate is the spreading factor and it represents the number of symbols sent per bit of information" [13, p. 37]. There are currently eight different spreading factors defined, namely from slowest to fastest datarate: *SF12*, *SF11*, . . . , *SF5*.
- **coderate** CR: The ratio between added bits for forward error correction and actual count of bits. Configurable values include $\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$ and $\frac{4}{4}$. The payload bits are grouped in pairs of four.

The symbol rate gets callculated according to following formula [21]:

$$R_S = \frac{B}{2^{SF}} \tag{3.1}$$

and the chirp rate is defined as

$$\begin{aligned} R_{chirp} &= B \cdot \frac{R_{chip}}{2^{SF}} \tag{3.2} \\ &= \frac{B^2}{2^{SF}} \tag{3.3} \end{aligned}$$

Configurations which differ in their chirp rate are called *orthogonal*, meaning their signals do not interfere and may coexist. This is quite useful to reduce the chance of collisions and/or to increase the channnel efficiency. Signals that will collide are e.g. (SF7, 125 kHz), (SF9, 250 kHz), (SF11, 500 kHz) as their chirp rate is identical.

The bit rate R_b is defined as follows:

$$R_b = R_S \cdot \mathrm{SF} \tag{3.4}$$

Be advised that R_b is not concerned with the coderate. For this one we have to take the Time-on-Air calculation into account:

$$ToA = \frac{2^{SF}}{B} \cdot N_{symbol} \tag{3.5}$$

where N_{symbol} represents the number of symbols in a packet. This is for SF5 & SF6:

$$\begin{aligned} N_{symbol_{(SF5,SF6)}} &= N_{symbol_{preamble}} + 6.25 + 8 \\ &+ \left\lceil \frac{\max\left(8 \cdot N_{byte_{payload}} + N_{bit_{CRC}} - 4 \cdot SF + N_{symbol_{header}}, 0\right)}{4 \cdot SF} \right\rceil \cdot (4 \cdot CR + 4) \end{aligned} \tag{3.6}$$

$N_{bit_{CRC}}$ represents the number of CRC bits used, namely 0 or 16. $N_{byte_{payload}}$ is the actual transmitted payload in bytes. $N_{symbol_{header}}$ represents the number of header bits in *explicit header mode*, which allows for the receiver not to have knowledge about configured coderate, payload size and CRC setup on the transmitter side. Of course spreading factor, bandwidth and syncword still have to be identical.

For all other spreading factors:

$$\begin{aligned} N_{symbol} &= N_{symbol_{preamble}} + 4.25 + 8 \\ &+ \left\lceil \frac{\max\left(8 \cdot N_{byte_{payload}} + N_{bit_{CRC}} - 4 \cdot SF + 8 + N_{symbol_{header}}, 0\right)}{4 \cdot SF} \right\rceil \cdot (4 \cdot CR + 4) \end{aligned} \tag{3.7}$$

The syncword (4.25) is two symbols shorter compared to SF5 and SF6.

For slow communication, additional synchronization bits are inserted interleaving the symbols, so the receiver does not loose track, i.e. keeps locked on the carrier. Semtech calls this feature *Low Data Rate Optimization* (LDRO). The Time-on-Air formula changes accordingly:

$$\begin{aligned} N_{symbol_{LDRO}} &= N_{symbol_{preamble}} + 4.25 + 8 \\ &+ \left\lceil \frac{\max\left(8 \cdot N_{byte_{payload}} + N_{bit_{CRC}} - 4 \cdot SF + 8 + N_{symbol_{header}}, 0\right)}{4 \cdot (SF - 2)} \right\rceil \cdot (4 \cdot CR + 4) \end{aligned} \tag{3.8}$$

The mentioned formulas are all implemented in the author's toolbox called *flora-tools*, or in Semtech's *LoRa Calculator* [22]. Simple LoRa decoder and encoder extensions are available for GnuRadio. A reverse engineering approach regarding the symbols' encoding can be found in [23].

A further helpful fact is, that the spreading factor number represents the number of raw bits (including FEC bits) contained within a symbol for disabled LDRO.

3.1.1 Technical Advantages

Shannon-Hartley Theorem

Semtech is arguing in [24] that long-range wireless communication and its inherent high path-loss might result in

$$SNR \ll 0\,\mathrm{dB} \tag{3.9}$$

Consider the Shannon-Hartley theorem. It gives a analytical theoretical upper limit on the channel capacity C (i.e. bitrate) for a given bandwidth B and SNR.

$$C = B \cdot \log_2(1 + \frac{S}{N}) \tag{3.10}$$

$\frac{S}{N}$ here is the non-logarithmic ratio between the signal's and noise's RMS powers. Considering $\frac{S}{N} \ll 0$ and applying taylor-series approximation, one can conclude that

$$C \approx B \cdot 1.433 \cdot \frac{S}{N} \tag{3.11}$$

This means that by increasing the bandwidth or power linearly, the channel capacity approximately grows linearly as well. Semtech argues therefore that wideband modulations deliver more link budget for the power consumption (respectively can reach further distances for the same power provided) than narrow-band modulations, such as FSK, do [24, p. 11]

table E.0 gives an overview of the different mentioned datasheet values and the maximum achievable link distance under the assumption of a cubic free-space path loss and no antenna losses or gains, as well as the required energies for transmission for the SX1262 radio.

Free-space path loss (FSPL) assumes the the power of a transmitter being evenly distributed on a surrounding sphere. With increasing distance to the transmitter node, the sphere's surface grows by the square, therefore reducing the perceived power by the square at this distance. The FSPL is given as

$$\text{FSPL} = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{3.12}$$

Introducing a cubic rather than square decay, the author hopes to give a better approximation of the actual perceived path losses due to multi-path propagation, fading, antenna losses, or smooth obstacles such as trees. For a better, more precise approach ITU is recommending using topological data and raytracing for complex, high-altitude environments such as the alps [25].

The resulting formula for the maximum distance (for 1 % PER) is therefore

$$\text{Link Budget} = -(\text{Sensitivity} - \text{Tx Power} + \text{RF-Switch Insertion Loss}) \tag{3.13}$$

$$\text{Max. Distance} = \frac{10^{\frac{\text{Link Budget}}{10 \cdot 3}}}{4\pi} \cdot \text{Wavelength} \tag{3.14}$$

where Tx Power = 22 dBm, RF-Switch Insertion Loss = 1.5 dB and Wavelength = $\frac{868\,\text{MHz}}{300 \times 10^6\,\text{m/s}}$

Further Arguments

LoRa does have a very high process gain allowing for *SNR*'s as low as −20 dB for SF12.

Additional advantages are listed in [24, p. 11], including:

- Bandwidth scalability: Configuration allows for narrowband and wideband setups.
- Constant Envelope modulation scheme: Tx power does not vary (similar to FSK) allowing for a more efficient PA.
- $BT > 1$: Resilience/Immunity regarding in-band and out-band interferences.
- Multipath/Fading resistance due to wideband chirp.

- Doppler resistance
- Long-Range capability
- Symbol orthogonality

3.1.2 Technical Disadvantages

LoRa's speed is currently limited to 62.5 kb/s in the SX1262, in contrast to GFSK being configurable up to 300 kb/s.

On [24, p. 12], figure 3, an increased drop in *LoRa*'s sensitivity can be seen at 11 kb/s. Additional listings of higher-speed spreading factors (SF5, SF6), their sensitivity and regarding the longer recommended preamble indicate this as well. The conclusion is that *LoRa* is not fitted yet for speeds beyond 62.5 kb/s, making the use of faster modulations such as GFSK a requirement.

Even though Semtech argues about the channel efficiency being higher compared to FSK due to the orthogonal symbols (see "Network Planning Example" [24, p. 22]), they neglect the fact about the asymmetric distribution of the bitrates and sensitivities, whereas the FSK channel plan is symmetric.

An inherent misconception of *LoRa* is its need for the number of symbols required to transmit a given payload size, having to be a rational divisor of the actual required bit count (see the ceil operation in eq. (3.7)). E.g. this means for SF12, 125 kHz (coderate $\frac{1}{4}$, no explicit header, no CRC, no LDRO) that the same number of symbols is transmitted, independent of the payload being defined as 1 or 6 bytes, resulting in 5 unused, transmitted bytes. As *LoRa* is configured in explicit header mode for this thesis, the ceiling overhead gets small compared to the header overhead.

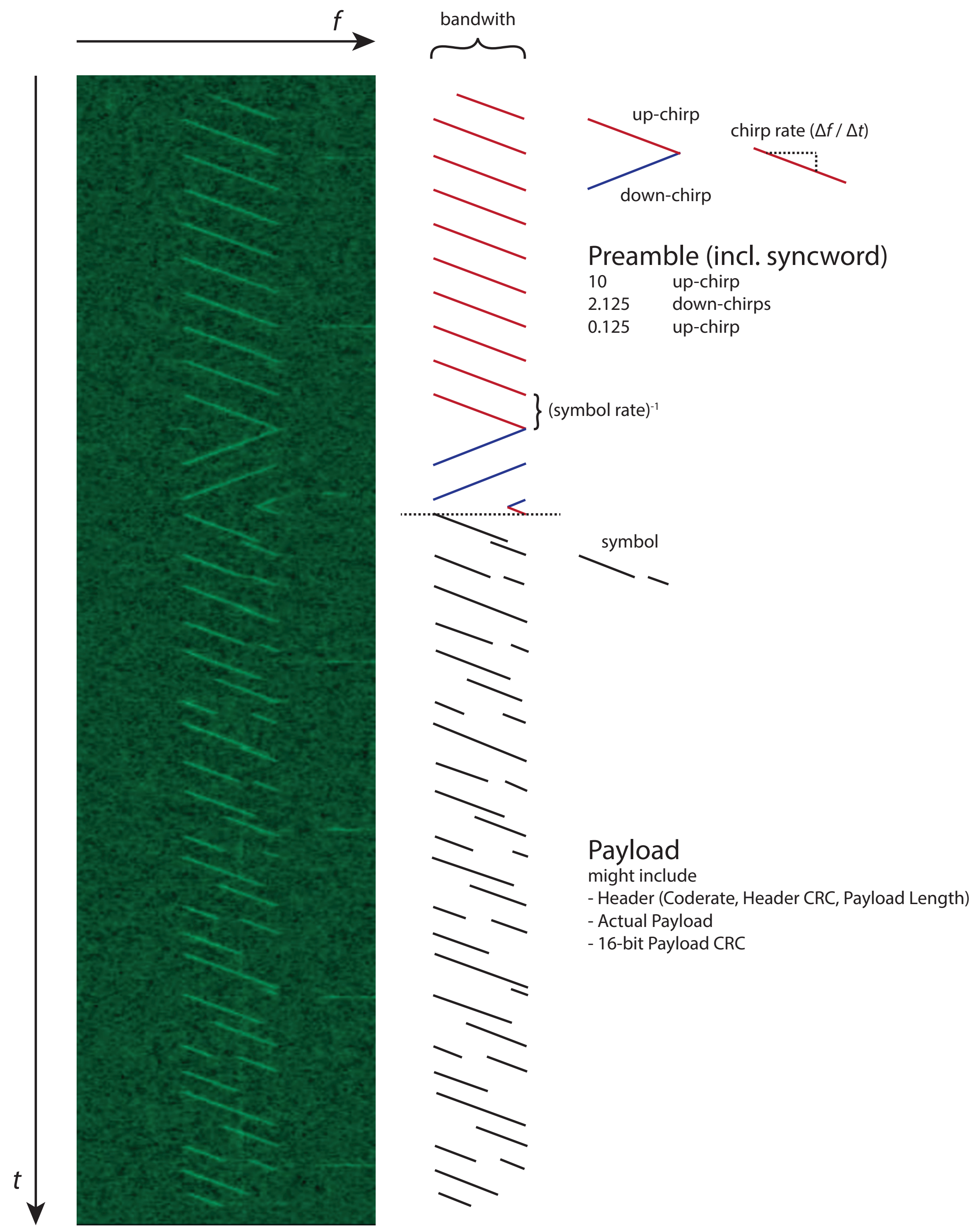


Figure 3.1: Physical signal structure of a captured LoRa packet as seen on a waterfall spectrum.

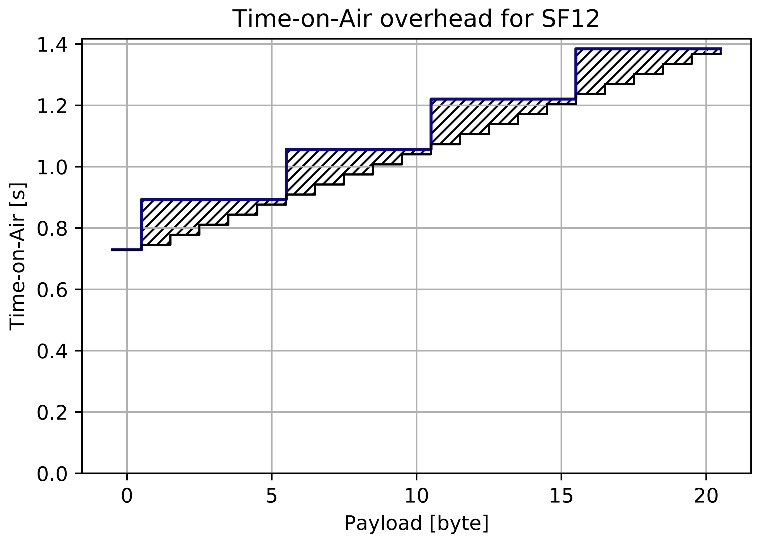


Figure 3.2: Time-on-Air overhead for *LoRa* SF12, 125 kHz. The hatched area represents the wasted ToA ceiling overhead compared to the non-ceiled ToA given a set payload in bytes.

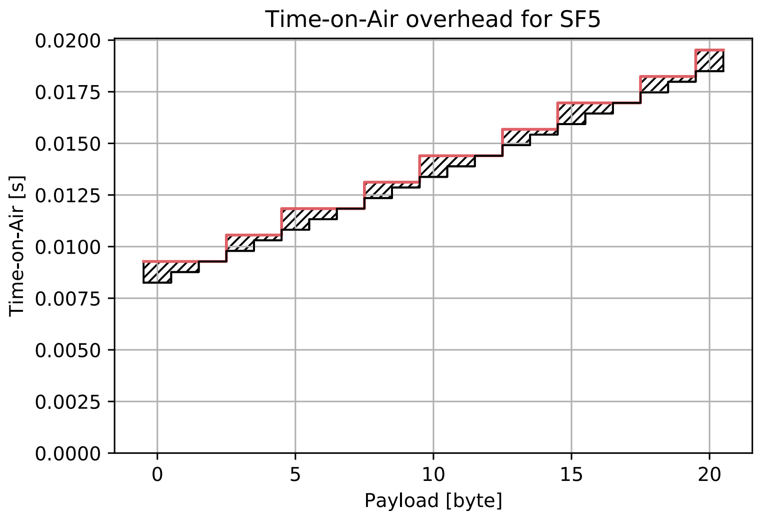


Figure 3.3: Time-on-Air overhead for *LoRa* SF5, 125 kHz. The hatched area represents the wasted ToA ceiling overhead compared to the non-ceiled ToA given a set payload in bytes. In comparison to SF12, the overhead does have less of an impact due to the finer steps.

This behaviour is additionaly suboptimal due to the fact, that all spreading factors, and coderates behave differently.

3.1.3 LoRaWAN

LoRaWAN has in particular its fair share in the popularity of *LoRa*. It defines a MAC layer protocol for star topology networks with one base station in the center and Aloha-like contention for the nodes. *LoRaWAN* allows for different parties renting the service for secure telegram exchange to a cloud-based backend, e.g. enabling large scale and far-away distributed IoT appliances, such as water supply or air quality monitoring services.

It has been introduced as a service by Swisscom in 2015 in Switzerland. As of 2018, free access to *LoRaWAN* maintained by tech enthusiasts and the *The Things Network* initiative is gaining popularity, making hobbyist projects in IoT more affordable.

As the node count increases in comparision to the available base stations, the chance of interferences and collisions grows, the networks performance begins to degrade. A paper about LoRaWAN's ability to scale, came to quite pessimistic conlussions in this regard [26], not being unanswered by the industry [27].

A typical configuration which is frequently used in *LoRaWAN* is SF7, 125 kHz.

Chapter 4

Low-Level Hardware & Software Implementation

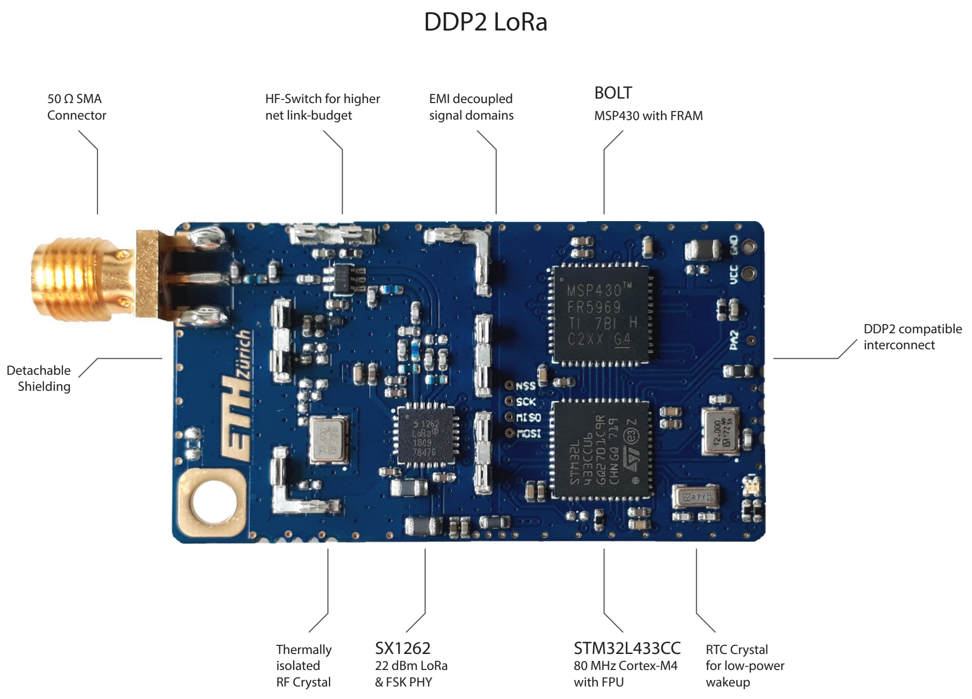

DDP2 LoRa
50 Ω SMA Connector
HF-Switch for higher net link-budget
EMI decoupled signal domains
BOLT
MSP430 with FRAM
DDP2 compatible interconnect
Detachable Shielding
Thermally isolated RF Crystal
SX1262
22 dBm LoRa & FSK PHY
STM32L433CC
80 MHz Cortex-M4 with FPU
RTC Crystal for low-power wakeup

Figure 4.1: The newly designed *DPP2 LoRa* communication board designed by R. Da Forno and M. Wegmann.

This chapter will give insight into the ideas, concepts and development of the latest custom built radio platform at TIK, called *DPP2 LoRa*.

4.1 Interface Specification

The new *DPP2 LoRa* was designed to match the same interface specifications and form factor as its predecessor. Board-to-board connector, 3.0 V power supply, BOLT interface, as well as debug and programming connections are identical. Adapters for the use in FlockLab and development carrier board had therefore not to be redesigned.

4.2 Component Selection

Major components, including radio chip and MCU series, had largely been preselected in advance for this thesis. Yet, minor decisions regarding the replacement of the MCU by a even less power-consuming, pin-compatible model from the same vendor featuring cryptographic accelerators, has been taken by the author.

The overall BOM costs sum up to approximately 10 USD for small quantities. All components are specified to withstand a temperature range of −40 °C to 80 °C.

4.2.1 Radio

With the announcement of the SX1262 by Semtech late 2017 featuring advances in Rx power consumption at the same sensitivity levels (50 % less!), higher LoRa speeds (introduction of SF5), and an increase in maximum transmission power (58 % more) compared to the SX1276 radio, the inclusion of this new radio would render a must. Competing System on Chip (SoC) solutions with similar RF specifications have not been existent as of this report. In the future, a SoC solution with integrated power managment and radio might become the best choice.

Semtech recommends to use an external RF path switch for transmission and reception to the antenna for best performance in the 868 MHz band [28, 29], as link budget will be higher due to the better sentivities even tough the insertion path loss of −3 dB. It is controlled directly by integrated logic in the SX1262 via `DIO2`. For even better performance, the RF switch's power supply can be turned on and off via a MCU pin and PMOS-FET transistor circuit and is being filtered in contrast to Semtech's reference design.

The radio interfaces as a SPI (Serial Peripheral Interface) slave with the master MCU at 12 Mb/s, using the four lines `SCLK` (serial clock), `MOSI` (master-out-slave-in data), `MISO` (master-in-slave-out data) and `NSS` (active-low chip select signal). Additional lines from radio to MCU such as `BUSY` for operation and state change monitoring and `DIO1` for IRQ

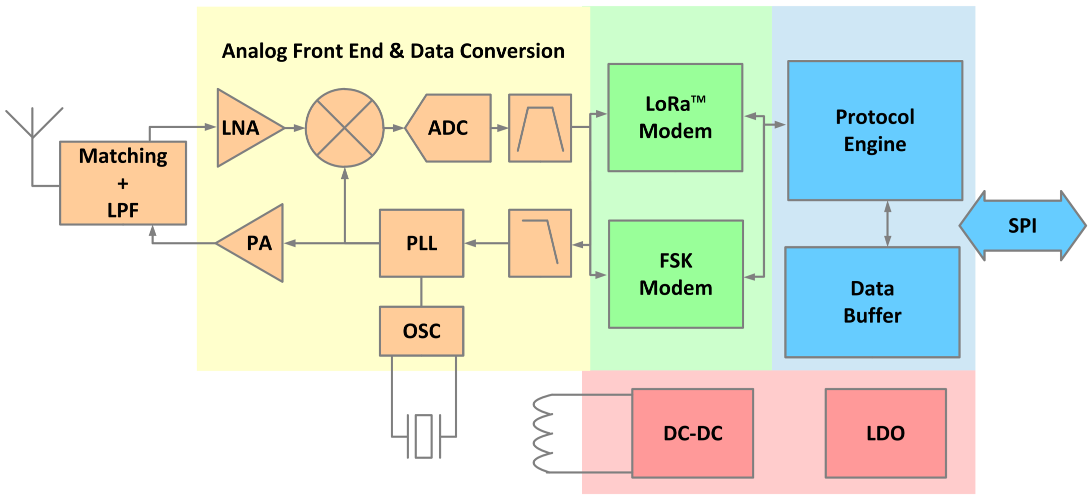


Figure 4.2: SX1262's architecture as presented by Semtech [13, p. 1]

indication are required.

Due to the deacreasing semiconductor process sizes, the unbalanced transmission output path has a low characteristic impedance of roughly 12 Ω. A low-cost impedance transformation circuit, notch filter (first harmonic) and low-pass filter (higher-order products) guarantee a high suppression of unwanted intermodulation products to respect FCC and CEPT limits.

For better sensitivity performance two pins serve as a balanced Rx input (as this supresses 2nd order products quite efficiently in differential LNA topologies), requiring a balun circuit.

All passive components are of high quality supplied by vendors such as Murata. As can be shown with the transmission coefficient regarding an impedance mismatch

$$T = \frac{2Z_L}{Z_L + Z_S} \tag{4.1}$$

with Z_L being the load impedance (antenna or antenna switch) and Z_S being the supply impedance (the impedance at the radio pins): Even a $\alpha = 10\,\%$ impedance mismatch will only result in a $\approx 8.9\,\%$ or $-0.4\,\text{dB}$ drop in transmission power:

$$\text{Power Ratio} = \frac{T_\alpha^2}{1+\alpha} \tag{4.2}$$

$$= \frac{\left(\frac{(2+2\cdot\alpha)}{(1+\alpha)+1}\right)^2}{1+\alpha} \tag{4.3}$$

$$= \frac{\left(\frac{2\cdot(1+\alpha)}{2+\alpha}\right)^2}{1+\alpha} \tag{4.4}$$

This is in expected range of production tolerances. Antenna build-quality, coaxial-line and parasitic connector capacitance are considered to be more of a concern. [1].

The SX1262 can make use of a 32 MHz standalone crystal (XTAL) or a temperature-controlled crystal oscillator (TCXO) powered by the `DIO3` pin. While latter will have a smaller maximum drift of ±2ppm in the temperature range of −40 °C to 80 °C compared to the AT-cut XTAL with ±25ppm and therefore have better perfomance for long transmissions (> 0.4 s) the former will not consume as much current (1.5 mA) and is less expensive. *DDP2 LoRa* has therefore footprints for both solutions, with the XTAL mounted by default.

Semtech did not document means to calibrate the XTAL via a clock output pin, making calibration in the future quite cumbersome.

4.2.2 MCU

Earlier communication board designs at TIK all have included a TI MSP MCU. During the last years, ARM-based low-power microcontrollers gained further popularity due to their performance and feature-rich ecosystem. STMicroelectronics (ST), as one the major MCU vendors, is offering different ARM architectures for ultra-low-power applications, namely Cortex-M0 and Cortex-M4 with versatile clock and power control. Other vendors such as TI, Cypress, Microchip/Atmel, NXP, Freescale, Dialog, Nordic, and new competitors like Espressif are competing as well.

The final decision for a 48-pin QFN solution by ST was made due to its comprehensive middleware and hardware configuration solution STM32CubeMX and STM32Cube HAL and the author's experience with ST's MCU product lines. STM32CubeMX allows for better porting of the firmware onto different ST MCU's and development kits such as the ST NUCLEO-L476RG board in connection with Semtech's SX1262 based development

[1] Information provided by a radio industry expert

kit.

STM32L433CC, STM32L443CC (STM32L433CC ABI compatible MCU with cryptographic accelerators), and STM32L462CE (160 KiB SRAM, 512 KiB Flash) are all compatible with *DPP2 LoRa*.
As *LoRaWAN* benefits from cryptographic extensions including AES, we decided for STM32L443CC. Due to import regulations in China which only allow certified cryptographic products to be imported, we had to settle for STM32L433CC, as TIK did not want to shift its board assembly to another supplier.

For precise timings and time keeping, a 12 MHz (±25ppm) high-speed AT-cut XTAL (actually a variant of the 32 MHz radio XTAL) is used for the MCU's system clock during operation. An additional 32.768 kHz (±25ppm) RTC XTAL is being supplied for keeping track of time during low-power modes and sleeping requiring an additional 250 nA of current consumption [12, p. 15]. An integrated temperature probe inside the STM32L4 could be used to control the HSE and LSE oscillator's tuning registers to limit drift down to (±4ppm) [2]. One might consider to measure the clocks' actual frequency on the auxilliary clock outputs for given temperatures and construct a lookup table.

4.3 Board Design

The communication board's design has been conducted in Altium Designer. Its dimensions are only slightly bigger by some millimeters compared to the *DPP2* communication board. The PCB uses 4-layers with the outer layers being signal layers and the inner two layers serving as power planes. To minimize the track widths on the RF path the overall thickness is only 1.2 mm, compared to the 1.6 mm used as a default board thickness for generic boards.

The radio's BOM is nearly identical to Semtech's recommendations regarding best performance [28].

4.3.1 Power & Shielding Concept

For the sake of compatibility with the existing facilities all components run on 3.0 V, supplied via internal GND and VCC power on the two inner layers of the *DPP2 LoRa* 4-layer PCB.

[2]Information by a radio industry expert

The radio's supply is additionally filtered via two EMI beads and signals separated via serial 1 kΩ resistors, to block interference while the radio is transmitting. High-frequency noise can follow parasitic paths back to the carrier board or sensor interface board can degrade the ADC's precision.

Clips for a detachable can shield have been placed to even allow for radio module certification required by the governmental authorities (in our case the BAKOM). The radio's VCC power plane was only extended under the shielding to further limit unwanted radio emissions.

Low-impedant bypass-capacitors near every chip's supply pins guarantee reliable operation for high Tx power and processing transients. It might yet be useful to decrease the amount of unused bypass-capacitance in the future to further limit parasitic leakage currents through the capacitors. Stitching vias were applied to further enhance fast-transient characteristics.

A big flaw in using a 3.0 V is the inherent inefficiency. MCU and the digital part of the radio are designed to be driven by 1.8 V, reducing dynamic power. The integrated DCDC circuit inside the radio for the power-amplifier stage is designed to be directly connected to a low-impedance lithium-ion or lithium-polymer battery (i.e. $\approx 3.7\,\text{V}$) [13, p. 36]. The consequence is that we waste valuable power through the first DCDC- and LDO-converter on the carrier board, and do not fit the optimal operating point. Luckily, the STM32L4 come with high-efficiency built-in voltage regulators (assuming pin output load to be high impedant), mitigating the problem a bit on the MCU side [12, p. 1].

4.3.2 RF Path

We foremost followed Semtech's recommendations and guidelines in placement and routing of the SX1262's subcircuit for best practices, reducing parasitic antennas and unwanted high-frequency emissions.

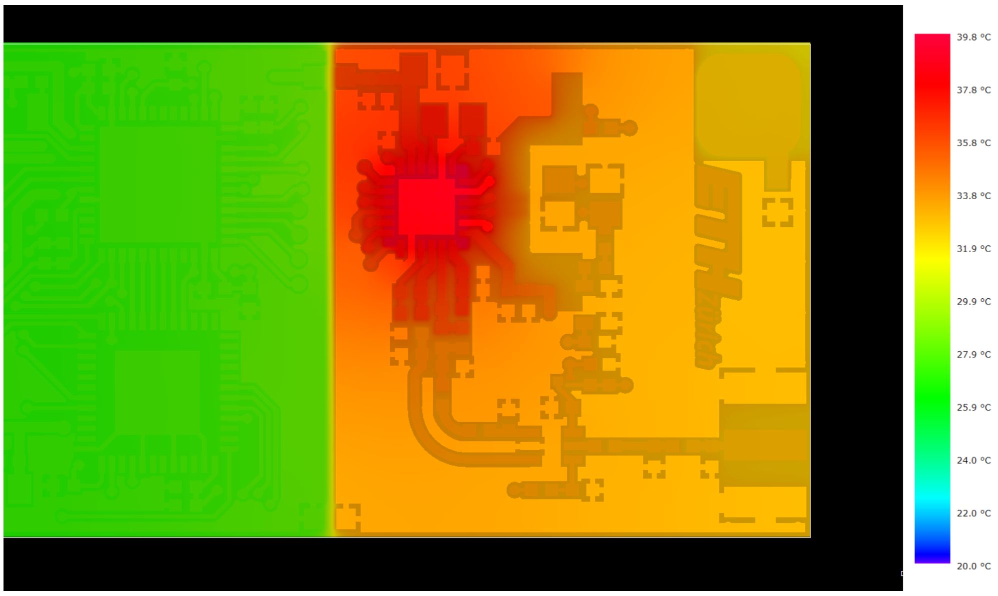
To fit in the complete RF path, it had to be slightly curved by 90 degrees as shown in fig. 4.1 over the radio chip. According to RF experts, this does not pose a relevant problem for 868 MHz designs. The RF tracks' thickness had to be adapted to match 50 Ω impedance for the inter-layer thickness of the top and first inner layer (the lesser the layer thickness, the lesser the track width).

The SMA connectors surrounding GND pour and central solder pad have been additionally reduced to limit the parasitic capacitance around the connector.

4.3.3 Thermal Isolation of Radio Crystal

As Semtech explains in [28, p. 12], due to the heating up of the SX1262 during long transmissions ($> 0.4\,\mathrm{s}$), the near radio XTAL can heat up as well, inducing unwanted drift. To mitigate this problem, Semtech recommends to place much copper mass and thermally conducting vias around the SX1262 for dissipate thermal energy away from the XTAL, and to place directly a copper keepout between SX1262 and XTAL to hinder thermal flow.

To check the thermal characteristics of the communication board, the author conducted a FEM simulation inside *PCB-Investigator* based on an actual, early revision of the communication board. The simulation was based on the actual material parameters, two-sided $10\,\mathrm{W/m^2}$ free heat convection, and $231\,\mathrm{mW}$ of static heat dissipation of the SX1262 (assuming operation at $22\,\mathrm{dBm}$). The results can be seen in fig. 4.2.



*

Figure 4.2a: Thermal simulation of early *DPP2 LoRa* design. Top Layer: Note the drop between the hot red square (SX1262's location) left, and the more yellow zone to the right with the four squares, where the radio's XTAL is placed. The large green zone to the left is induced by splitting the power and signal domains between MCU and radio.

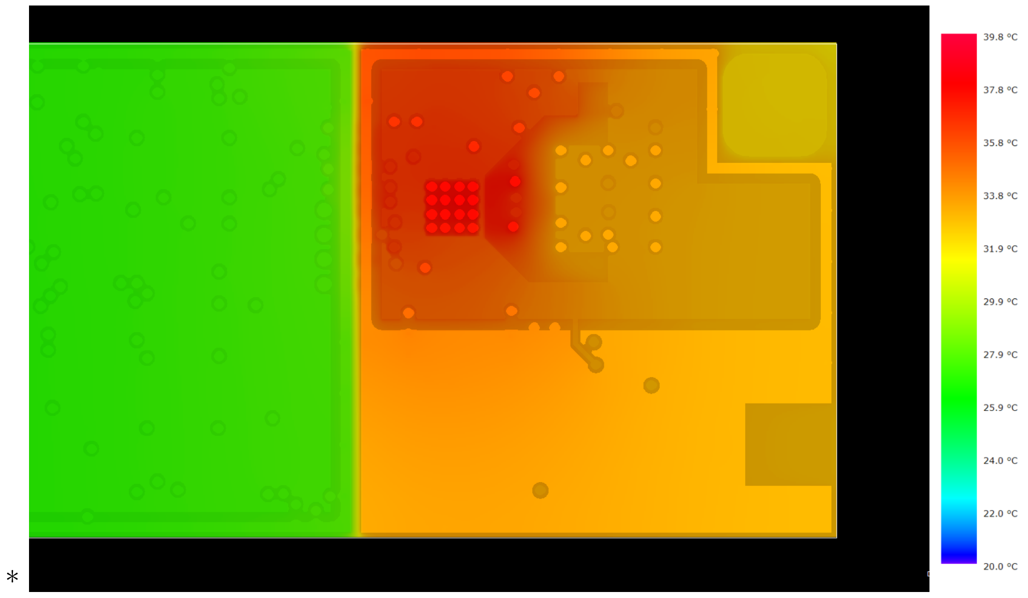


Figure 4.2b: Inner VCC layer.

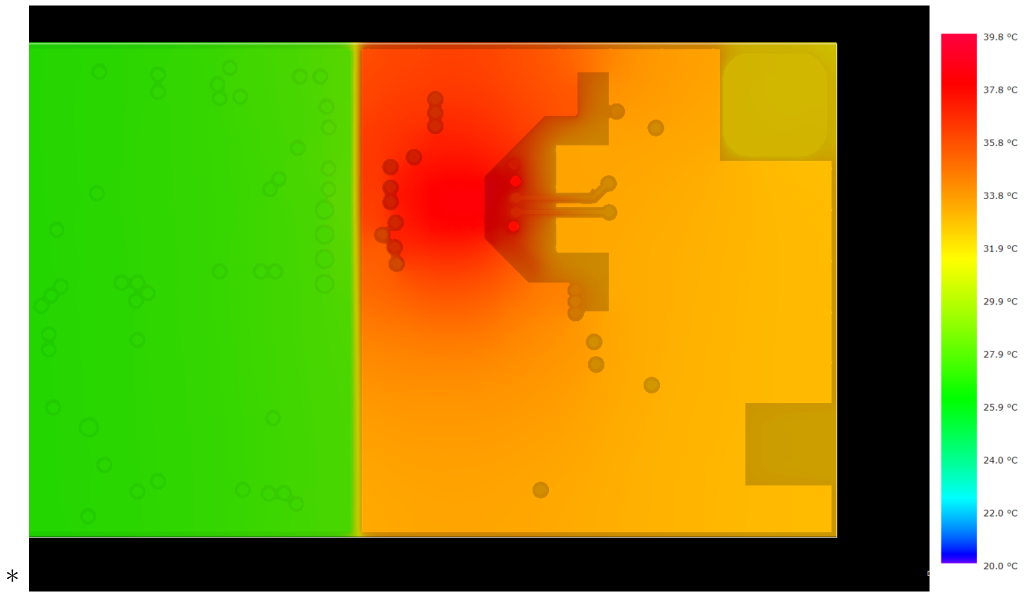


Figure 4.2c: Inner GND layer.

*

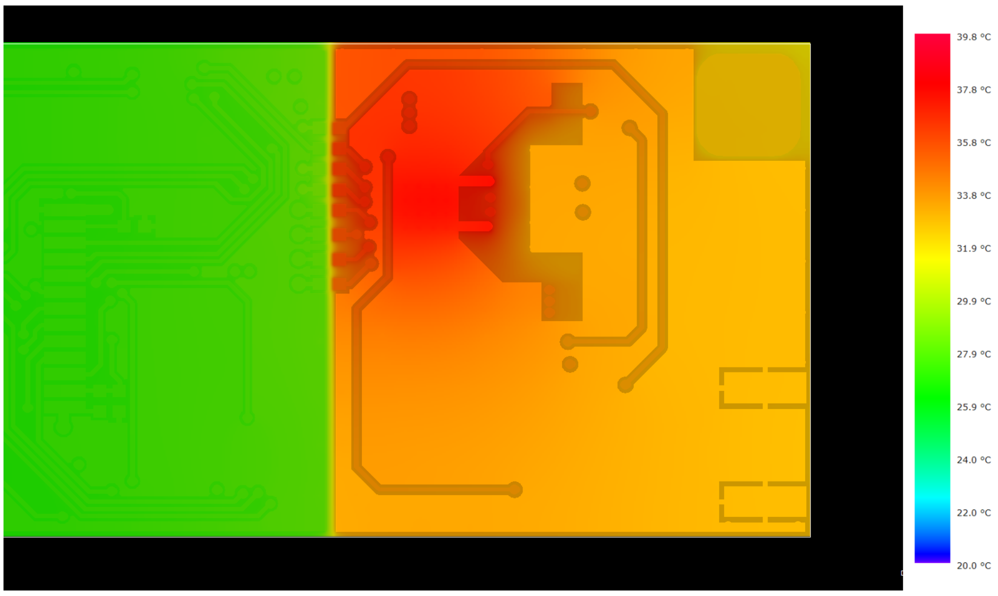


Figure 4.2d: Bottom Layer.

4.3.4 Tapeout

Three pre-series prototype boards were ordered by TIK from the Shanghai-based PCB vendor *PCBCart*. They have been assembled and tested in iterative manner by the author, regarding correct functional behaviour, RF performance and power. Minor fixes for the final tapeout had to be integrated, such as routing important indication signals (i.e. `BOLT-IND`, `RADIO-DIO1`) also to dedicated wakeup pins. The MCU's and radio's crystal footprint had been mirrored.

In the pre-series prototype, the board-to-board connector's gender had been mixed up. The author therefore had to design and produce a small intermediate adapter board by his own for faster turnaround time.

Once all crucial subsystems had been verified to be working, a production batch order for 90 assembled units had been given to *PCBCart*. As mentioned before, a last moment change order had to be taken for replacing the STM32L443CC with the STM32L433CC without cryptographic extensions for import to Shanghai. For the near feature, a shift in policy in import regulations is underway, relaxing the requirements for intermediate subcomponents in product supply-chains [30]. The boards had been arriving beginning July 2018, with 6 of them lacking the LEDs and one with a misplaced connector and lacking the LEDs aswell. As of this writing, no functional flaw or missing feature has been reported.

Due to increased shortage [31] on the MLC-capacitor market beginning April 2018, the bypass capacitors had been replaced by *PCBCart*. As of this report, it is not clear, whether worse MLCC quality, flux residuals, the reflow process, chip variation or firmware issues (on STM32L4 or BOLT, though hardly unlikely regarding the span of measured values) are the cause for the high variation in quiescence current up to 5.2 µA as seen in fig. 4.3.

Theoretically, the integrated circuit subsystems consume in total at least

$$\begin{aligned} I_{total} &\geq 0.008\,\mu\text{A (STM32L4 Shutdown-mode)} \\ &+ 0.050\,\mu\text{A (BOLT)} \\ &+ 0.160\,\mu\text{A (SX1262 cold sleep)} \\ \\ &\geq 0.218\,\mu\text{A} \end{aligned} \tag{4.5}$$

Spot measurements conducted by the author on three randomly selected communication boards also indicate such problems. The overall values were lower tough (down to

0.72 µA) due to longer measurement intervals resulting in the settling of the dynamic electro-chemical effects in MLCCs. The values have been overly smaller than in the pre-series boards most probably due to the tidier process involved compared to the manual hand-soldering and type of flux used by the author.

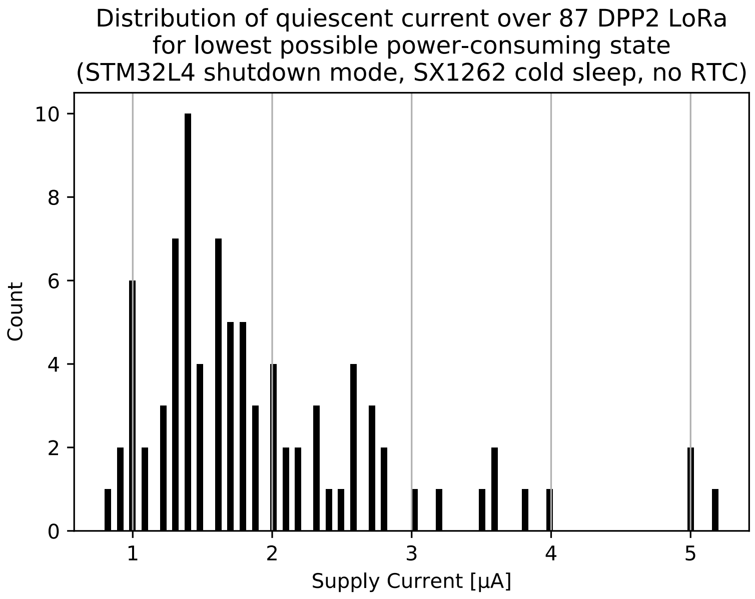


Figure 4.3: Measurements by R. Da Forno. Note the high outliers at 5 µA

4.4 Firmware Design

This section will give an overlook over important design decisions and configuration specialities of the communication board's firmware.

4.4.1 Structure

The firmware stack is called *flora* as a play on words on *LoRa* and *flock*. For an overview of its file structure and usage see appendix A.

4.4.2 Development Kit

Semtech provides its own (European) development kit for the SX1262 (aswell as variations of SX1261/2 for other regions) called *SX1262DVK1CAS*. It is including the radio circuit as a Arduino-compatible shield, a NUCLEO-L476RG MCU development board, and a RGB resistive touch screen display.

The also provide their own comprehensive MCU-independent *LoRaWan* stack, including drivers for SX1262 and other Semtech *LoRa* PHYs [32], which has been partly used as base for this thesis.

The development kit's demo application is yet written in an another framework which uses the Mbed ecosystem and a component support package [33]. It features evaluation of all differnt operation modes of the SX1262 directly via a the touch-screen interface on the development kit.

For the scope of this thesis, a STM32CubeMX setup for both development kit and communication board has been established to allow the actual firmware to be run with equal functionality on both targets, easening development. On the hardware side two minor changes have to be undertaken to make the development board compatible with the communication board.

1. `DIO1` has to be connected via a jumper wire to one of the TIM2 capture inputs.
2. As the STM32L476RG is lacking an external high-speed crystal, a precise clock has to be applied to the MCU via the MCO clock output of the built-on ST-Link, by ensuring the correct solder bridge configuration.

Both steps are described in much more detail in the official *flora-tools* documentation (see appendix A).

4.4.3 STM32CubeMX

ST provides a comprehensive low-level hardware configuration and middleware API ecosystem called STM32Cube. The associated STM32CubeMX is a graphical surface for configuration of the MCU's core and peripherals registers.

STM32CubeMX generates (and even regenerates) ready-to-build project scaffolds and stub code for different embedded-ARM IDEs such as Keil uVision, IAR Ewarm, Atollic TrueStudio, or SW4STM32 (with a method to decapsule the generated GNU Makefiles). High-level features, such as the integration of FreeRTOS and a FAT16 based filesystem by a single click can be quite useful. On the downside, STM32CubeMX's source code is not available, and that proven build systems such as CMake are not supported, making it a bit less fitted for *Free and Opend Source* based projects.

Configuration

The STM32CubeMX configuration is stored in the corresponding

- `platform/flora_dpp_comboard/flora_dpp_comboard.ioc`
- `platform/devkit_sx126xdvk1xas/devkit_sx126xdvk1xas.ioc`

XML-file of the main source code repository. *DPP2 LoRa*'s configuration is still based on the STM32L443CC as the ABI and feature-set beside the lacking in cryptographic extensions is identical.

The peripherals pin-interconnect matrix is not complete and requires clever planning for an optimal board-layout (also including an unbrickable bootloader fallback design).

The system clock was only set to 48 MHz due to an early bug in the STM32Cube-HAL setting the the PLL-divider to zero, which caused unexpected behaviour for higher frequencies. For the sake of maximum performance the system clock can be increased to 80 MHz in the future. The current setup can be seen in fig. 4.5. Keep in mind that this requires remeasuring and recalibration of the black-box delay figure (the time it takes for an external pin interrupt to get captured by the high-speed timer, plus the time it takes for a internal compare timer interrupt to set a pin). For calibration of the RTC clock, `PA2` (exposed by a connected VIA) can be used to output a 512 Hz clock based on the low-speed oscillator's 32.768 kHz.

TIM2 is used as the main timing and scheduling source for protocol and radio based operations (such as precisely timed Tx or Rx windows). Running at 8 MHz, it has 32-bit in counter depth, with an additional 32-bit extension in software). This allows for a granularity down to 125 ns with a total range of up to 73117.8 years. A capture register is configured to latch the current counter's value each time the SX1262's `DIO1` line shows a positive edge. Two output compare registers are used for the precise triggering of software

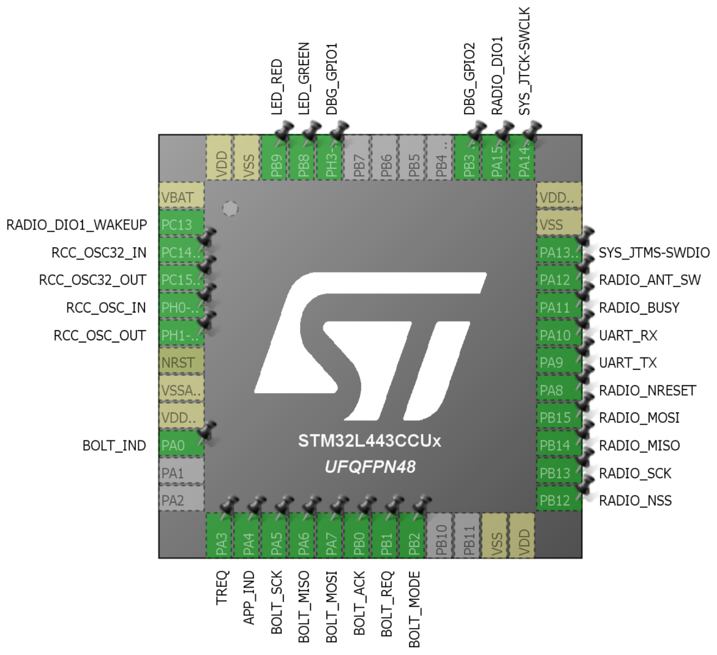


Figure 4.4: DPP2 LoRa MCU pinout as seen in STM32CubeMX.

interrupts (such as setting the radio into Tx-mode or as a fallback timeout counter for dead-lock safe operation). Routing `NSS` to `PB3`, using TIM2's dedicated compare-register pin-output, precise operation could become even completely independent of software timings. The TIM2 interrupts are configured to have the highest priority level, preempting other interrupt sources (UART, SystemTick) and system loop.

RTC is configured to also run internally with a 32.768 kHz tick frequency. The minimum granularity regarding RTC alarms and interrupts is therefore 30.517 57 µs. For wakeup from Stop Mode, one has not to forget to enable `EXTI18` via the NVIC configuration pane.

The RTC's backup registers (which are powered-on also in standby- or shutdown-mode) are used as a boostrap mechanism for jumping into the integrated ROM-bootloader in a well-defined state from the running system. This has proven especially useful for the mass-programming of the development kits, as `nRESET` and `BOOT0` line were not accessible via a RS-232 control signal (CTS, RTS) as common in embedded engineering.

USART1 is used for communication via the CLI and runs at 115'200 baud (8N1) with DMA1 channel 4&5 enabled. SPI1 interconnects with BOLT, SPI2 with the radio chip.

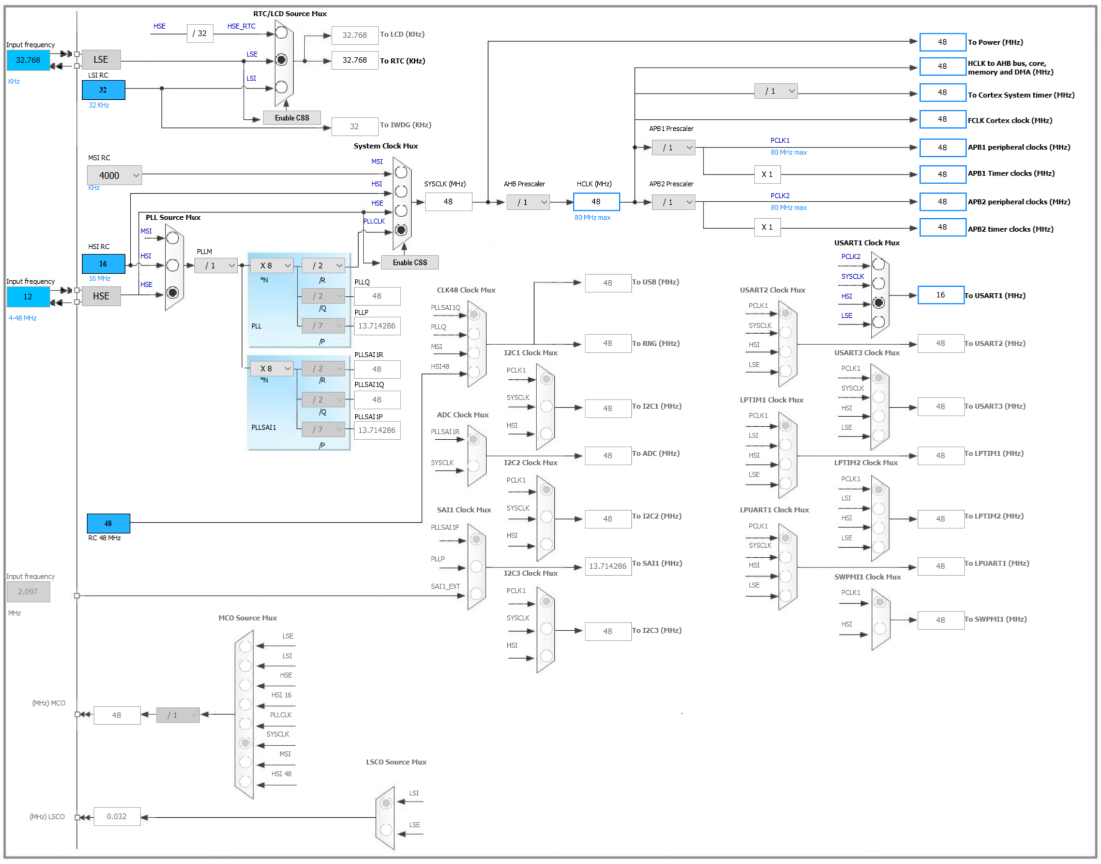


Figure 4.5: DPP2 LoRa MCU clock configuration as seen in STM32CubeMX. Note the two input frequencies 32.768 kHz from the low-speed crystal and 12 MHz.

For debugging and monitoring purposes, a red and green LED are available on the edge of the board. While the former can only be turned on or off, the latter is driven by a high-resolution 24 KiHz PWM signal from TIM16 using dimming for advanced signalisation patterns.

The last 2 KiB block on flash memory is configured to store the node's configuration (unique ID, type of node). It can be set dynamically via the CLI. The author's motivation is to have only one single firmware image for all nodes and setup the configuration via a script seperately.

Further configuration changes can be easily done by installing STM32CubeMx and opening the regarding `.ioc` file. The newly generated code will only overwrite the setup routines. User code is untouched.

4.4.4 IDE Configuration & Debugging

We use the freely available Atollic TrueStudio for STM32, an Eclipse-based IDE, acquired by ST in 2017. Its key features include support for C11 via the GNU GCC 7 ARM compiler (benefiting of *anonymous structs*), newlib-standard & newlib-nano, an integrated visual GDB debugger interface for ST-Link and Segger J-Link, FreeRTOS utilities, and code completion.

To maintain support for collabarative editing of the project files (include path information, symbols, etc.), the author added support for patching the project's XML files (.project & .cproject) inside *flora-tools* to keep include directories in sync (see appendix A for further information).

Via a Segger J-Link Pro connected to the FlockLab node and intranet, one can even use the IDE to remotely run the debugger in a real-time fashion.

4.4.5 Bare Metal or Operating System?

Former TIK projects built on dedicated mote operating systems such as the free and open source Contiki OS or TinyOS. While their use in academia is still present, a large part of industry and academia builds upon the more task scheduling and IPC centric FreeRTOS.

While Contiki OS tries to shape the target MCU and platform bindings to match the OS, FreeRTOS does have less conventions how it can be integrated in a new platform, or how interfacing with the interface has to look like.

Support for low-power modes is enabled in FreeRTOS by a callback when all tasks are idle [34]. Enabling FreeRTOS on the STM32L433CC requires some additional RAM and Flash ($\approx$ 1 KiB RAM and $\approx$ 4 KiB ROM) which is negligible compared to the use of stdlib's `printf` features and amount of buffer space used.

As the low-level design for a extremely precisely timed wireless stack had top priority, the author decided to base the stack on a bare-metal approach, enabling FreeRTOS in the future.

The author expects off-the-shelf ultra-low-power multi-core MCUs to arrive in the near future with FreeRTOS as one of the early OSs to support simultaneous low-level multiprocessing (see PULP [35]).

4.4.6 Command Line Interface & Scripting

For development, testing and demoing purposes, the author decided to develop a small and modular command line interface for UART featuring auto-completion, command expansion, command history, VT100 support (colors and navigation), help utilities and mode switching between human readable output and JSON-encoded machine output for more complex testing scenarios. It is the intention of the author to release the CLI part of the code as a standalone library in the near future.

The command line interface's input- and output-FIFOs are processed via non-blocking circular DMA routines interconnecting with the UART peripheral at 115'200 baud (8N1). The input gets processed every 10 ms in a non-preemptive, non-time-critical manner, processing iteratively every new character in the input buffer.

Full documentation of the *DPP2 LoRa* commands is available via the `help [command]` command or in the base repository. The CLI has already proven useful in the lab for using the *DPP2 LoRa* as a jamming device via the continuous-wave (`radio cw`) command.

4.4.7 Low-Power Operation

To minimize energy consumption in case of idle-operation, various low-power modes can be employed.

One key aspect of having a reliable, yet tight sleeping schedule, is the drift induced by the clocks. Assuming ± 25 ppm clock accuracy, one needs to plan for operations such as switching into Rx mode to start more early and finish later depending on the time passed since the last synchronization.

For this thesis, the author assumes the worst case, in which mutual nodes might have a differential clock drift of $\pm 2 \cdot 25$ ppm $= \pm 50$ ppm (this can happen e.g. during winter, when sun-exposed and snow-covered nodes see very different temperatures). Drift compensation and calibration based on a precise time synchronization scheme can mitigate this effect. Yet assume the average Rx current consumption to be $\approx 6\,\text{mA}$: To mitigate the uncertainty regarding drift offset we have to extend our Rx window on each side by 50 ppm for every second that passed since the last synchronization. Assuming that every consecutive transmission is received successfully and contains timing information for resynchronization, we only have to account for one Rx extensions. As the extension scales lineraly with the time since the last synchronization the additional power consumption can be seen as independent of the synchronization period. The extension of the Rx window results in an additional power consumption of

$$I_{\mathrm{Rx}_{\mathrm{overhead}}} = 6\,\mathrm{mA} \cdot 2 \cdot 50\mathrm{ppm} \tag{4.6}$$
$$\approx 600\,\mathrm{nA} \tag{4.7}$$

Assuming that no synchronization message will be received by the radio, the overhead will scale with the allowed number of missed synchronization messages + 1, times the slot count for LWB/Gloria. Further research in this area has yet to be done.

4.4.8 MCU

The STM32L4 supports several highly configurable operating modes [12, p. 12], namely (from highest to lowest consumption)

- Run Mode: Maximum performance with up to 80 MHz clock frequency.
- LP-Run Mode: Reduced voltage and clock up to 2 MHz with higher efficiency.
- Sleep Mode: Processing is waiting for interrupt. Core's clock is gated resulting in less power consumption. Flash can be powered-off. Wakeup source can be all interrupt sources.
- LP-Sleep Mode: Similar to LP-Run Mode, voltage and clock is reduced. Flash can be powered-off. Wakeup source can be all interrupt sources. It consumes $\approx 140\,\mu\mathrm{A}$ with RTC, TIM2 and USART1 enabled on the STM32L433CC.
- Stop Mode (0, 1 & 2): SRAM and peripheral register retention. All high-speed clocks are disabled. Wakeup source can be character recognition at peripherals (e.g. I2C address, UART code), RTC, dedicated wakeup pins, or reset line. Stop Mode 2 (with RTC) consumes $\approx 1.3\,\mu\mathrm{A}$ of current.
- Standby Mode: Only RAM2[3] can have retention enabled. Brown-out dedection and voltage regulators are still enabled. Pins have to be explicitely enabled for output. Wakeup source can be RTC, dedicated wakeup pins, or reset line. It consumes $\approx 100\,\mu\mathrm{A}$ with RTC enabled.
- Shutdown Mode: Voltage regulators are disabled. Wakeup source can be RTC, dedicated wakeup pins, or reset line. It consumes $\approx 260\,\mathrm{nA}$ with RTC enabled.

The author has decided to use Stop Mode 2 as base for deep-sleep operation. As the firmware uses around 13 KiB RAM with Gloria and over 25 KiB incorporating LWB, reinstantiating state in Standby-Mode would have had increased the code's complexity

[3] Part of the SRAM (16 KiB) can be kept powered while disabling the other (48 KiB)

by alot (i.e. loading static data, reconfiguring IO, reloading funciton-call stack). With Stop Mode 2, only the system's high-speed clock configuration has to be reloaded. When not having enough time for the MCU to go into Stop Mode 2, it switches into LP-Sleep Mode (making use of the Batch acquisition sub-mode for low-power IO transfers).

It is without question that the Shutdown- or Standby Mode are the most optimal low-power modes for times above 0.1 s [36, p. 17], and should be integrated as well in the future.

4.4.9 Radio

The SX1262's timings regarding sleep mode are quite large in comparison to the the STM32L443's wakeup times (7 µs for Stop Mode 2): Wakeup from the cold state (160 nA, without register retention) will take $\approx$ 4.5 ms (in contrast to the datasheet with $\approx$ 3.5 ms [13, p. 52]), respectively $\approx$ 340 µs from warm state (600 nA).

4.4.10 Event-Driven Design & Driver

For best power-consumption and performance the author decided to structure the radio's operation in an event-driven manner. Each asynchronous operation is followed by a registered callback which is invoked in an interrupt from TIM2. This may also be the capture interrupt caused by the `DIO1` line from the radio.

Semtech in contrast supplies a driver structure and examples containing a main loop for processing the IRQs and polling timers, wasting precious resources. Due to the changes by the author, additional functions and some minor tweaks and bugfixes were neccessary on the original Semtech driver (see the GitHub issues by *atokulus* on [32]). Since its relase in March 2018, the driver did not get an update though, and the author's version is therefore still up to date.

4.4.11 Mass-Programming

By using the CLI (`system bootloader`) or applying an active high level on `BOOT0` pin (`DBG_GPIO1`) and resetting the MCU via `nRESET`, the MCU can jump into the integrated ROM-bootloader, which allows to program the target via other means (including UART, I^{2}C, USB, CAN, etc.) than just JTAG/SWD. The bootloader mode is indicated via a constantly on red LED on the communication board.

A Python 3 script integrated into *flora-tools* with contributions by the author [37] implements ST's UART booloader protocol, allowing to mass-program multiple devices at once, and requiring only a serial port to each target. The script is also used for target programming in FlockLab.

4.5 Radio Operation

The SX1262's way of operation is much more simplified compared to SX1276 with its built-in configurable multi-timers and state transitions. Using a minimal set of states for hybrid *LoRa* and GFSK communication, similar package engine between both modems, and minor higher-level control features such as the *LoRa* CAD (channel activity detection) and *sniff mode* [13, p. 70]. The radio's interface design is very non-prohibitive regarding unusual state transitions or payload buffer operations by the user. fig. 4.6 shows an overview of all operating states. The default operation would look as follows:

On the initialization, the reset radio is put into Standby mode, in which we configure the radio (center frequency, modem parameters, payload). In this mode, the radio's internal oscillator is running to provide the SPI interface.

We then have the option to switch into Sleep mode to reduce power consumption in case of idle operation. The SPI interface is shutoff. Reviving operation involves to toggle the `NSS` line low and high and waiting for the `BUSY`[4]signal to get low.

Other options involve putting the radio into Rx mode which for active reception, or Tx mode for transmission of the set payload. Both transitions involve the enabling of the external 32 MHz crystal. After completion due to having received a message (Rx), a timeout that occured (Rx, and even Tx) or the transmission of a message (Tx), the radio returns to the Standby mode.

Special cases involve (non exhaustive) the *sniff mode*, continuous-Rx or crystal activation in Standby mode. Have a look at the the datasheet for the full spectrum of options [13].

Experiments by the author have shown that commands and their operating mode transitions in the SX1262 can be precisely triggered by timing (i.e. delaying) the rising-edge of the `NSS` (see fig. 4.7).

[4]Indicates that the SX1262 is either internally processing the last command, recalibrating or sleeping. The SPI interface not available.

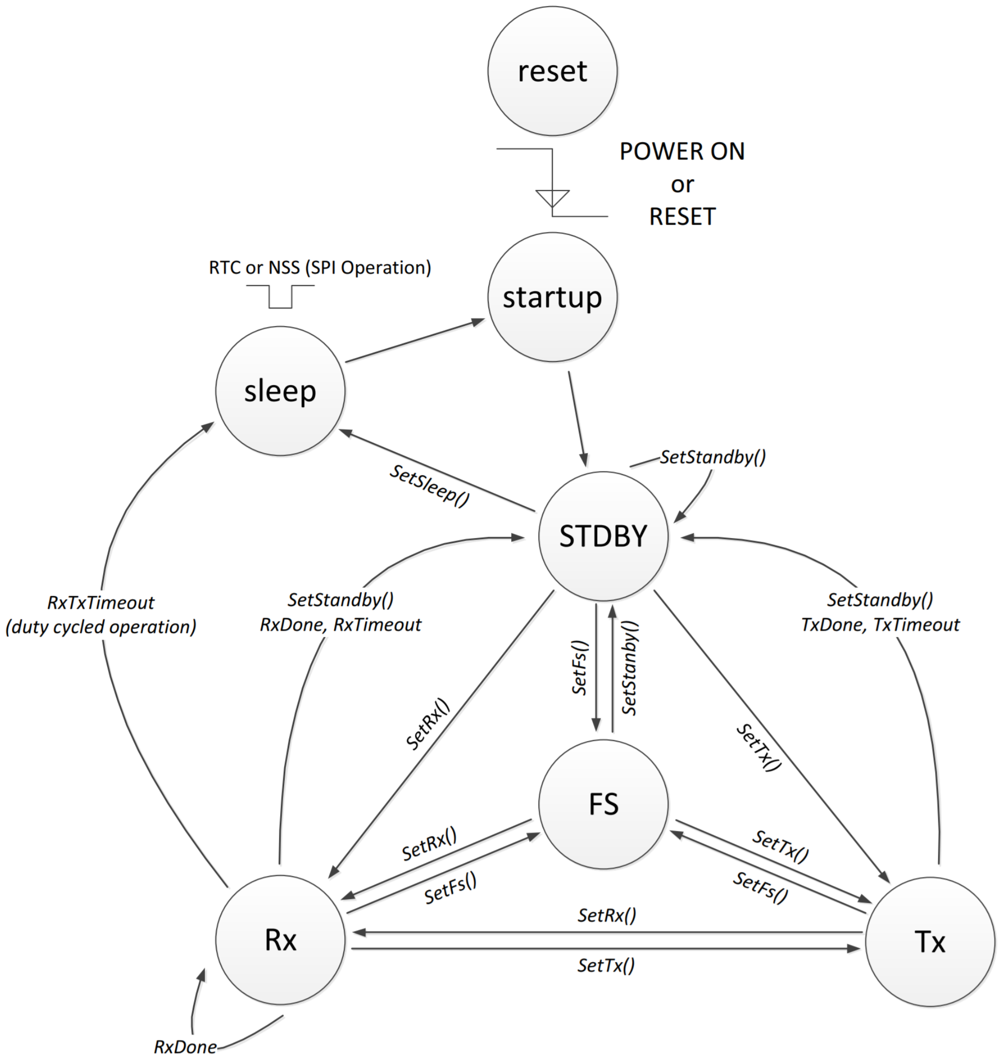


Figure 4.6: SX1262's operating modes and transitions as presented by Semtech [13, p. 59].

Commands that are allowed to be timed precisely by the rising-edge of `NSS` are among others

- `SetTx`: Sends the message in the radio's payload buffer according to the preconfigured settings.
- `SetRx`: Puts the radio into the receive mode.
- `SetRxDutyCycle`: Enters the *sniff mode*.

Some commands have not such behaviour: They execute directly after the last SPI clock cycle, independent of `NSS`. Due to the nature of the STM32Cube SPI drivers, the time

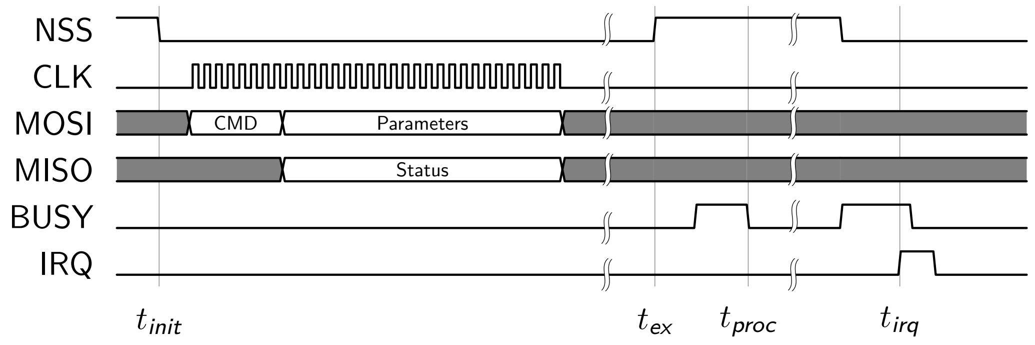


Figure 4.7: The figure shows the author's way of timing certain commands on the SX1262. The moment of `NSS` to rise can be arbitrarily delayed (t_{ex}), resulting in the execution/processing of the command with a precise offset just after the `BUSY`'s falling-edge (t_{proc}). Internal timeout counters only start after t_{proc}. See [13, p. 49] for further details regarding the SPI interface.

taken for a SPI transfer is very difficult to precisely control due to overheads and non-deterministic memory bus behaviour. Some commands that are executing directly are listed below. Being able to time these command would haven been useful for the following reasons:

- `SetCAD`: For more precise CAD information.
- `SetTxContinuousWave`: For custom-built OOK schemes.
- `SetTxInfinitePreamble`: Useful for the precise in-sync jamming of another *LoRa* signal for research purposes and conformity measurements for certification.

4.5.1 Timing Analysis

For safe and precise operation of a radio it is crucial to know its inner workings in much more detail than from the datasheet's description. Especially the timings (i.e. how long does a state transition take) are quite important in defining a protocol's slot times and are useful for modelling purposes.

To have a robust, precise and easily replicable measuremnt setup for a high amount of experiments and datapoints, the author decided to run a series of measurements on the development kits using a Tektronix MSO4104B oscilloscope and TAP1500 probes allowing for a channel skew below 1 ns and sample rate up to 20 GHz. The oscilloscope is interconnected via LXI/GPIB to a computer, which is setting the radios' and oscilloscope's configurations automatically, triggering the experiments, capturing the waveforms, and analyzing the data.

Up to three development kits could be interconnected via 50 Ω coaxial line with −40 dB attenuators attached to protect the radios from destroying each other (see fig. 4.8). The on-board wire lengths have not been time-compensated due to their very small contributions to delay (10 cm 0.33 ns).

For ease of access, the probes can be connected without the grounding clip on the bottom side of the NUCLEO-L476RG board on the large header connector `CN5`: `NSS` Pin 27, `DIO1` Pin 29, and `BUSY` on Pin 31. For TIM2 synchronization measurements, `DIO1` on the synchronization sink has to be connected to `DBG_GPIO1` (`PC9`, `CN5` Pin 1). For more than one sink the signal has to be buffered with a high-speed schmitt-trigger or ampflifier. The author used a low-capacitance P-MOSFET (TP0610L/T) and passive pull-down for his measurements.

Key figures that were measured (using the time definitions of fig. 4.7) contain

- $t_{ex} \rightarrow t_{proc}$ for Tx, Rx, CAD, *sniff mode* operation.
- $t_{ex} \rightarrow t_{irq}$ for Tx, Rx, CAD operation (successful and failing)
- $t_{init} \rightarrow t_{ex}$ for configuration, payload transfer, sleep, wakeup operation.
- *sniff mode* timer precision and overhead.
- Timing & synchronization precision of MCU's TIM2.
- Simultaneous Tx of two radio's and the impact of mutual offset.

Other experiments contain feature evaluation such as

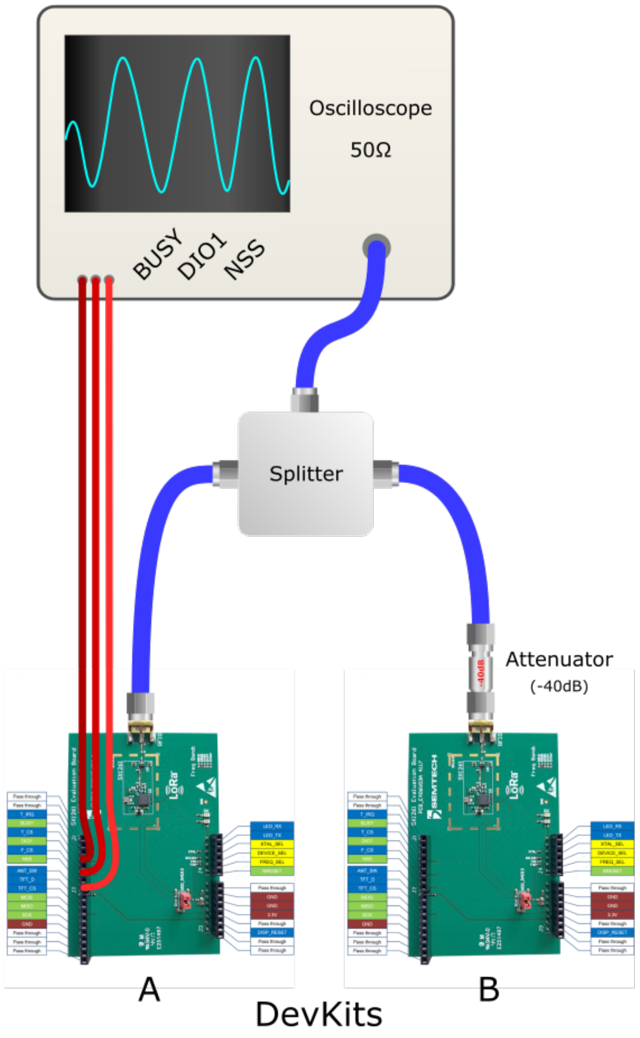


Figure 4.8: A simple timing-measurement setup: The three digital lines NSS, DIO1 and BUSY are being captured from one single development kit.

- Simultaneous writing/reading of SX1262's payload buffer while in Tx/Rx.

Due to the many different transitions and possible operations, the author conducted over 25 distinct experiments. As base for diversity and different radio configurations, the full set of available ISM bands, payload lengths (0 B to 255 B) and power levels (−9 dBm to 22 dBm) has been used in a randomized way. Modulation and bandwidths are based on table D.1. See appendix F for the complete result bulletin.

4.5.2 Discussion

Measurements of timing characteristics cannot provide a real upper limit which is safe under every possible setup. Our perspective on the SX1262 is that of a black box, with complex internal processing we cannot understand in full detail. The author's attempt trying to isolate possible internal ROM firmware by reading and searching the complete register access range (65536 bytes) for magic bytes and RISC-shaped instructions did not succeed.

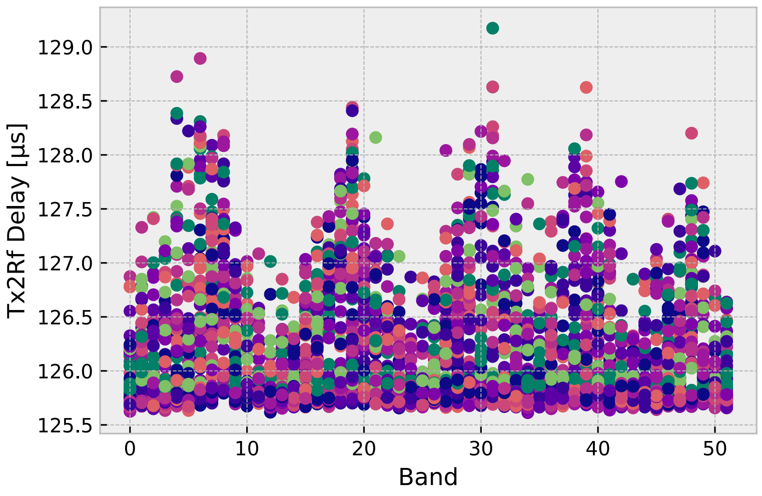


Figure 4.9: The delay of the radio going into Tx mode ($t_{ex} \rightarrow t_{proc}$). Note the dependency on the the selected band. Yet the synchronization accuracy does not depend on the selected band.

Nevertheless by randomly changing the radios' configuration, including frequency, power, modem, payload length and other aspects, we might have included a very large portion of the possible design and state space. There has been no event known to the author, besides messages from foreign nodes and the occassionaly crashing GPIB interface, to cause real outliers in the data way beyond the major distributions.

The sample count of each experiment was limited by the amount of time taken (e.g. the ToA of a packet), the amount of data to be transported over GPIB, its significance for the overall protocol design (e.g. we are not really interested in the implicit header mode measurements).

The following paragraphs give some interesting findings regarding the SX1262.

Synchronization

One could use several principles to reconstruct the timing information of a packet, namely:

- *Preamble detected* IRQ: The radio will fire an IRQ as soon as it has detected a preamble, which can be at an arbitrary position of the incoming preamble. It is threrefore not considered to be precise.
- *Rx Done* IRQ: The radio will fire an IRQ as soon as it has completely received the message and processed the CRC. The IRQ does not accurately fall on the ending of the received message: It seems as if the overhead symbols at the end of the transmission (addressed earlier) do only partially contribute. By using a lookup-table for each possible payload length we can have standard deviations as low as 3 µs. See fig. 4.10
- *Header Valid* (*LoRa*) & *Syncword Detetected* (GFSK) IRQ: The radio will fire an IRQ as soon as it has received a complete valid header (*LoRa*) or when it detects the syncword appended to the preamble (GFSK). This is the most precise way of time information recovery and can be suitable for time-of-flight aware time synchronization schemes. It comes yet with an expensive overhead in case of small packages. See fig. 4.11

Constructive Interference

Having three development kits interconnected with each other, the author ran an experiment evaluating the effects of one (fig. 4.12) and two (fig. 4.13) slightly offset transmissions (with the same power level) and one Rx window.

The Rx window is dimensioned to be the same length as the preamble's ToA. A relative offset of zero means for a given node that its Tx preamble and Rx window are perfectly aligned. A relative offset of *-1* means that the Tx preamble's end is coherent with the Rx window's start, *+1* means that the Rx window is prepending the Tx preamble. The measured zone is triangular in shape, as every combination can be achieved.

Based on the results the author's conclusion is that for high signal levels (we are yet in a cable-setup), the *capture effect* is dominant, picking the first occuring transmissions

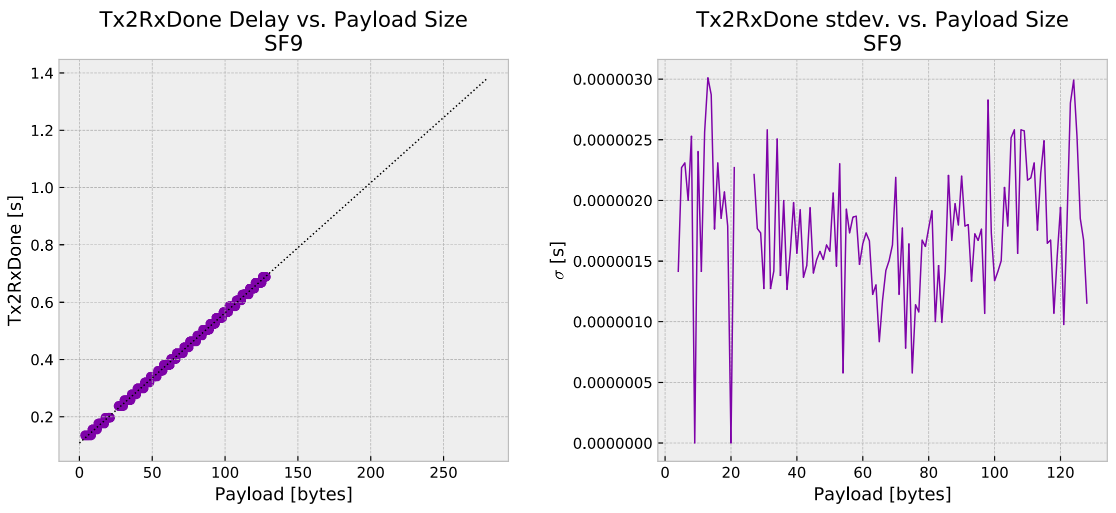


Figure 4.10: Measurements depicting the delay in transmission t_{ex} and full reception t_{irq} given the payload length for SF9. Using the *Rx Done* IRQ, one can get quite accurate results with a standard deviation below 3 µs.

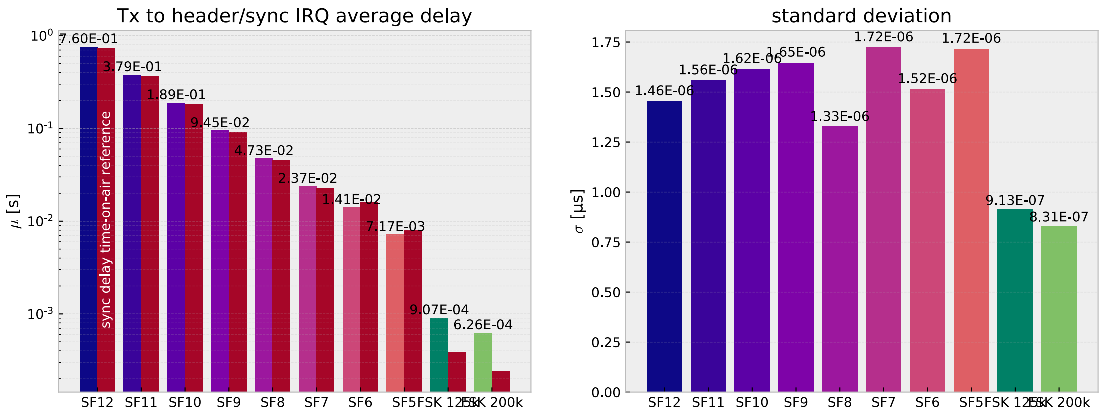


Figure 4.11: *Header Valid* (*LoRa*) & *Syncword Detetected* (GFSK) IRQ based synchronization can deliver precise results. Interestingly the slow modes such as SF12 do not show worse (even better) standard deviation than faster modes. The accuracy seems therefore independent of symbol time (which varies exponential for the different modes).

(if not due to power level mismatch). This observation would be in line with [20]. Due to the high fluctuation in SX1262's SNR estimation (±10 dB)) and offsets below symbol length, a reliable statement about *constructive interference* is hard to be made from this data. More elaborate measurements on FlockLab, hand-tuning of the power level with an adjustable attenuator (as available in the laboratory) and spectrum analyzer, or even a the replication of the signal with a software-defined-radio are required.

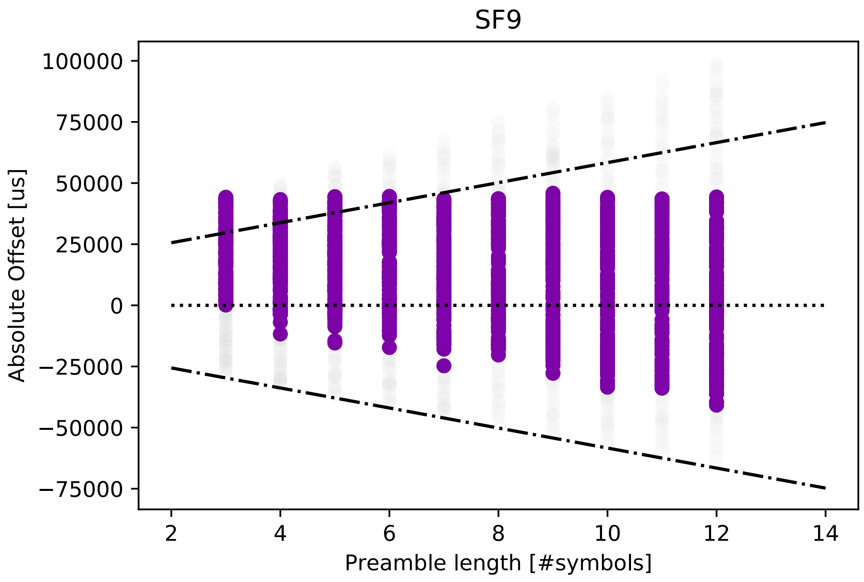


Figure 4.12: Tx-Rx sweep for SF9. Colored dots represent successful reception, gray dots failed ones. The lines mark the relative offset range of ±1

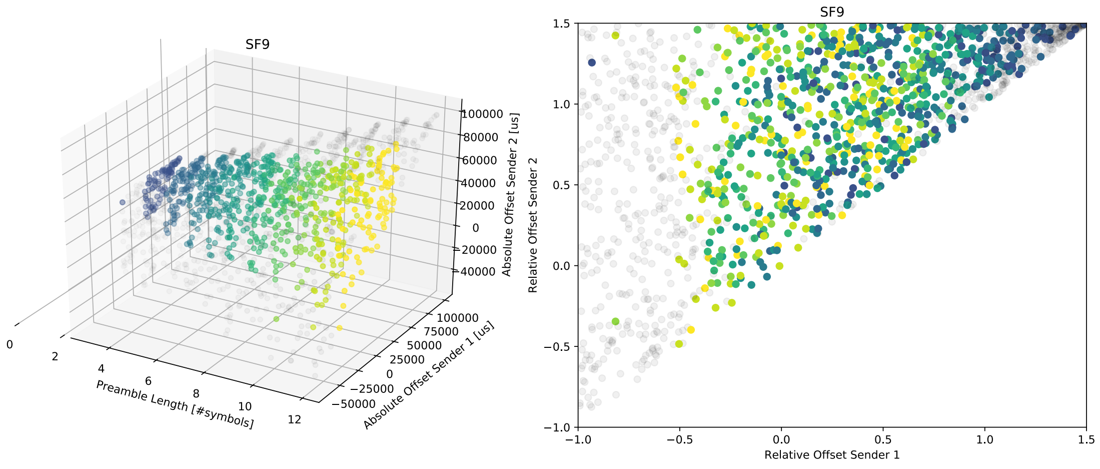


Figure 4.13: Tx-Tx-Rx sweep for SF9. Colored dots represent successful reception, gray dots failed ones.

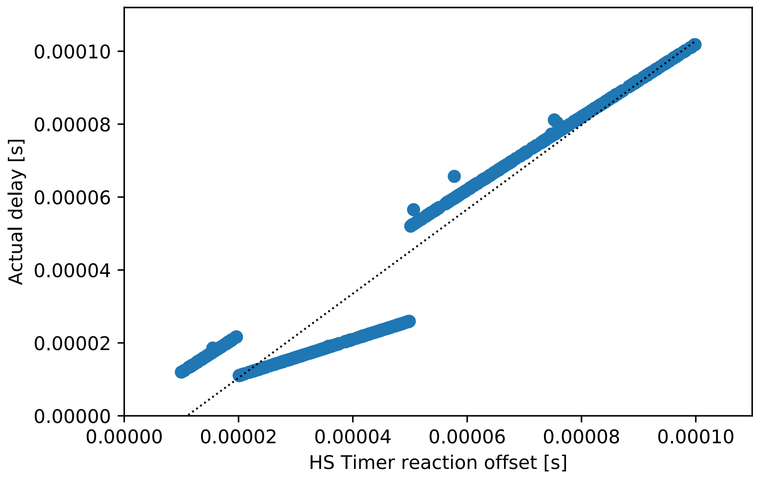
Another very important observation is the fact that the Rx node seems not to receive a packet in case of a very short preamble already being sent when the Rx node switches into Rx mode. The author assumes that the the automatic-gain-control cannot adapt its noise floor level in such a short time. Best practice has shown to be extending the Rx windows some symbols before the actual transmission begins. This is in contrast to [38, p. 9], where an Rx window can begin after the first preamble symbol.

MCU TIM2 Synchronization

Precise time-synchronization requires also to have exact data on the MCU's capture input and compare output. Similar to the radio chip, the MCU can also be seen as a block box. We can only measure the time it takes for an input signal to provoke an output reaction. We cannot know the precise moment when a timer's interrupt get's processed without deeper knowledge of the inner architectural details.

To measure and tune TIM2, we apply a rising-edge signal at the capture input pin, schedule an operation for the future and then acquire the offset the reaction (rising-edge on an output pin) takes for the scheduled operation.

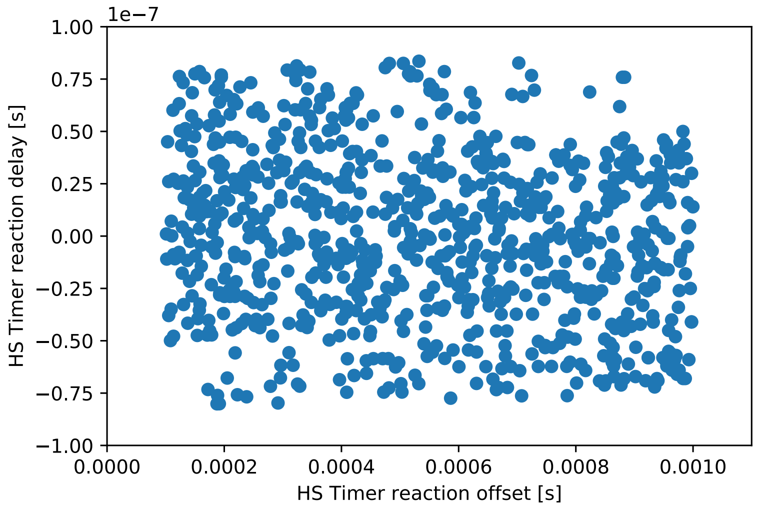
We can recalculate the overall *black-box delay* ($\approx 2\,\mu s$) and give measures on the overall precision (see fig. 4.13).



*

Figure 4.13a: Measurements regarding the accuracy of TIM2 compare interrupts for different times below 100 µs in the future indicate erroneous behaviour. Above 100 µs, the timings are in line.

The limitation of TIM2 is that the scheduled moment should be at least 100 µs in the future as indicated by the measurements in fig. 4.13. The compare interrupt won't execute on a given precise moment otherwise.



*

Figure 4.13b: The datapoints depict the deviation from the linear regression of the actual offset for a TIM2 compare interrupt in the set range 100 µs to 1000 µs. This allows to make assumptions about the precision of TIM2 regarding reactions such as setting a GPIO. We reach an accuracy of ≈ 175 ns, way below the radio's precision.

4.6 Undocumented Details regarding SX1262

The author has tried different *hacks* regarding the improvement of certain details.

- Custom *LoRa* syncword: Via a register, custom 2-byte syncwords can be specified, serving as a method to add a way of network addressing to the nodes. Other syncwords than the prespecified and documented *public* and *private* ones are not that reliable and do not allow for the high sensitivities (as tested on the FlockLab). The author can therefore only give limited recommendation to change the syncword to a different value, e.g. for near-range networks with high RSSI.

- Received data can be retransmitted without reading and writing again the radio's payload buffer. One can read and overwrite just part of the payload buffer (e.g. the header). This allows for fast retransmissions that may be required in protocols such as Glossy.

- Writing and Reading while Rx or Tx operation is in process: This allows to chain transmissions together with minimal delay. Writing the payload has to be controlled very cautiously, as overrunning the radio's payload pointer will result in part of the new payload being transmitted (with correct CRC!).

4.6.1 SX1262's Specialties

The SX1262 features several improvements over its predecessor.

- *sniff mode*: Puts the radio into a periodic sleep-receive pattern, allowing for long preambles to wakeup the device. As TIK's protocols a highly synchronized and precision of the internal RC-clock is quite bad ($\approx 0.5\,\%$), this mode is more interesting for less optimized protocols.
- CAD (channel activity detection): This *LoRa*-only feature allows to recognice channel activity even in between packets and does not depend on the RSSI threshold-principle of the earlier radios. As the CAD is very fast and can directly jump in Rx mode, it is used as a bootstrap mechanism for Gloria/LWB.

4.6.2 Erroneous Behaviour

Some things would not work out on our development kits such as the *sniff mode* in GFSK (ending in a corrupted state with glitches on the `BUSY` signal), the `SendInfinitePreamble` command would retransmit the previous package which led to quite some confusion, and power levels and preamble length had been cropped in the depths of Semtech's driver.

Regarding the SX1262 being a new product, and pre-series worked flawlessly we are quite happy.

4.7 FlockLab Integration

As the communication board is compatible with its predecessor *DPP2*, only minor changeson the adapter boards ID memories and FlockLab backend had to be undertaken by R. Da Forno. 18 new communication boards have been installed on different observers.

Additionaly, the author assembled a SWD adapter cable to make communication board remotely debuggable via a Segger J-Link Pro.

Chapter 5

Protocol Design & Implementation

This chapter introduces a new flooding-mechanism called Gloria which builds upon experience with Glossy. The author will give insight and arguments regarding his design decisions and show some approaches for performance improvements. A new event-driven simulation model and it's visualization and analysis platform is pre- sented. The chapter then concludes with the firmware implementation details.

5.1 Rules for a new Protocol

The new *DPP2 LoRa* communication board and the elaborate evalution of different modes span up a very large design space for a low-power, yet fast and reliable communication scheme. The author had to find an approach which can take use of all the SX1262's benefits, ensuring requirement- compliant allocation of the available resources time, energy and throughput.

The new protocol is designed bottom-up, ensuring that timings and operations are actually feasible in hardware.

Even without a basic idea about how to design our protocol we can make rudimentary assumptions and rules, which might limit our design space and easen overall complexity.

5.1.1 The right configuration

Experience has shown that the best performance in terms of power and time is achieved most often by applying the highest possible bitrate, even if this implicates setting the highest Tx power level available (in our case 22 dBm).

One key aspect is the very high efficiency for radios (such as SX1262 and our predecessors) to have their peak Tx power efficiency at their maximum power level. We can show this by simply studying the datasheet values in [13, p. 30]. Including the HF-switche's (measured) insertion path-loss of 1.5 dB and respecting the actual Tx power emission values, we can reconstruct the Tx efficiency as seen in fig. 5.1.

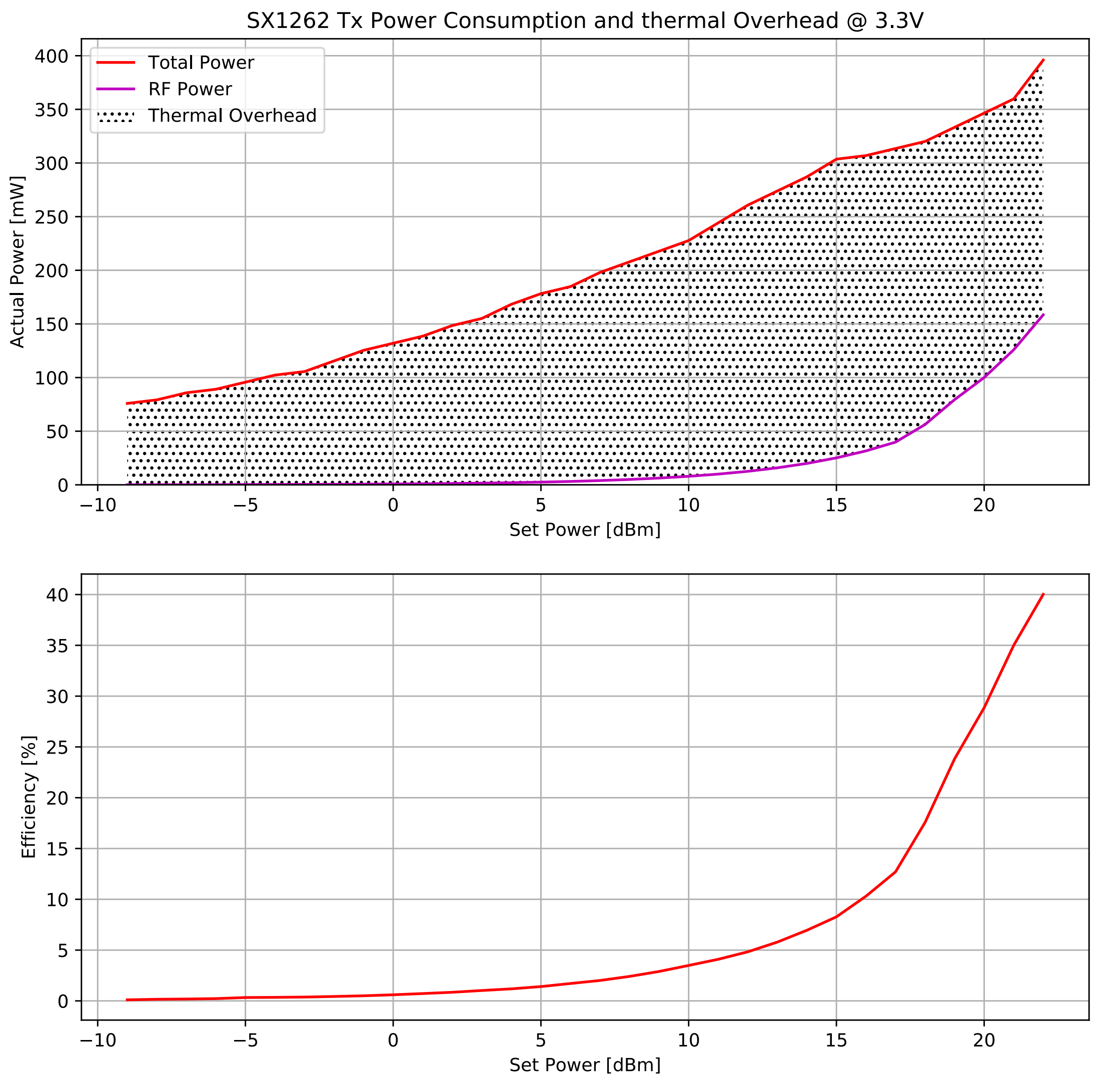


Figure 5.1: For a set Tx power level on the SX1262 we get different levels of actual emitted Tx power and wasteful power consumption, most of it being dissipated as heat and some as unwanted interference, in- and off-band.

Evaluation of the different configurations shown in table E.0 confirms this suspicion. Calculating the optimal transmission mode for each path-loss and power level given by it's Tx power consumption per single Bit (near the radio's sensitivity range), we can show, that the decision is rather trivial:

The optimal configuration is chosen by sorting and looking up the configurations with sensitivities lower than path-loss plus power level and lowest energy-per-bit consumption. The energy-per-bit calculation regards the different packets overheads for maxium payload length (255 Bytes).

The arbitrary set margin (5 dB) (to redcuce package-error-rate or *PER*) can be interpreted as additional path-loss shifting the heatmaps on the x-axis in left direction. For more detailed data on PER, coderate and sensitivity see [39, p. 5].

Marking the path of optimal configuration and power level for a given path-loss, we find that by simply sending with highest Tx power on all modes besides the fastest one, we can approximate an optimal solution. For the fastest mode (GFSK 250 kb/s), we try to limit overpowering. Our protocol should therefore follow this consideration.

5.1.2 Multi-Hop or Star-Topology

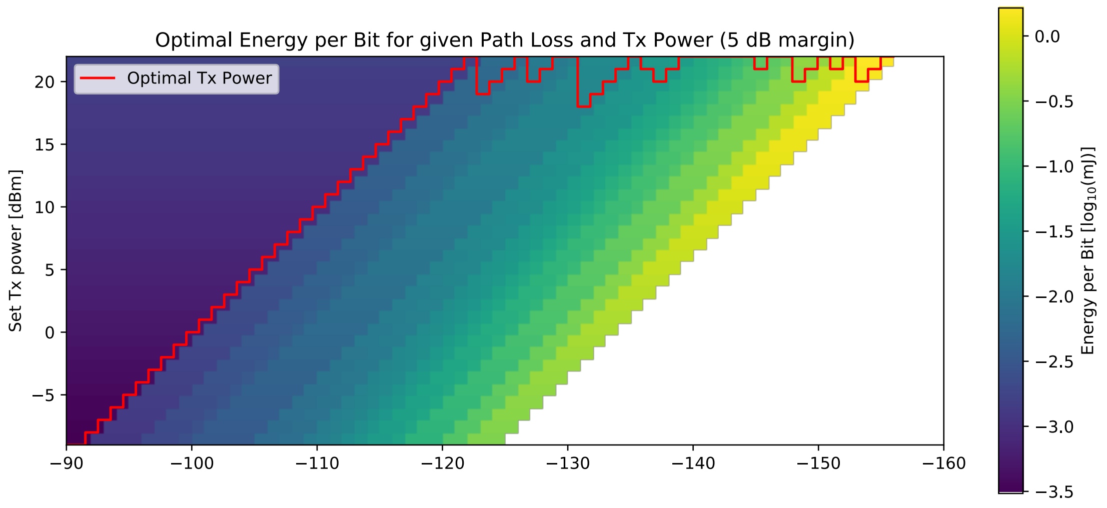
It is without question, that multi-hopping at highest Tx power levels can increase the range of a network drastically. But what happens with more tightly coupled and near-range networks, where we can freely set Tx power and still be fully connected?

Yet we can find ourselves in the situation, where a group of nodes can be interconnected directly from a base node at the fastests transmission mode in one single hop, implying the network to form a star topology. Such a toplogy can be encountered in very near-range node groups.

Assume that the same network could be interconnected via a lesser fixed Tx power level by relying on the additional hops that might show up. The ideal case for such additional hops to occur would be to be distributed on equidistant sections (splitting the path-loss in uniform way) between the nodes of the weakest link (highest path-loss), i.e. the same energy is used between all involved nodes.

As the distances decrease, path-loss also decreases, implicating lesser required power levels and therefore less Tx power consumption on a single node for a single message. The additional transmissions, Rx and CPU processing induced by the hops counteract this effect.

By assuming free space path loss with different path-loss exponents, one can analytically calculate the power budget for different scenarios. A low exponent such as 2 represents



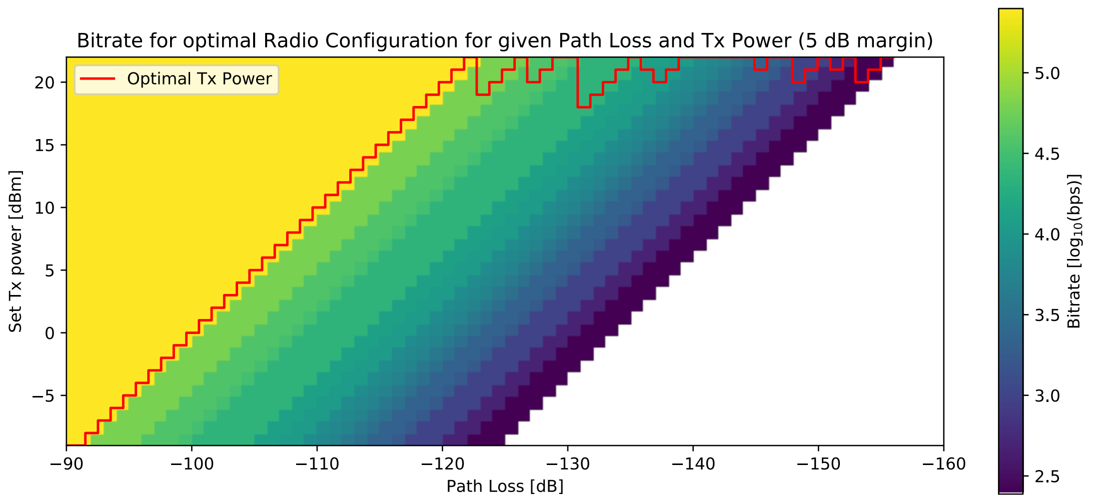


Figure 5.2: Upper heatmap: The colors indicate the best possible consumed energy-per-bit for a given path-loss and set Tx power level on the SX1262. For the white region, we cannot establish a link. The red path shows the optimal configuration in case of being able to choose the Tx power level for a given path-loss.

Lower heatmap: The colors indicate the bitrate for the best possible configuration.

Note the consistent diagonal regions with homogenous color differences: They represent the different configuration modes.

long-range line-of-sight topologies where an additional hop does not account for large power reduction, while an exponent of 5 has the characteristic of a more chaotic topology with complex obstacles such as rocks or concrete walls, and where hopping around obstacles in short distances is expected to be much more rewarded in terms of power consumption (and reliability).

The most benificiary point of having additional hops would be if the star-topology network required 22dBm to work, due to the fact that the power-dissipation is reduced the most the that operating point for every decibel less power (see fig. 5.1).

To calculate the total net-power-consumption (we could be sending at different bitrates, and have therefore to normalize in time), the geometrical assumption introduced by the FSPL formula takes a very important role.

Recall that for an exponent of 2, the spherical surface around a node is only a quarter the size at half the original radius, thus having four times the power-density on its surface. We can therefore only send with a quarter of the original Tx power level to encounter the same power density at half the radius.

The evolution of path-loss can also be simplified by stating that each doubling of distance will require $\approx exp \cdot 3\,\text{dBm}$ more power. For a given n hop count (with $n = 1$ being star-topology), and set FSPL exponent exp we can calculate the newly required power

$$P_{\text{Tx}_{\text{required}}}(n, exp) = \frac{P_{\text{Tx}_\text{r}\text{equired}}(1, exp)}{n^{exp}} \tag{5.1}$$

The total power over all nodes then accumulates to

$$P_{\text{total}}(n, exp) = \text{Hop Count} \cdot (P_{\text{Tx}_{\text{consumption}}(\text{P}_{\text{Tx}_{\text{required}}}(\text{n}, exp))+\text{P}_{\text{Rx}}+\text{P}_{\text{CPU}}+\text{P}_{\text{CPU}}}) \tag{5.2}$$

Assuming we set $P_{\text{Tx}_{\text{required}}}(1, exp) = 22\,\text{dBm}$, $P_{\text{Rx}} = 18.15\,\text{mW}$ and $P_{\text{CPU}} = 3.3\,\text{mW}$ we get the total power consumption as seen in fig. 5.4. For reasons of clarification, fig. 5.5 shows an example calculation for pure Tx power consumption with hop count 1 & 2.

Even if we introduce a power margin, the values do not get shifted, as the perceived Tx powers have to be kept proportionally the same for all hop counts compared to the near-sensitivity case without a margin.

The conclusion is that by introducing lower power levels in non-overpowered star-topologies we can only gain in power efficiency in very near and highly lossy constellations. The

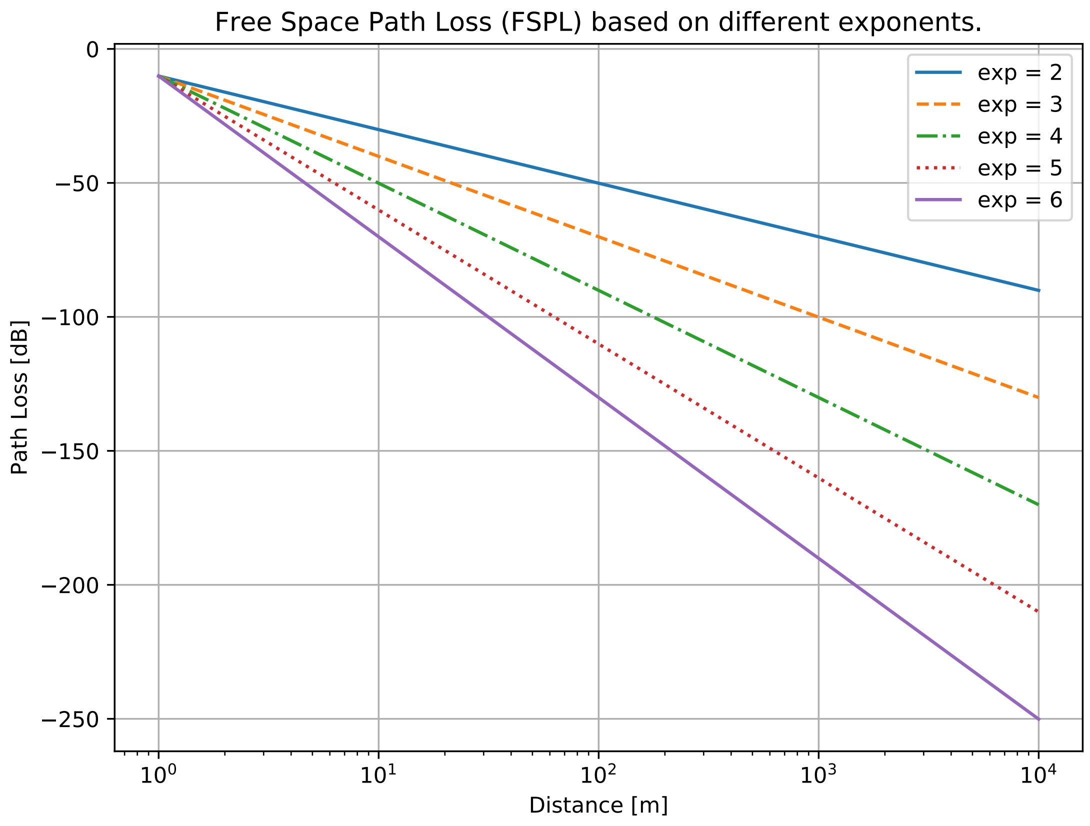


Figure 5.3: Evolution of path-loss depending on FSPL exponent and distance.

contradiction you might see for an exponent of 2 to have higher current consumption by introducing multi-hopping is due to the fact that we have much higher possible maximum link distances and the power reduction is less dramatic when adding a hop (as the hop does not circumvent fading and obstacles as it can be interpreted with exponent 5).

The knowledge we gain is the fact, that for very short link distances and high-powers at the fastest transmission mode, we can gain some power efficiency by lowering the overall Tx level (as a high FSPL exponent seems to be more probable). By utilizing a time-of-flight aware time synchronization protocol and being able to range link distances, we can identify these scenarios, and benefit from optimization.

Yet for most cases, having less hops at a high (non-overpowered) Tx power level is optimal by a high chance. A downside of less hops is that more free space is consumed by the higher signal levels, resulting in limitations for space-division multiple access (SDMA) based MAC protocols.

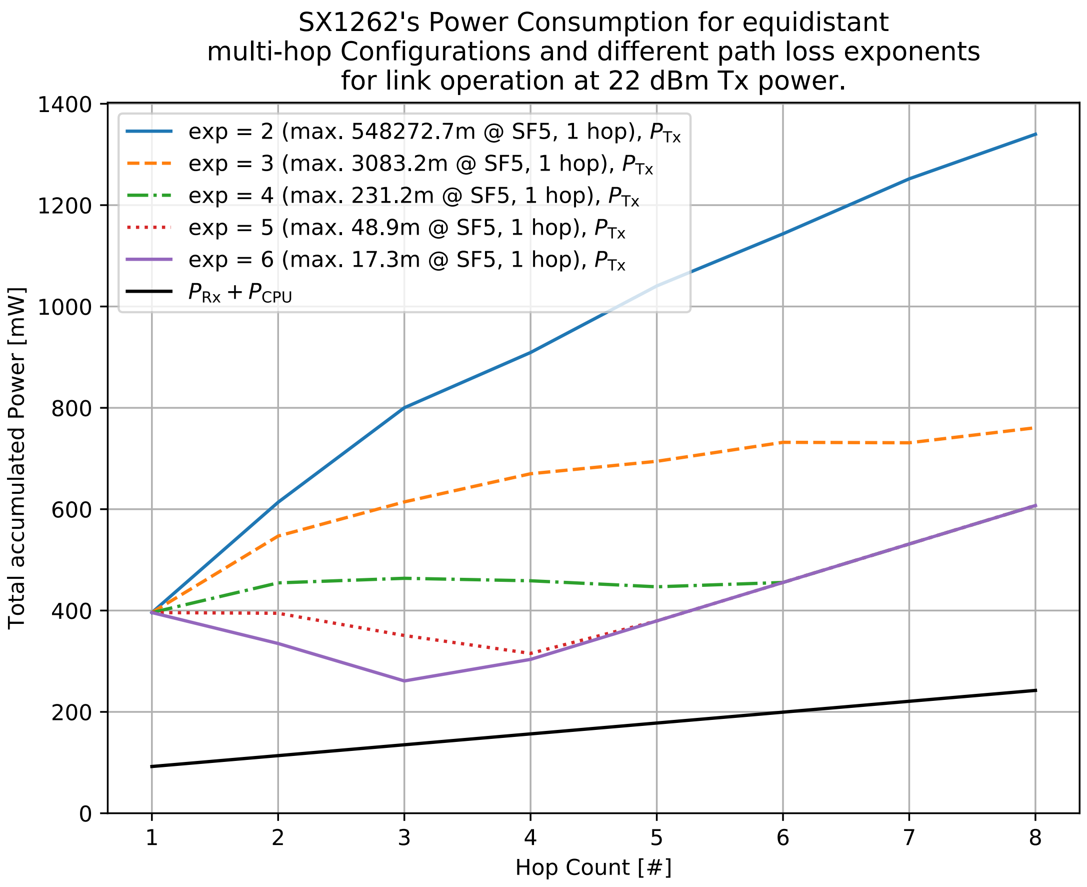


Figure 5.4: Total power consumption for different FSPL exponents and introduced hops. The black line depicts the accumulated linear overhead in Rx and CPU power consumption that comes with more processing time and transmissions.

5.2 Gloria

Gloria is a new message-flooding protocol and mechanism introduced by the author based on experience with Glossy .

Gloria's goal is to enable most different radio configuration modes and scenarios with options for dynamic power levels, reduced power consumption in one-to-one communication and base for precise ($< 2\,\mu s$) time synchronization.

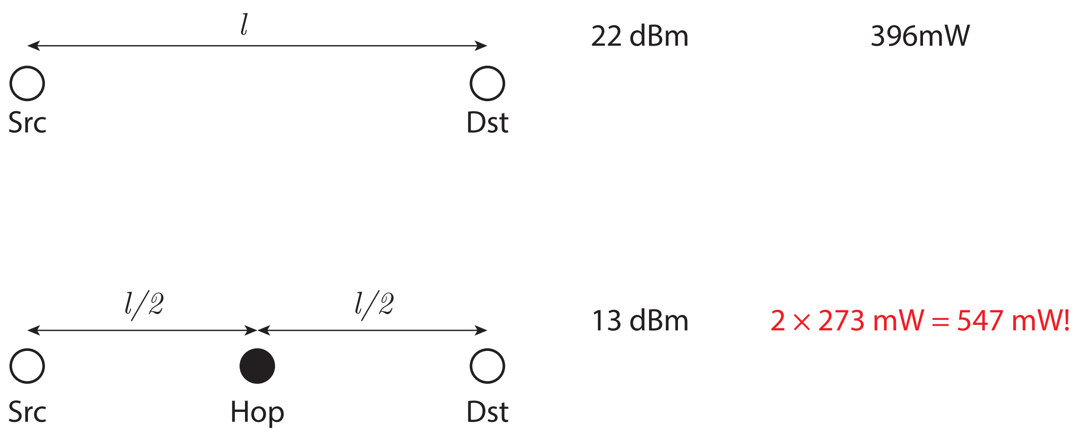


Figure 5.5: Multi-Hop example for non-overpowered link introducing one intermediate node at $exp = 3$

5.2.1 Explicit Header

To allow for dynamic payload lengths in packets, SX1262's packet engines support the addition of a header which defines the messages payload length, CRC settings and coderate (only *LoRa*) on the fly without having the Rx node to be set for the expected payload length. The respective packet layouts are documented in [13, p. 40,44].

The flexibility introduced by the explicit header facilitates easier recovery from erroneous states (e.g. a node being configured with the wrong payload length and having its local time offset). In case of *LoRa* we can additionally profit of the exclusive *Header Valid* IRQ, which allows for a precise and constant synchronization offset regardless of the set payload length. If a header is not received correctly, the radio will stop the reception whilst the transmission is still ongoing, allowing us to spare some energy.

A side effect is the the high overhead and the accompanying power consumption. For *LoRa* transmissions a 20 symbol overhead is prepended, adding a large power consumption bias, which can be critical for WSN's with expected short message lengths (GFSK's overhead is only 1 symbol or byte).

LoRaWAN is also using explicit headers to (among others) limit state between sensor nodes and base to a minimum (i.e. nodes do not have to announce their packet size beforehand).

5.2.2 Slot Layout

Gloria is as Glossy structured into *floods* and *slots*: A slot is defined by its set moment in global time and duration in which a single transmission or reception can occur for a node.

A flood represents the entirety of wireless activity beginning with the initial transmission slot and propagating slot after slot until a slot count threshold or other limitations (retransmission count, maximum hop count) are reached.

For Gloria, the slot layout is defined by the maximum expected time it takes for the SX1262 to allow for all possible required transitions (Rx-to-Tx and Tx-to-Rx)[1] between the transmissions and receptions, a safety margin called *gap*, pre-listening time and the maximum expectet Time-on-Air (therefore allowing dynamic message lengths).

The slot layout's construction can be seen in fig. 5.6. The respective delay figures are calculated for every single mode seperately. To understand all formulas and safety margins see the *flora-tools*'s `flora_tools/gloria.py` file. An added safety factor of 2 times the standard deviation per delay figure and a high gap time of 800 µs ensure robust and time-critical operation.

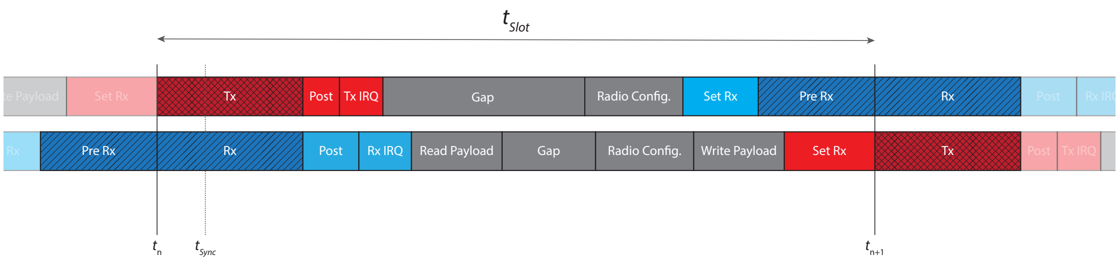


Figure 5.6: Gloria slot layout (sizes not to scale): The slot begins with the execution of the transmission, the moment of t_{ex} on the sender. The receiver nodes have to be in pre-listening (*Pre Rx*) for a defined symbol length. The actual synchronization pulse t_{Sync} is constantly offset and approximately defined by the end of the header (*LoRa*) or syncword (*GFSK*). For all examined configuration, pre-listening time had the bigger impact than reading and setting of the payload. The layout is therefore defined by the overhead of the Tx-to-Rx transition. Post proccessing (*Post*) represents the time required for the radio to finish the Rx or Tx operation.

The figures for the different delays and offsets have been extracted from the measured data and are processed in *flora-tools* to generate lookup tables for the different transmission modes and payload lengths. This allows to deploy changes to the firmware in

[1]additionally Rx-to-Rx transition is relevant for Gloria ACK

faster time compared to having to do manual adaptions. Some example values are listed in table 5.1.

Gloria Modulation Index	Payload [byte]					
	8	16	32	64	128	255
3 (SF9)	0.19924	0.2402	0.32212	0.46548	0.7522	1.32564
5 (SF7)	0.0573488	0.0727088	0.0931888	0.139269	0.236549	0.420869
7 (SF5)	0.0193914	0.0232314	0.0309114	0.0475514	0.0808314	0.146111
9 (FSK 200k)	0.00305338	0.00337338	0.00401338	0.00529338	0.00785338	0.0129334

Table 5.1: Example Gloria slot times in seconds, as statically generated by *flora-tools*.

Setting the beginning of the slot with the actual transmission has simplified implementation. One can argue that we could also benefit of defining the slot's beginning as the theoretical end of the syncword transmission. This would allow us to decouple different preamble lengths and time synchronization information.

To minimize Rx power consumption, the radio is configured in such way to only listen for the duration of the defined preamble length plus the pre-listing offset (see `StopTimerOnPreamble`, [13, p. 61]). If a preamble is detected, it will listen further until in receives the header correctly, or is stopped by a timeout from the MCU's TIM2 shortly after.

5.2.3 Flood Layout

The flood timings are defined in such way to let the MCU be able to wakeup and start pre-listening before the actual begin of the first Gloria flood. After the maximum count of Gloria slots and the flood being considered finished, there is time reserved for the MCU and radio to transition into sleep mode.

Time information is always ensured to reference the current flood's slot marker. To limit the timestamp payload transported in broadcast synchronization messages, floods are only allowed to be initiated at exactly 128 µs time steps. Six bytes of timestamp information then have a range of

$$\begin{aligned} t_{Range} &= 128\,\mu\text{s} \cdot 2^{6 \cdot 8} && (5.3) \\ &\approx 1142.5 \text{ years} && (5.4) \end{aligned}$$

still with the theoretical accuracy of 125 ns. The large timestamp range mitigates wraparound problems already at very low level, easining acquisition, montioring and debugging.

5.2.4 Enhancing the Flood

Due to the increase in Time-on-Air regarding *LoRa* and its high spreading factors, wasteful retransmissions should be limited to a minimum.

The author introduces several enhancements for the reduction of power consumption (among others). We show extensions on top of the default Gloria flood layout as seen in fig. 5.7.

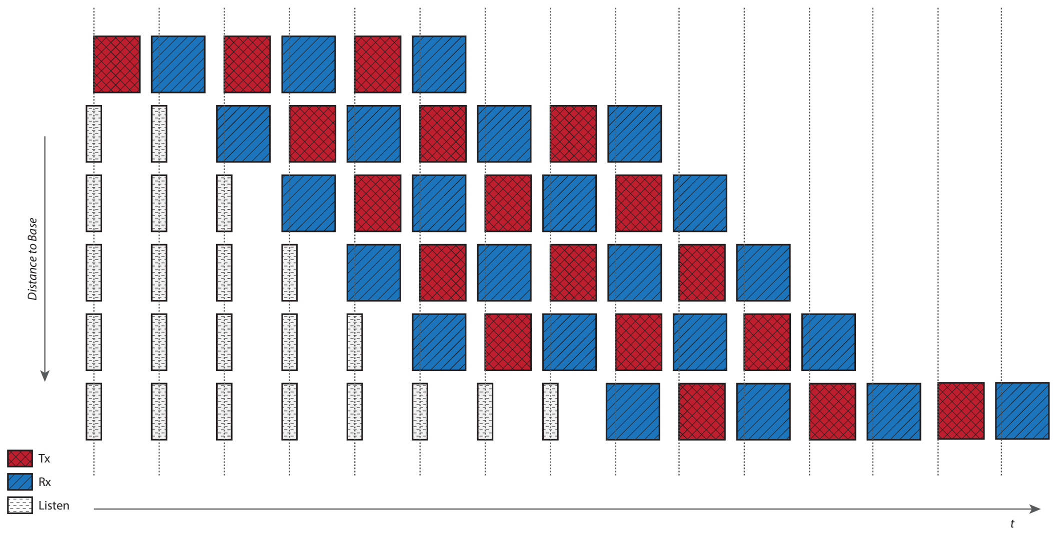


Figure 5.7: A Gloria flood without modifications (sizes not to scale).

Gloria Rx Window Removal

A first modification in Gloria is the removal of unnecessary Rx windows as shown in fig. 5.8. The retransmission of incorrect messages is highly improbable due to a 8-bit network identification number and 16-bit CRC checksum: If they mismatch, the node will not retransmit but relisten for the message. We see a small penalty on the Gloria slot times as we now have to consider the case of a Rx-to-Rx transition as seen in fig. 5.9.

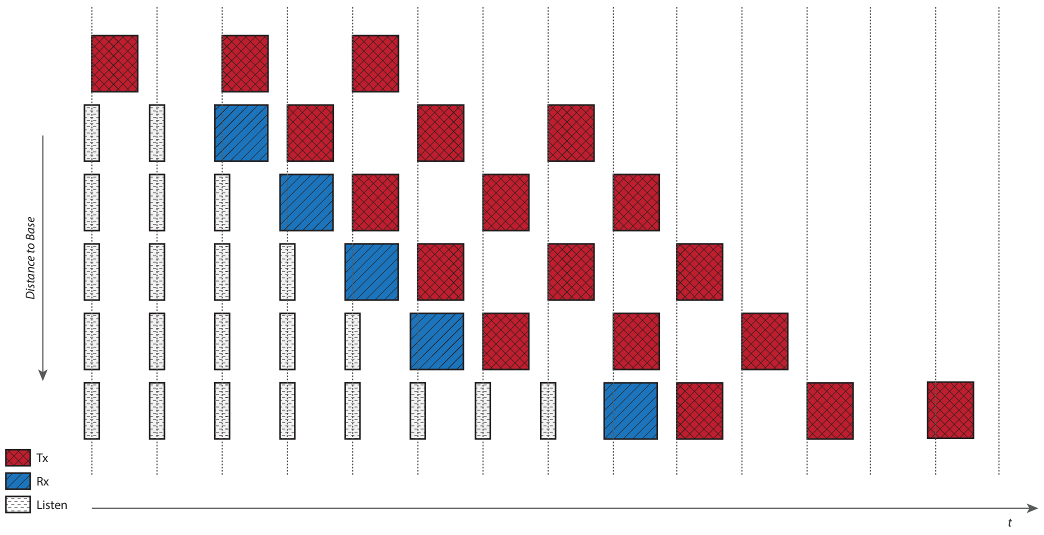


Figure 5.8: A Gloria flood without intermediary Rx slots (sizes not to scale).

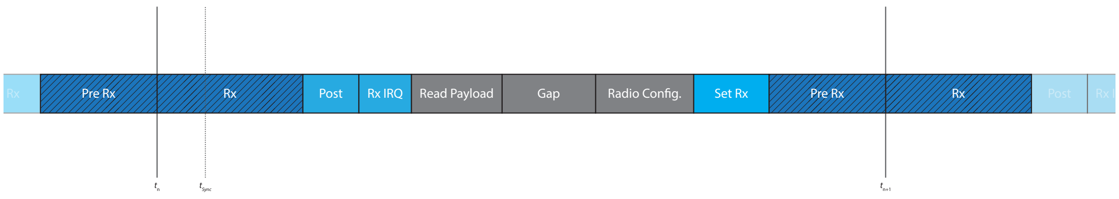


Figure 5.9: For Gloria ACK and Rx window reduction, slot time is defined by a Rx-to-Rx transition.

Gloria Power Level Increase

A further improvement can be achieved by increasing the power level each retransmission (depending on the Gloria modulation index) as seen in fig. 5.10. By this mechanism we can get more information on a node's required power level for communication with the base, eventually being able to reduce overall power for data transfers. A 2-bit field in the Gloria Header is used to communicate the applied Tx power. Additionally, we reduce the overall maximum flood slot count: As the author assumes that higher powered transmissions can pass the network in a single closed chain, we set the floods slot count limit to

$$N_{slot} = (2 \cdot \text{Retransmission Count} - 1) + (\text{Max. Hop Count} - 1) \tag{5.5}$$

rather than

$$N_{slot} = (2 \cdot \text{Retransmission Count} - 1) \cdot (\text{Max. Hop Count} - 1) \tag{5.6}$$

As seen before, high Tx power can be (under the circumstances of long-range communication without obstacles) more efficient than lower Tx power levels. We also gain in faster floods, which enable us to apply higher throughput and range-extension (see table E.0, FSK 200k, in comparision to Glossy with FSK 250k, but higher flood times.)

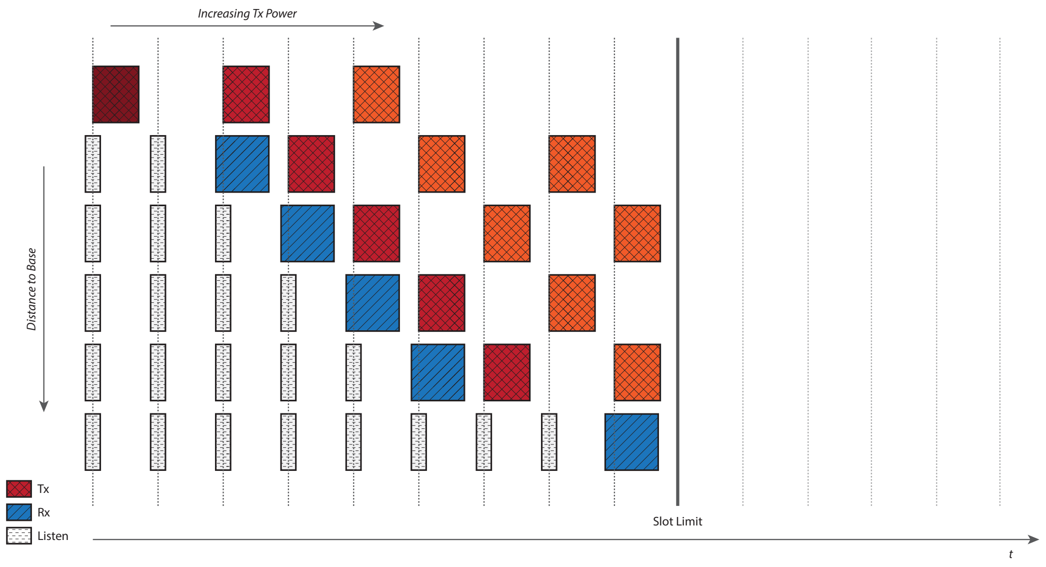


Figure 5.10: A Gloria flood with increasing power levels of the retransmissions.

Gloria ACK

As a last extension the author designed a mechanism which can interrupt a flood in a one-to-one communication scenario (e.g. data transfer between sensor node and base), by adding addressing and an interleaved *ACK* slot in between every normal Gloria slot. fig. 5.11 explains the basic principle. Do not confuse the ACK message to be required to be received by the initiating node, as it is purely optional and can get lost.

Implementation requires an additional *ACK required* field and node address field (which is replacing the timestamp field, as Gloria ACK flood are not considered to be transporting time information). In addition we have to increase overall Gloria slot times as seen in fig. 5.9.

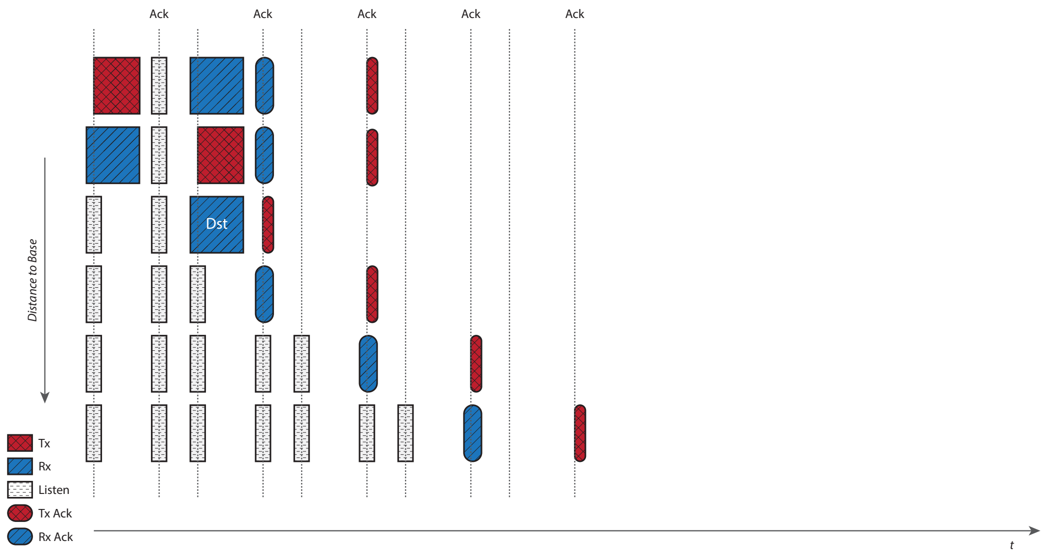


Figure 5.11: A Gloria flood with interleaved *ACK* slots. As soon as the designated destination node receives the message it emits an acknowledgement message (ACK) and then goes to sleep. Each node that receives the ACK message waits for one slot time, retransmits the ACK and goes to sleep as well.

Even with all the overhead added, Gloria floods can still reach an average datarate of more than 2 kB/s for the fastest transmission mode.

Have a look at appendix C where the author tries to formulate analytical boundaries on the overhead of Gloria ACK.

5.3 LWB

The author's goal is to adapt the concepts of LWB to incorporate different transmission modes while keeping the complexity of distributed state low and guarantee reliable operation.

5.3.1 LWB Rounds and Slots

The low-level part of the new LWB consists *LWB slots* (from now on simply *slot*) and *LWB rounds*. A round is built up by different slots (carrying each a Gloria flood), following a scheme defined by the round's type. Each round[2] begins with a *slot-schedule slot* and ends with a *round-schedule slot*. It has a fixed transmission mode.

The slot-schedule slot transports information for the round's slot assignment: It announces the type of round, how many slots are present, and (in case of a data round) which nodes might initiate a flood with a defined maximum payload length (therefore setting the slot's total time) and at which power level they may do so. If a node receives the slot-schedule it can follow the round.

The round-schedule slot transports information about the time of every next planned round for each transmission mode. If a node receives the round-schedule, it can listen for new slot-schedules at the given moment and transmission mode.

If a node looses track of the round-schedule and has no knowledge left about future rounds it transitions into out-of-sync or CAD state, where it tries to scan for messages on the different transmission modes to resynchronize (further explanation below).

The different round types currently defined are:

- Sync Round: Guaranteed to be sent at least every 268.435 s. Includes following slots:
  - Sync Slot: Message with time information and identifying the round as a sync round.
  - Round Request Slot: Contention based slot (each sensor node can initiate the flood), which indicates to the base to schedule a stream-request round for each transmission mode. Useful when a node joins the network but no stream-request round is being scheduled.

[2]Except for a sync round which does not have a slot schedule, but a smaller sync slot to reduce overhead.

    - Round Schedule

- Stream-Request Round: Allows to request a LWB stream on the base, which the base then will schedule according to the given stream period, priority and expected payload length.
    - Slot-Schedule Slot: Message with time information and identifying the round as a stream-request round.
    - Several[3] pairwise Stream-Request and Ack Slots: The former is contention-based, latter is for the base to acknowledge the stream-request.
    - Round Schedule
- Data Round: Allows to exchange data without contention for defined streams.
    - Slot-Schedule Slot: Message with time information and identifying the round as a data round.
    - Several[4] pairwise Data and Ack Slots: The former is contention-based, latter is for the base to acknowledge the stream data.
    - Round Schedule

5.3.2 Data Acquisition: LWB Streams and Links

Assume now a network topology as shown in fig. 5.12 on the left side. Nodes can be interconnected with higher speed to the base the better their, and their surrounding neighbours' (and so forth) interconnecting channels are (i.e. less path loss, e.g. by near-distance, line-of-sight and/or little interference). We do now only focus on the flood-connectivity from the base node's perspective.

The author's LWB version supports Gloria Modulation Index 3 (SF9), 5 (SF7), 7 (SF5) & 9 (FSK 200k), mapping to *LWB Modulation Index* 0 to 3, for a high variation in speed and long-range capability.

In the beginning stage, all nodes are considered to be out-of-sync. They are scanning predefined channels and modes for activity. If the fail to find valid LWB activity in the LWB sync period, they will backoff the scanning process, and stay in low-power mode. After the backoff time they rescan the channels. If they fail again, the backoff is growing exponentially (up to $\approx 6\,\mathrm{h}$).

Each LWB sync period, it is guaranteed that the base will have at least sent one synchronization flood (including time information among others) on the slowest, most long-range

[3]Defined by stream-request count and maximum slot count by base.

[4]Defined by slot-schedule and maximum slot count by base.

transmission mode, SF9 (Gloria Modulation Index 3). Having detected the activity via the *LoRa* CAD feature, the nodes will receive this message. As Gloria Modulation Index 3 is defined to only allow for one hop and one transmission, the nodes will not retransmit the message, controbuting to the flood. If they had received a message at Gloria Modulation Index 5 or higher they would have continued the flood natively (even if there was no time information, e.g. in the LWB data slot). Other messages which resynchronize the nodes are slot-schedule and round-schedule message.

The nodes now follow the rounds (if they happen to receive respective slot-schedule), contributing in the floods as plain relays. The local LWB link tables are updated for every message that is received correctly, storing the power level and transmission mode for the message's initiator node.

If they have a LWB stream to register they will transmit a round-request in the Sync round and/or try to successfully make a stream-request in the best possible stream-request round (if the LWB priority is lower or equal to the transmission mode's LWB Modulation Index.). The stream-request includes information regarding the link-power to the base node, so the base-node knows at which power level to send a LWB Ack for the stream-request and data slots.

If the stream-request fails, consecutive stream-requests are randomly put in backoff (number of stream-request slots to wait is 4, 8 & 16). If even then the stream-request is not acknowledged, the node tries the next slower long-range transmission mode (if priority allows it). As the stream-request keeps unacknowledged the node can disable its stream-requests for a certain duration completely.

Every time a stream gets successfully registered at the base, it can be scheduled as soon as the stream's period is due and the base can be expecting new data. If no data is arriving at the base, respectively no ack is arriving at the sensor node, the registered stream can get reset (including the option to move the stream-request to a slower transmission mode) after several misattempts.

Every single successful stream-request increases the stream-request slot count for the given transmission mode (which gets decreased every stream-request slot without registration). The stream-request slot count defines how many stream-request slots get scheduled in a stream-request round, and whether even a stream-reqeust round on its own has to be scheduled. This allows to scale up the stream-request rounds with increasing number of new unregistered streams.

As data arrives at its designated node, the service which listens onto the respective stream ID is being notified. A service is the interface between LWB and the higher-level application (e.g. a service can represent a *geophone* or *network_ control* stream).

Whenever an expected message is not arriving, the link manager decreases the respective link's counter until it gets downgraded.

The base recalculates its round-schedule everytime a round-schedule has to be sent. As only nodes which are connectected with a faster or equally fast transmission mode as the round-schedule slot's round is set to, only rescheduling of rounds with faster or equally fast transmission modes is allowed. Slower, more long-range rounds keep their position in time, as more weakly connected nodes would not receive the would. The base has therefore to construct the round schedule every round *a priori*, without the option to resize the layout midst a round.

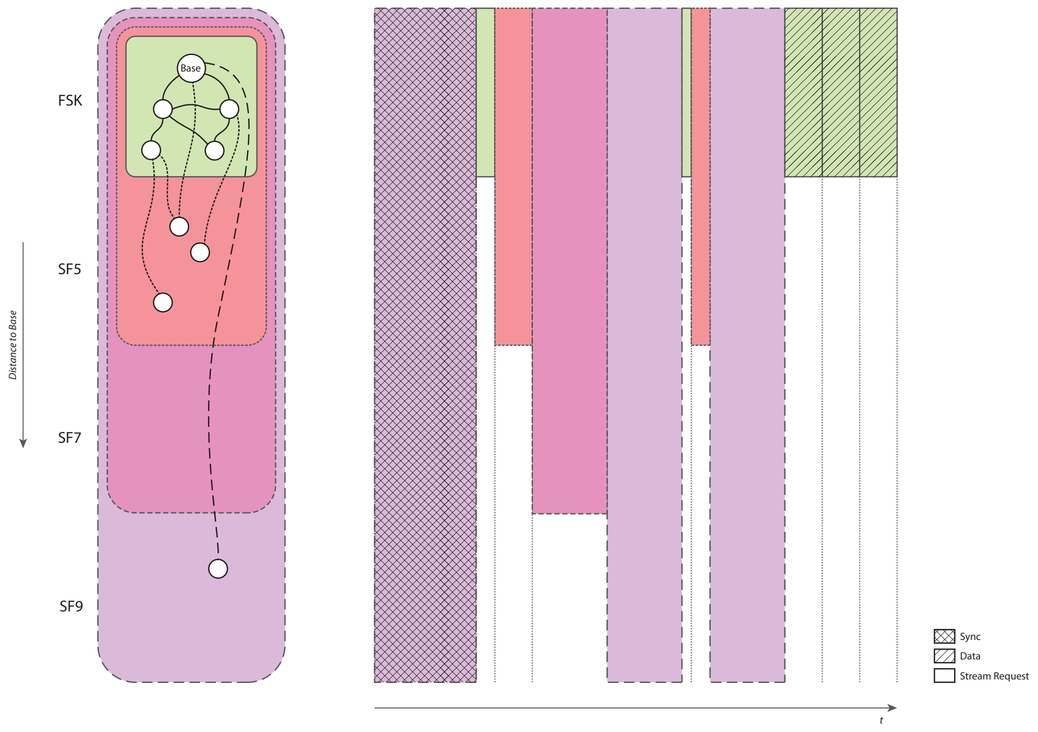


Figure 5.12: The base concept of the author's LWB version: Rounds are scheduled globally with different transmission modes. Each node keeps track of the best transmission mode available. In this example all streams have low priority and only the best interconnected group of nodes are scheduled in data-rounds (right) after their successful stream-request (mid).

5.4 Simulation Model

To study different protocol approaches regarding LWB/Gloria, the author wrote his own event-driven protocol simulation engine for wireless networks based on Python 3.7, Pandas. To study the simulation traces, a new trace visualization tool based on ECMAscript 7 and SVG has been built.

The simulation is event-driven, similar to other protocol simulation tools and hardware-description language simulators. A rough overview of the structure is given in fig. 5.13:

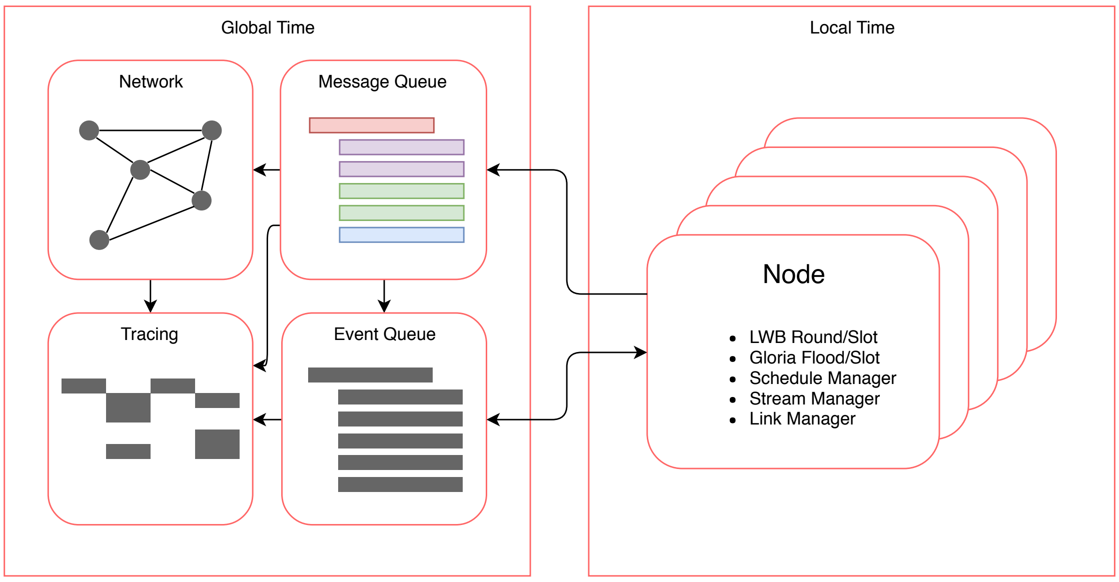


Figure 5.13: The structure of the *flora-tools* wireless-protocol simulation engine split in two time domains: Each node can register events on the event queue which only affects the node itself (e.g. scheduling the reception, timeout or CAD scanning events.). Wireless communication is simulated via the message queue and network component. The arrows indicate which component is invoking which other component.

Basically, a list of events is processed iteratively in ascending order of time. Every event is assigned to a procedure to run in context of a defined node. The procedures replicate operations of LWB & Gloria, e.g. setting the node into Rx mode and register a new event for the Rx timeout, at which point it is decided, whether a message is available. The events only ever concern one single node at a single global time (not excluding the possibility of the node not knowing about global time).

In the message queue, all wireless transmissions get registered with start and end time marker. Each time a *Rx Timeout* event is called on a node, the network component is used to calculate whether a message is available and if it falls in the correct range of the Rx window. If multiple messages are available the first one is being considered.

A *Rx Done* event is registered, who's procedure determines whether a message was actually received correctly. Messages in between can be seen as interference if they have the same transmission mode, and different message content (defined by a hash) or high enough time offset (e.g. some µs). The perceived power of the message and interferences are compared at the Rx node including path-loss calculation. If the message's power overweights the accumulated interference power to surmount SNR and sensitivity limit (actual values regarding SX1262), the message is considered received.

A special case can happen when the Rx Timeout window is larger than the transmitted message: The node which processes the Tx Done event asks the message queue to register a special type of Rx Done event (namely *Rx Done Before Timeout*) for all nodes that are listening, so they can receive the message before their Rx Timeout event is called. This extension is important for the resynchronization process.

A node's local time is always coupled by an offset to the global time (of which the node does not know). Using the offset, a node registers an event in global time, which ensures that all events are processed in causal order.

All important procedures get traced and saved in a JSON-formatted file (including slot positions, energies from Tx and Rx operation and meta-information about the network). A ECMAscript 7 (JavaScript) based web application then allows to visualize and browse the trace interactively by mouse. A large benefit over the tools in Cooja (a supplemental Contiki toolchain and simulation environment) is the platform independency, fluent navigation and zooming. Screenshots can be seen in fig. 5.14 and fig. 5.15.

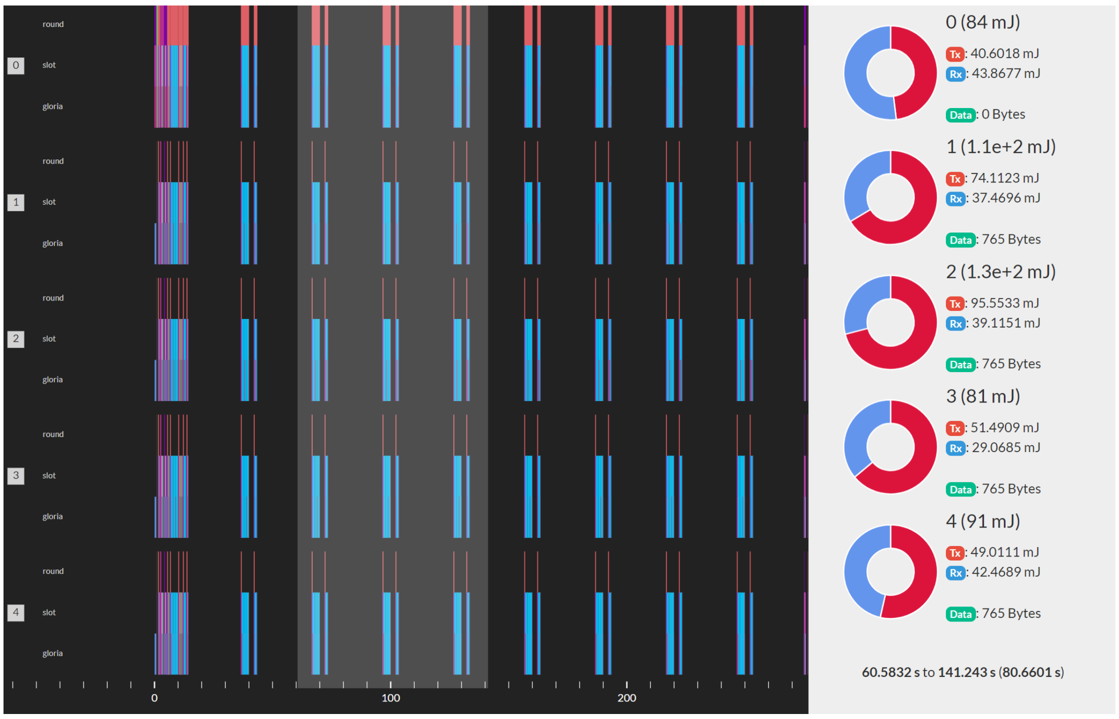


Figure 5.14: Screenshot of the trace visualizer: Five tracks represent the five nodes in the network, with the x-axis being the time in seconds. The tracks are separated into three zones for LWB rounds, slots, and Gloria communication. By selecting a time range with the pressed right mouse button (semi-transparent white region.), we get statistics on the power consumption and throughput for each node (right side).

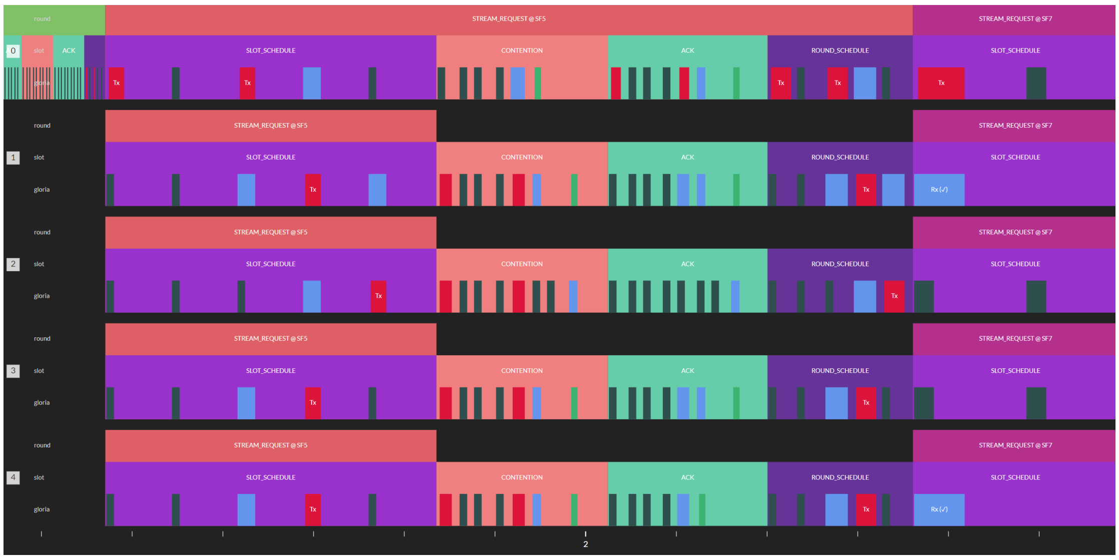


Figure 5.15: Screenshot of the trace visualizer: We see a zoomed in version of a stream-request round with transmission mode SF5 (first subtrack), and only one stream-request slot (second subtrack). The dark boxes represent non-successful receptions, blue successful ones. Green slots mark the transmission of a Gloria ACK (third subtrack).

5.5 Hardware Implementation

The hardware implementation of LWB/Gloria resembles strongly the event-driven pattern of the simulation. This allows for minimal overhead compared to loop-based protocol implementations, and lets other lower-priority tasks run concurrently, respectively puts the MCU into sleep mode if the system is idle (i.e. waiting for external activiation).

The protocol is structured into more files than LWB/Glossy, allowing better understanding and tracing of erroneous behaviour. The chain of events follows a nested path through the different protocol levels of their respective source code file (e.g. `lwb_round.c` $\rightarrow$ `lwb_round.c` $\rightarrow$ `gloria.c`). The lowest level `gloria_radio.c` then interfaces with the *flora* radio driver.

Most functions are designed to only take one struct pointer as a parameter, ensuring tidy calling conventions.

All floating-point operations have been replaced with lookup tables generated by *flora-tools*, to accelerate LWB's and Gloria's slot time and offset calculations.

5.5.1 Gloria

Gloria can be fully controlled via the CLI (i.e. initiating, listening, and timing of floods), so one is able to isolate Gloria completely from the complexity of LWB.

The built-in time synchronization adjusts directly TIM2's counter offset, so all subsystems can directly benefit from the new time information (e.g. the green system LEDs of the nodes are flashing in sync). For floods without time information, only the flood's slot times get adjusted.

The radio is being put into the best available sleep mode as soon as possible, i.e. the time left for wakeup and reconfiguration of the radio at the next occasion is at least double the given wakeup time.

Verbose output on a flood's status after execution helps to understand and analyze complex scenarios on the FlockLab.

Gloria Header

The Gloria header as defined in `protocol/gloria/gloria_structures.h` including time information for synchronization can be seen in fig. 5.16, without in fig. 5.17.

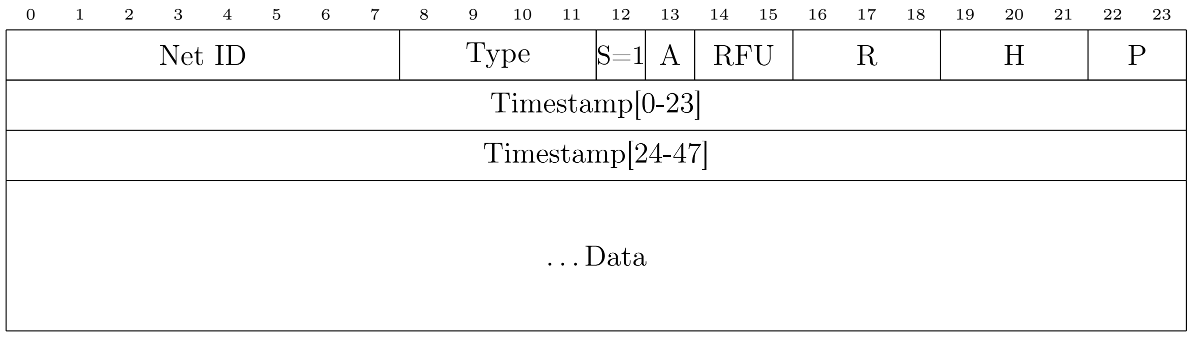


Figure 5.16: Gloria header with synchronization enabled.

`S`: Sync, `A`: Gloria Ack enabled, `RFU`: *Reserved for Future Use*, `R`: n'th Retransmission, `H`: n'th Hop, `P`: n'th Power Level

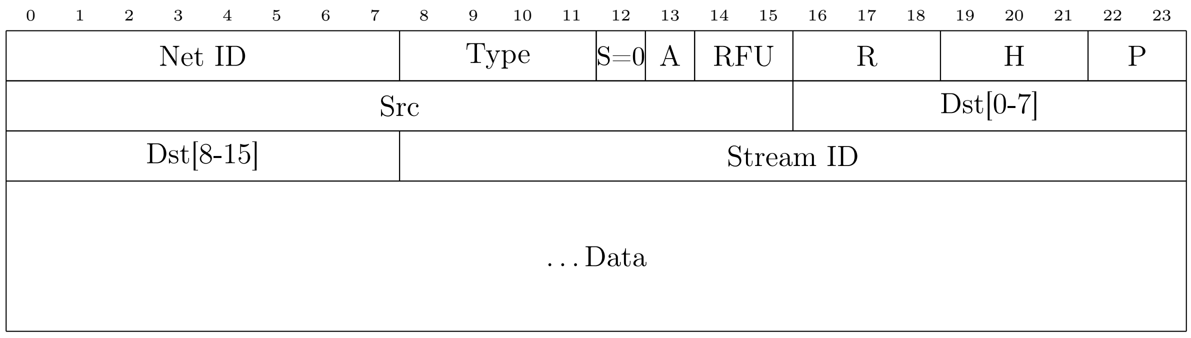


Figure 5.17: Gloria header without synchronization enabled.

`S`: Sync, `A`: Gloria Ack enabled, `RFU`: *Reserved for Future Use*, `R`: n'th Retransmission, `H`: n'th Hop, `P`: n'th Power Level

5.5.2 LWB

The implementation is, as of this writing, very minimal and hardly tested (as testability requires nearly the full implementation for a high-level protocol). Yet subsystems such as the CAD for resynchronization, have been shown to functionally work, including the low-power backoff.

The schedule manager (the unit which calculates the round- and slot-schedule), link manager (the unit which tracks the link table and counter) and stream manager (the unit which manages of streams) are also all implemented with minimal feature set.

Chapter 6

Results

This chapter provides an overview of the different tests the author has undertaken as well as their results.

For the timing measurements see section 4.5.1 and appendix F.

6.1 Evaluation of Transmission Modes

The author has written a script which iterates over every *DPP2 LoRa* node on the FlockLab and transmits a $\approx 55\,\mathrm{B}$ message for every LWB transmission mode and the three power-levels (0 dBm, 10 dBm, & 22 dBm), with CRC and explicit header enabled. The other nodes are put into Rx mode and log the reception. The source code is available in *flora-tools* `analysis/analyze_links.ipynb` and `flocklab/measure_links.py`. The script can automatically schedule a test on the FlockLab and time the serial commands accordingly.

By comparing timestamps and Rx content, the author was able to reconstruct the pairwise receptions. An interactive plot in Jupyter notebook then allows to see the connectivity map for a given configuration (transmission mode, preamble length, power level). The user can also highlight a given node and the respective RSSI levels.

An excerpt of the analyzed receptions is shown in fig. 6.1 and fig. 6.2.

The results indicate that high-range transmission modes can or even should be configured as single-transmission single-hop Gloria floods. This moves the power consumption to the base nodes, which have a much higher power budget at their disposal.

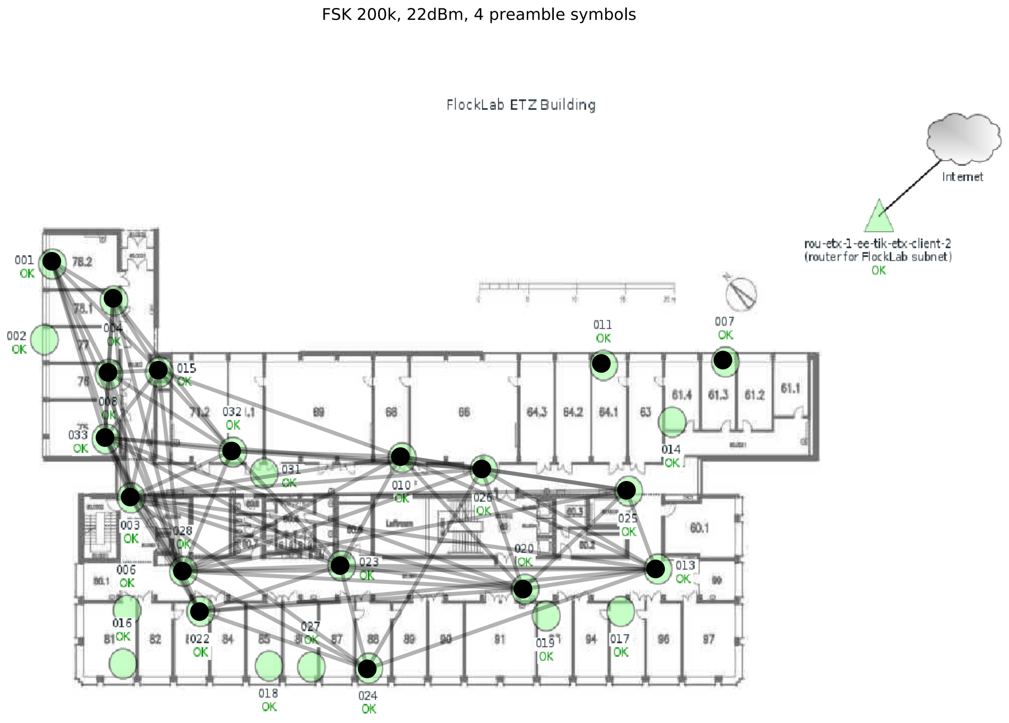


Figure 6.1: Available links on FlockLab for a single run for FSK 200k. Only a very short preamble lenght of four symbols is required. Node 7 and 11 were not enabled yet during the test.

6.2 Evaluation of Gloria Floods

Similar to section 6.1, the author designed a test to check reliability of Gloria Floods on the FlockLab. The source code is available in *flora-tools* `analysis/analyze_flocklab_gloria.ipynb` and `flocklab/measure_gloria.py`.

Iterating over each node, a synchronization flood is being sent on SF9. The other nodes are awaiting the synchronization flood via an infinite Rx window. After the nodes have been synced, a second flood for a given LWB Modulation Index is scheduled in the future on all nodes (≈ 0.3 s), with the selected node being the initial node.

As all nodes have been synced (at least with high probability), they sleep until the begin of the second flood.

The test can be repeated over 40 times in a single hour.

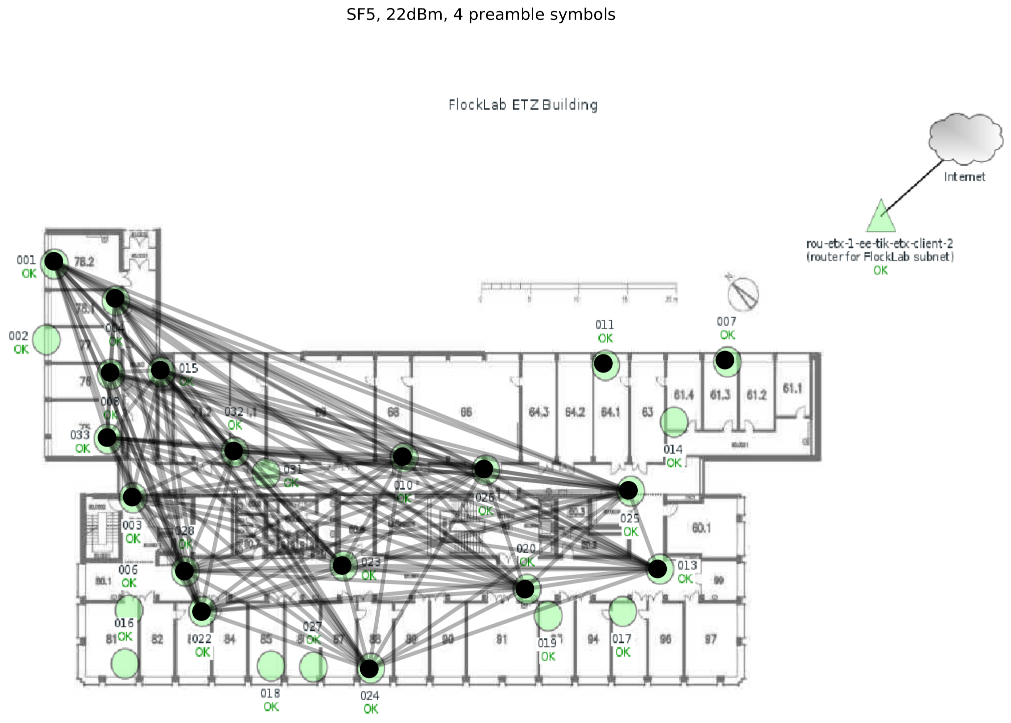


Figure 6.2: Available links on FlockLab for a single run for SF5. Note that every node is reached by node 23 even for SF5.

Analyzing the serial logs, the author's script reconstructs the flood-based links and accompanying metrics. An interactive map similar to fig. 6.1 shows the links for a given transmission mode and hop count.

The script can also run Gloria ACK based floods: For every scheduled flood, the message's destination is randomly selected from the available Rx nodes. By analyzing the remaining retransmissions (as the flood can be interrupted before a given node has conducted all retransmission), we can give a number on the reduction in transmission activity which allows a conclusion on the reduction of power consumption.

As of this writing, bit flips in the *flora* UART stack introduced errors in the serial log, limiting the validity of the current values. Observation of the channel activity by an Ettus B210 SDR (software-defined radio) yet indicate Gloria and Gloria ACK being working correctly.

6.3 Time synchronization

Using a table-top setup similar to the timing measurements in section 4.5.1, the author conducted measurements regarding the synchronization precision. By applying different schedule offsets (10 s & 20 s) and different combinations of synchronization node and 2-nd flood initiator node, he was able to isolate the clock offset between the development kits ($\approx$ 2 ppm) and the actual synchronization error ($\approx$ 1 µs).

6.4 Life-time

By analyzing the theoretical power consumption of a Gloria flood with maximum count of retransmissions and maximum power level (3, respectively 22 dBm) and with the exclusion of the MCU's power consumption, the author estimates that a node with a battery of 12 A h @ 3.7 V allows for roughly some hundred MB up to several GB of accumulated network traffic (see appendix B).

Chapter 7

Conclusion

It has been shown how a new communication board featuring the Semtech SX1262 radio chip with *LoRa* modulation technique can be incorporated into a new MAC layer protocol called LWB/Gloria for WSNs.

Foremost, *LoRa* keeps its promise of highly flexible long-range wireless communication. The thorough measurement and inclusion of the timing characteristics, an event-driven firmware design, as well as the new simulation and toolchain platform *flora-tools* provide the ground for precise time synchronization and concurrent transmissions. Yet, indication for constructive interference could not be shown by the author, and the transmission-offset sweep measurements rather indicate a very strong influence by the *capture effect*, which is similar promising for concurrent-transmission protocols.

The introduction of the *flora* firmware stack for both communication board and development kit facilitated protocol design by far (especially during the lead times involved with the tapeout and production of the communication boards).

The author has shown that for long-range transmission modes, high Tx power levels can be considered most efficient regarding reliability and power consumption.

Care has to be taken regarding the very long Time-on-Air for high spreading factors, due to reduced data-throughput and the higher required energy-per-bit. Our protocol can cope with this fact, by promoting high datarate transmission modes and the reduction of wasteful retransmission due to new flooding-mechanisms such as Gloria ACK and dynamic power level to limit overpowering.

The thesis also underlines the importance of automated test setups for fast, reliable and reproducible measurements, which allow fast adaptions of variables (such as the preamble length), as well as comprehensive interface and logging mechanisms for both user- and machine-interfacing.

Chapter 8

Future Work

The evaluation of the LWB part of LWB/Gloria and its extension has high relevance for the future. Due to the complexity and unforseen scenarios, meaningful measurements on the FlockLab, as well in the field, examination and the understanding of the holistic effects and overall performance is very important.

The author's LWB might benefit of extensions allowing for short notification rounds similar to eLWB, extensions for near real-time interactive communication with one or a group of nodes (e.g. running the CLI over *LoRa*), a network configuration layer, fallback mechanisms to reliable long preamble lengths, watchdog integration, elaborate stream control for upgrading or downgrading, improved stream and link lookup in the stream- and link-manager, or a central control featuring an intuitive interface for management of the nodes (including remote firmware upgrades). Other aspects involve the development of a secure communication layer, or the delegation of the base's round announcements to another node, allowing for a hybrid approach of LWB and Dozer style communication.

The simulation and trace visualization can be further extended to incorporate a model of FlockLab's network topology and channels, as well as the visualization of the FlockLab logs via *flora-tools*. Dynamic manipulation of the simulation's network model enables complex (automated) testing scenarios.

Gloria can be extended with a frequency-hopping scheme. Additional extensions such as a more fine-grained adaptive power-control (APC), automatic frequency control (AFC), and listen-before-talk (LBT) complying with [40] let us use the new SRD-assigned bands from 870 MHz to 873 MHz in Switzerland as of 2018 according to [41], benefiting of less interference. Implementing the *offset-CT* approach as shown in [20] might have a positive impact on PER.

Concerning the author's timing measurements: A drawback in the experiments' design is the cable & table-top setup which would never allow for signal levels near the sensitivity limits. A sophisticated two-chamber setup and highly calibrated *LoRa* source, as well as precise measurement equipment for longer distances would be required.

The integration of a small-footprint real-time operating system such as FreeRTOS into the *flora* stack should be considered as it might easen the interfacing with *BOLT* due to its thorough IPC tools.

Last but not least, the implementation of the *Standby Mode* can further limit idle power consumption. It requires to design a more thorough state recovery approach (handling call stack, heap, and fast reinitialization of peripherals on wakeup).

Appendix A

flora and *flora-tools*

A major contribution to this thesis is the actual firmware stack called *flora*, and supplementary tools for analysis, simulation, code-generation, and deployment called *flora-tools*.

A.1 *flora*

flora is available at `https://gitlab.ethz.ch/tec/research/dpp/com_boards/dpp2_lora` including basic documentation regarding the installation and setup of the toolchain.

A.1.1 Filetree

```
/
└─ apps ........ High-level application code invoked by overloaded system callbacks.
```

```
├── doc ....... Rudimentary documentation. See also README.md.
├── lib ....... Peripheral and protocol support
│   ├── arch ....... Platform-specific implementation (e.g. for STM32Cube HAL).
│   ├── bolt ....... BOLT driver. Yet to be implemented.
│   ├── cli ....... CLI and UART logic.
│   ├── config ....... Flash-memory based configuration storage driver.
│   ├── dpp ....... DPP specific definitions of messages.
│   ├── led ....... LED driver.
│   ├── protocol ....... Protocol logic.
│   │   ├── gloria ....... Low-level Gloria implementation.
│   │   └── lwb ....... High-level LWB implementation.
│   ├── radio ....... Wrapping driver for radio.
│   │   └── semtech ....... Vendor supplied radio driver.
│   ├── system ....... System routines (i.e. update loop, initialization).
│   ├── test ....... Supportive commands for measurements.
│   └── time ....... RTC- and high-speed timer driver.
├── platform ....... STM32CubeMx generated code for specific targets.
└── tools ....... Auxilliary supporting tools (e.g. flora-tools, GnuRadio setup).
```

A.2 *flora-tools*

flora-tools is (currently) available at `https://github.com/Atokulus/flora_tools` including basic documentation, as well as an installable Python 3 package (`https://pypi.org/project/flora-tools/`).

Appendix B

Gloria Energy Consumption

The author conducted theoretical calculations regarding the energy consumption of Gloria floods based on their transmission mode and average payload. The calculations can reproduced in `flora_tools/analysis/analyze_gloria.ipynb`.

The author assumes the worst case for Gloria (without ACK), in which every node has the full retransmission count for 22 dBm.

The battery is (naively) assumed to have 12 A h at 3.7 V

An overview of the total amount of Bytes being able to be sent, respectively processed by a single node, can be seen in table B.1.

Gloria Modulation Index	Payload [byte]					
	8	16	32	64	128	255
3 (SF9)	3.24e8	5.49e8	8.39e8	1.19e9	1.50e9	1.72e9
5 (SF7)	5.27e7	8.04e7	1.22e8	1.59e8	1.82e8	2.01e8
7 (SF5)	1.52e7	2.47e8	3.59e8	4.51e8	5.17e8	5.60e8
9 (FSK 200k)	1.99e9	3.18e9	4.53e9	5.75e9	6.64e9	7.20e9

Table B.1: The total amount of Bytes being able to be sent, respectively processed by a single node for a given payload length and Gloria Modulation Index. The total amount is smaller for SF7 than SF9 due to the retransmission and hop count being larger (2, rather than 1 respectively), and by incorporating the fact that, we do not account for the first transmission from the base.

Appendix C

Gloria ACK Analysis

In this chapter the author will give short insight into the analytical concepts behind Gloria ACK and its impact on overhead.

C.1 Glossy (without adaptions)

We assume that every node eventually receives the message and does process all retransmissions. Overall power consumption therefor scales linearly with the number of nodes.

$$\begin{aligned} E_{\text{Total}} &= N \cdot r \cdot (E_{\text{Rx}} + E_{\text{Tx}}) - E_{\text{Rx}} && \text{(C.1)} \\ &= N \cdot r \cdot (\frac{N \cdot r - 1}{N \cdot r} E_{\text{Rx}} + E_{\text{Tx}}) && \text{(C.2)} \end{aligned}$$

where r is the retransmission rate and N is the number of nodes. The preamble detection RX windows' power is not accounted for.

C.2 Glossy ACK

Given

$$E_{\text{Rx,Ack}} = \alpha \cdot E_{\text{Rx}} \tag{C.3}$$

and

$$E_{\text{Tx,Ack}} = \alpha \cdot E_{\text{Tx}} \tag{C.4}$$

we are assuming that all packages receive their destined nodes at the first chance. The acknowledgement message is only retransmitted once on a node after its reception.

We consider each node n_i by itself. The crucial term for describing when a node n_i is interrupted by the acknowledgement is given by the difference in arrival time of the source's data message and the destination's acknowledgement (l for *latency*):

$$\begin{aligned} l(n_i) &= \max(0, (d(n_{\text{src}}, n_{\text{dst}}) - 1) + \max(d(n_{\text{dst}}, n_i), 1) - (d(n_{\text{src}}, n_i) - 1)) && \text{(C.5)} \\ &= \max(0, d(n_{\text{src}}, n_{\text{dst}}) + \max(d(n_{\text{dst}}, n_i), 1) - d(n_{\text{src}}, n_i)) && \text{(C.6)} \end{aligned}$$

The upper bound total slot count (regarding the flood's layout) then can be approximated by

$$\begin{aligned} |S_{\text{Rx}_{n_i}}| &= \min(\text{ceil}(\frac{l(n_i)}{2}), r) && \text{(C.7)} \\ |S_{\text{Tx}_{n_i}}| &= \text{floor}(\frac{l(n_i)}{2}) && \text{(C.8)} \\ |S_{\text{Tx,Ack}_{n_i}}| &= H(2 \cdot r - l(n_i) - 1) && \text{(C.9)} \\ |S_{\text{Rx,Ack}_{n_i}}| &= H(2 \cdot r - l(n_i) - 2) && \text{(C.10)} \end{aligned}$$

where $H(x)$ represents the Heaviside function.

We also consider some adaptions for a better approximation: I.e. the source node itself has one Rx slot less than the other nodes and therefore has to be regarded seperately in the summation. If a node has a predecessor which received the acknowledgement directly after a Rx slot, the regarding child-child nodes might save one Rx slot.

$$\begin{aligned} E_{\text{Total}} = \sum_{n_i} (&|S_{\text{Rx}_{n_i}}| \cdot E_{\text{Rx}} \\ &+ |S_{\text{Tx}_{n_i}}| \cdot E_{\text{Tx}} \\ &+ |S_{\text{Rx,Ack}_{n_i}}| \cdot E_{\text{Rx,Ack}} \\ &+ |S_{\text{Tx,Ack}_{n_i}}| \cdot E_{\text{Tx,Ack}}) \\ &- E_{\text{Rx}} \end{aligned} \qquad \text{(C.11)}$$

An example case where total energy is lower than the upper bound can be seen in fig. C.1.

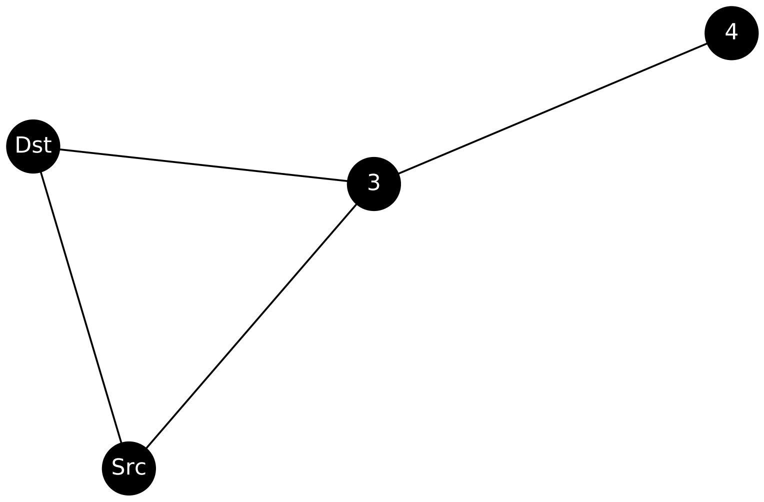


Figure C.1: Special case where the energy is lower than the upper bound for Gloria ACK: As node 3 will get the acknowledgement from the destination node directly after having received the data message from the source node, it will not forward the data message to node 4, just the acknowledgement. This means that we are using one Rx slot's energy less. Other extended yet similar topologies show the same effect.

C.3 Complete Graph Topology (best case)

Assume that every node can reach every other node in just one hop:

$$\begin{aligned} E_{\text{Total}} &= E_{\text{Tx}} + (N-1)\cdot E_{\text{Rx}} + E_{\text{Tx-Ack}} + (N-1)\cdot E_{\text{Rx-Ack}} \qquad (C.12) \\ &= (1+\alpha)\cdot E_{\text{Tx}} + (1+\alpha)\cdot(N-1)\cdot E_{\text{Rx}} \qquad (C.13) \end{aligned}$$

C.4 No reception of ACK (worst case)

The target node (and all other nodes) receive the message, but its acknowledgement does not get received by any other node. All other $N-1$ nodes therefore retransmit the message wastefuly:

$$\begin{aligned} E_{\text{Total}} &= (N-1)\cdot r\cdot(E_{\text{Rx}} + E_{\text{Tx}} + 2\cdot E_{\text{Ack}_{\text{Rx}}}) - (E_{\text{Rx}} + E_{\text{Ack}_{\text{Rx}}}) \\ &\quad + (E_{\text{Rx}} + E_{\text{Ack}_{\text{Tx}}}) \\ &= (N-1)\cdot r\cdot((1+2\alpha-\frac{\alpha}{(N-1)r})\cdot E_{\text{Rx}} + (1+\frac{\alpha}{(N-1)r})\cdot E_{\text{Tx}}) \end{aligned} \qquad (C.14)$$

For high node counts, worst case and untouched Glossy converge, with the acknowledgement messages contributing 2α energy overhead.

Appendix D

Gloria Modulation Settings

Gloria Modulation Index	Modulation Name	Bandwidth [Hz]	Symbol Rate [Hz]	Chirp Rate [Hz]	Coderate	Bitrate [bps]	Symbol Time [s]	Conf. Preamble Length	Preamble Time [s]	Sensitivity [dB]	ToA [s] (255 Bytes)
0	SF12	125000.0	30.517578	3.814697e+06	$\frac{1}{4}$	244.140625	0.032768	10	0.466944	-135.0	9.084928
1	SF11	125000.0	61.035156	7.629395e+06	$\frac{1}{4}$	439.453125	0.016384	10	0.233472	-131.0	5.033984
2	SF10	125000.0	122.070312	1.525879e+07	$\frac{1}{4}$	976.562500	0.008192	10	0.116736	-130.0	2.312192
3	SF9	125000.0	244.140625	3.051758e+07	$\frac{1}{4}$	1757.812500	0.004096	10	0.058368	-127.0	1.258496
4	SF8	125000.0	488.281250	6.103516e+07	$\frac{1}{4}$	3125.000000	0.002048	10	0.029184	-124.0	0.711168
5	SF7	125000.0	976.562500	1.220703e+08	$\frac{1}{4}$	5468.750000	0.001024	10	0.014592	-122.0	0.401664
6	SF6	125000.0	1953.125000	2.441406e+08	$\frac{1}{4}$	9375.000000	0.000512	12	0.009344	-116.0	0.233600
7	SF5	125000.0	3906.250000	4.882812e+08	$\frac{1}{4}$	15625.000000	0.000256	12	0.004672	-113.0	0.139840
8	FSK 125k	234300.0	15625.000000	NaN	NaN	125000.000000	0.000064	2	0.000128	-102.0	0.016832
9	FSK 200k	234300.0	25000.000000	NaN	NaN	200000.000000	0.000040	2	0.000080	-102.0	0.010520

Table D.1: Different radio configurations under focus throughout the thesis. 16-bit CRC and explicit header mode are enabled by default. The color-code is consistent throughout this thesis and *flora-tools*.

Gloria Band Index	Bandwidth [Hz]	Center Frequency [Hz]	Duty Cycle [‰]	Max. Power [dBm]
0	125000.0	863062500.0	1	14
1	125000.0	863187500.0	1	14
2	125000.0	863312500.0	1	14
3	125000.0	863437500.0	1	14
4	125000.0	863562500.0	1	14
5	125000.0	863687500.0	1	14
6	125000.0	863812500.0	1	14
7	125000.0	863937500.0	1	14
8	125000.0	864062500.0	1	14
9	125000.0	864187500.0	1	14
10	125000.0	864312500.0	1	14
11	125000.0	864437500.0	1	14
12	125000.0	864562500.0	1	14
13	125000.0	864687500.0	1	14
14	125000.0	864812500.0	1	14
15	125000.0	864937500.0	1	14
16	125000.0	865062500.0	10	14
17	125000.0	865187500.0	10	14
18	125000.0	865312500.0	10	14
19	125000.0	865437500.0	10	14
20	125000.0	865562500.0	10	14
21	125000.0	865687500.0	10	14
22	125000.0	865812500.0	10	14
23	125000.0	865937500.0	10	14
24	125000.0	866062500.0	10	14
25	125000.0	866187500.0	10	14
26	125000.0	866312500.0	10	14
27	125000.0	866437500.0	10	14
28	125000.0	866562500.0	10	14
29	125000.0	866687500.0	10	14
30	125000.0	866812500.0	10	14
31	125000.0	866937500.0	10	14
32	125000.0	867062500.0	10	14
33	125000.0	867187500.0	10	14
34	125000.0	867312500.0	10	14
35	125000.0	867437500.0	10	14
36	125000.0	867562500.0	10	14
37	125000.0	867687500.0	10	14
38	125000.0	867812500.0	10	14
39	125000.0	867937500.0	10	14
40	125000.0	868062500.0	10	14
41	125000.0	868187500.0	10	14
42	125000.0	868312500.0	10	14
43	125000.0	868437500.0	10	14
44	125000.0	868762500.0	1	14
45	125000.0	868887500.0	1	14
46	125000.0	869012500.0	1	14
47	125000.0	869137500.0	1	14
48	125000.0	869462500.0	100	26
49	125000.0	869587500.0	100	26
50	125000.0	869762500.0	100	7
51	125000.0	869887500.0	100	7

Table D.2: Bands used throughout this thesis.

Appendix E

Sensitivity Listings

*

Index	Bandwidth [Hz]	Bitrate [bps]	SSB FDA [Hz]	Modem	Sensitivity [dB]	SF	Max. Distance [m]	Energy per Bit [mJ] (CRC enabled, explicit header mode, coderate $\frac{1}{4}$)
0	4000	600.000000	800.0	GFSK	-125	NaN	6642.475035	0.66258800
1	20000	1200.000000	5000.0	GFSK	-123	NaN	5697.223594	0.33129400
2	20000	4800.000000	5000.0	GFSK	-118	NaN	3881.473250	0.08282350
3	83000	38400.000000	40000.0	GFSK	-109	NaN	1945.344841	0.01035290
4	50000	38400.000000	20000.0	GFSK	-109	NaN	1945.344841	0.01035290
5	250000	250000.000000	62500.0	GFSK	-96	NaN	717.237956	0.00159021
6	500000	250000.000000	125000.0	GFSK	-104	NaN	1325.348012	0.00159021
7	10400	244.140625	NaN	LoRa	-148	12	38814.732505	1.63971000
8	10400	5468.750000	NaN	LoRa	-135	7	14310.778644	0.07310630
9	125000	244.140625	NaN	LoRa	-137	12	16685.142913	1.63971000
10	125000	439.453125	NaN	LoRa	-133	11	12274.296149	0.91129100
11	125000	976.562500	NaN	LoRa	-132	10	11367.455538	0.41092100
12	125000	1757.812500	NaN	LoRa	-129	9	9029.490891	0.22782300
13	125000	3125.000000	NaN	LoRa	-126	8	7172.379560	0.12807400
14	125000	5468.750000	NaN	LoRa	-124	7	6151.720531	0.07310630
15	125000	9375.000000	NaN	LoRa	-118	6	3881.473250	0.04262820
16	125000	15625.000000	NaN	LoRa	-115	5	3083.163795	0.02556920
17	250000	488.281250	NaN	LoRa	-134	12	13253.480116	0.81985400
18	250000	878.906250	NaN	LoRa	-130	11	9749.819992	0.45564500
19	250000	1953.125000	NaN	LoRa	-128	10	8362.380619	0.20546000
20	250000	3515.625000	NaN	LoRa	-125	9	6642.475035	0.11391100
21	250000	6250.000000	NaN	LoRa	-123	8	5697.223594	0.06403710
22	250000	10937.500000	NaN	LoRa	-121	7	4886.485310	0.03655310
23	250000	18750.000000	NaN	LoRa	-115	6	3083.163795	0.02131410
24	250000	31250.000000	NaN	LoRa	-112	5	2449.044055	0.01278460
25	500000	1171.875000	NaN	LoRa	-129	12	9029.490891	0.34234300
26	500000	2148.437500	NaN	LoRa	-128	11	8362.380619	0.18657700
27	500000	3906.250000	NaN	LoRa	-125	10	6642.475035	0.10273000
28	500000	7031.250000	NaN	LoRa	-122	9	5276.305469	0.05695570
29	500000	12500.000000	NaN	LoRa	-119	8	4191.118409	0.03201850
30	500000	21875.000000	NaN	LoRa	-117	7	3594.705070	0.01827660
31	500000	37500.000000	NaN	LoRa	-111	6	2268.105565	0.01065710
32	500000	62500.000000	NaN	LoRa	-108	5	1801.620290	0.00639229

Table E.0a: Listing of datasheet supplied sensitivity values and 22 dBm Tx operation of SX1262. Path loss is assumed to be cubic for distance limit and energy per real bit data (taking account for the packet overhead). Preamble length is set according to table D.1.

*

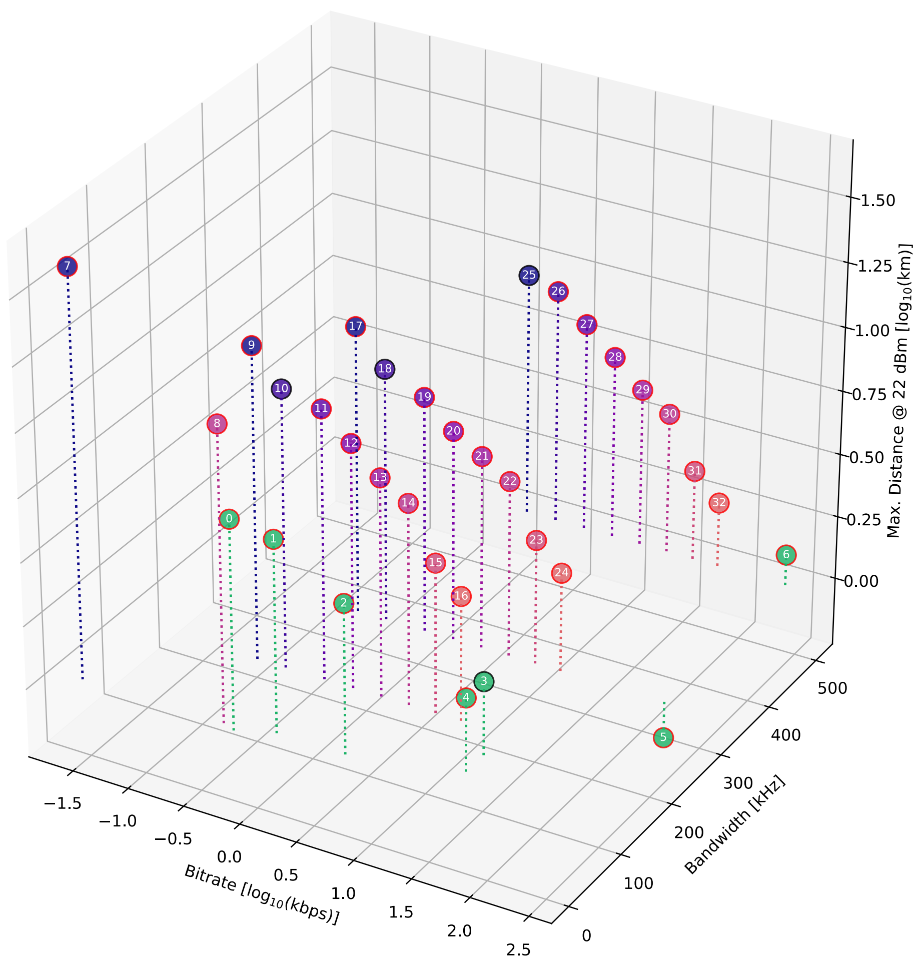


Figure E.0b: Achievable distances assuming cubic path loss and 22 dBm Tx power. Note the exponential decay for increasing spreading factors.

*

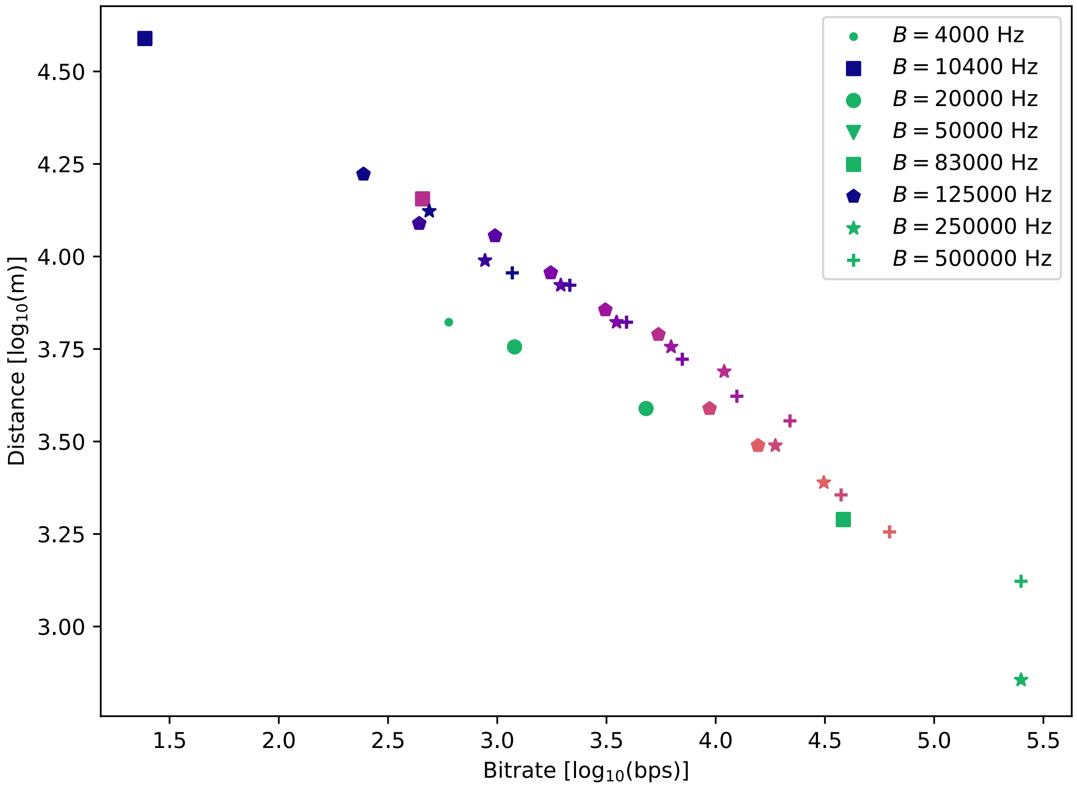


Figure E.0c: Achievable distances assuming cubic path loss and 22 dBm Tx power. Note the exponential decay for increasing spreading factors.

*

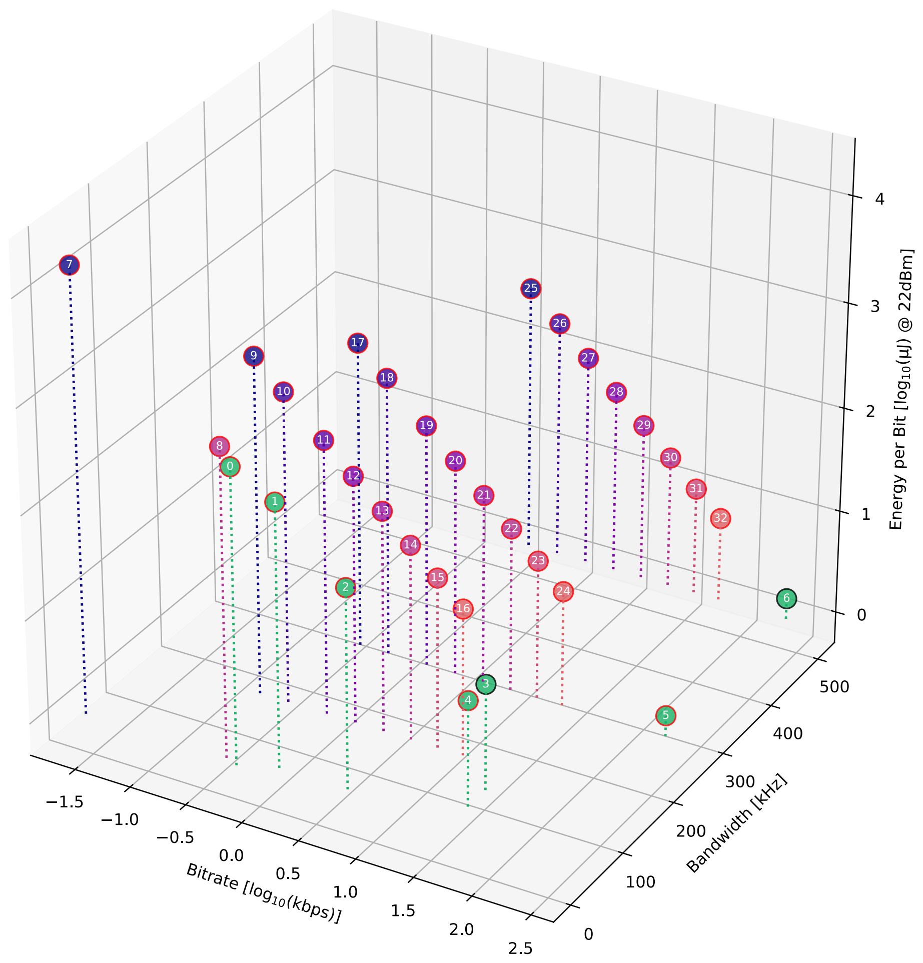


Figure E.0d: Consumed energy per bit assuming 22 dBm Tx power.

*

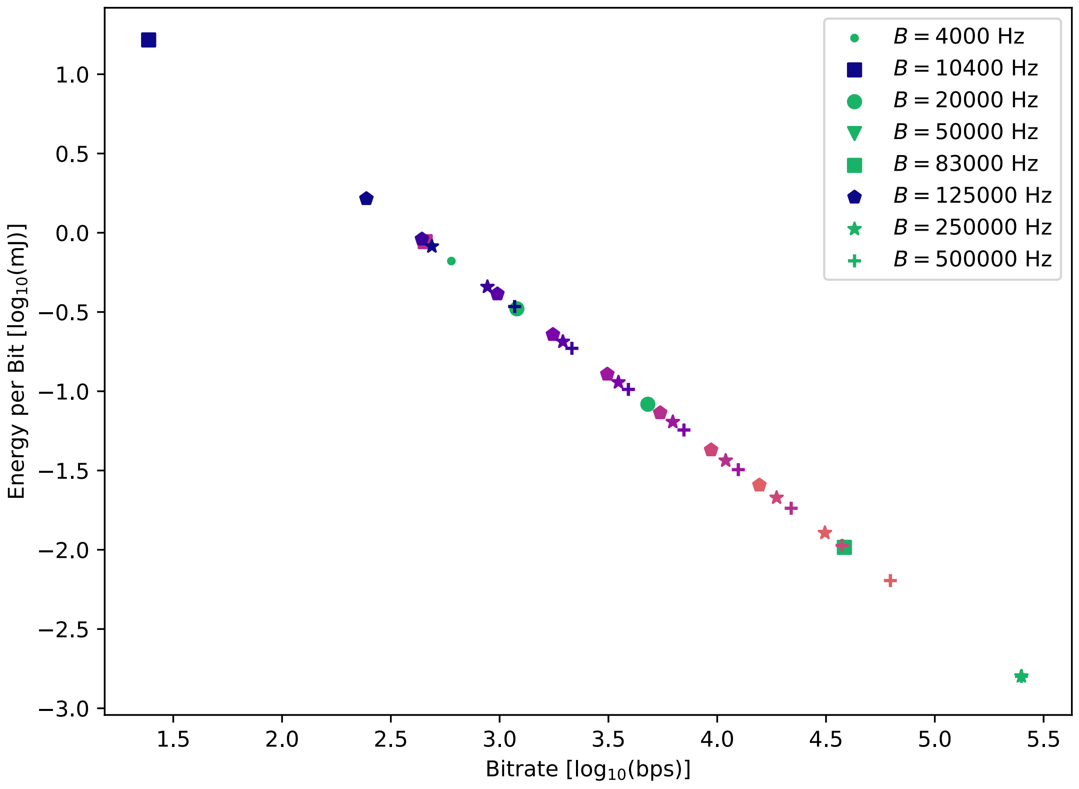


Figure E.0e: Consumed energy per bit assuming 22 dBm Tx power.

Appendix F

Timing Measurements

The appended timing measurements summarizes most important key figures of the SX1262. It is based on IPhython, and can be found and run inside `flora_tools/analysis/analyze_time.ipynb` via Jupyter.

F.1 Summary

Some timing happen to be static, i.e. they keep approximately the regardless of the set radio configuration. The timings are summarized in table F.1.

F.2 Procedure

On the computer side, the author wrote a Python 3 libry using *PyVisa* (a GPIB interfacing library) and *Pandas* (a data analysis library) to control and capture data from the Tektronix MSO4104B. The typical procedure would look as follows:

1. Setup the oscilloscope's channels, trigger, sample rate and time scale for a given precision and period.
2. Pre-configure the radios on the development kits via serial (e.g. the modulation parameters).
3. Set the oscilloscope in trigger acquisition mode.
4. Run the command on one development kit (e.g. sending a packet).
5. Wait for the oscilloscope to trigger.
6. Capture required waveforms from the oscilloscope.
7. Normalize the waveforms and mark all edges.

Figure	Short Description	Mean / Offset	Std. Dev.	Min.	Max.
Rx 2 Rf ($t_{ex} \rightarrow t_{proc}$)	Execution delay of `SetRx` command	85.235 µs	0.113 µs	84.925 µs	85.485 µs
Tx 2 Rf ($t_{ex} \rightarrow t_{proc}$)	Execution delay of `SetTx` command	126.195 µs	0.573 µs	125.613 µs	129.173 µs
Tx 2 Rf ($t_{ex} \rightarrow t_{proc}$)	Execution delay of `SetTx` command	126.195 µs	0.573 µs	125.613 µs	129.173 µs
Stdby 2 Sleep ($t_{init} \rightarrow t_{proc_{high}}$)	Init delay of `SetSleep` command	15.293 µs	1.829 µs	12 µs	20 µs
Sleep 2 Stdby ($t_{init} \rightarrow t_{proc}$)	Time taken for wakeup (retention enabled)	488.257 µs	10.144 µs	480 µs	520 µs
Configure Radio Tx LoRa ($t_{init} \rightarrow t_{ex}$)	Time taken for Tx configuration (no payload)	332.731 661 µs	26.102 µs	249.825 µs	372.774 µs
Configure Radio Rx LoRa ($t_{init} \rightarrow t_{ex}$)	Time taken for Rx configuration	271.142 µs	12.643 µs	248.562 µs	303.273 µs
Configure Radio Tx FSK ($t_{init} \rightarrow t_{ex}$)	Time taken for Tx configuration (no payload)	526.660 µs	26.700 µs	468.345 µs	575.706 µs
Configure Radio Rx FSK ($t_{init} \rightarrow t_{ex}$)	Time taken for Rx configuration	471.490 µs	14.073 µs	445.094 µs	507.195 µs
Set Payload ($t_{init} \rightarrow t_{ex}$)	Set 255 Bytes of Payload	519.949 µs	17.547 µs	483.405 µs	538.905 µs
Get Payload ($t_{init} \rightarrow t_{ex}$)	Get 255 Bytes of Payload	521.161 µs	15.937 µs	483.405 µs	540.145 µs
Set FS ($t_{init} \rightarrow t_{ex}$)	SetRx and `SetTx` for different modems	44.011 µs	24.991 µs	17.644 µs	78.233 µs
Process IRQ ($t_{irq} \rightarrow t_{ex}$)	IRQ reaction delay for reading out the radio's IRQ register	59.204 µs	0.486 µs	59.059 µs	66.166 µs
Set CAD ($t_{init} \rightarrow t_{ex}$)	Init delay of `SetCAD` command	69 µs	3 µs	67 µs	78 µs
TIM2 static offset of reaction for compare interrupt	Represents black-box sync delay	1.965 µs	0.0397 µs	−0.0801 µs (error)	0.1487 µs (error)

Table F.1: Summary of static operation timings based on author's measurements.

8. Log the time between the edges under observation.

The complete log is saved as a `.csv` file and then processed later in an analysis step generating all the plots.

See section 4.5.1 and appendix A for further information.

F.3 Bulletin

You find appended a complete printout of the
`flora_tools/flora_tools/analysis/analyze_time.ipynb` Jupyter notebook.

analyze_time

August 27, 2018

1 Preamble

This bulletin summarizes all important timing-measurements conducted on the SX1262 development kits with a Tektronix MSO4104B oscilloscope. See the `flora-tools` documentation and master thesis `Reliable 3rd Generation Data Acquisition` for further information.

If not indicated otherwise, the values are given in their non-prefixed version (e.g. seconds, hertz). Errors represent the standard deviation σ

```
In [1]: %load_ext autoreload
```

```
In [2]: import os
        import sys
        nb_dir = os.path.split(os.getcwd())[0]
        if nb_dir not in sys.path:
            sys.path.append(nb_dir)
```

```
In [3]: %config InlineBackend.figure_format = 'png'
```

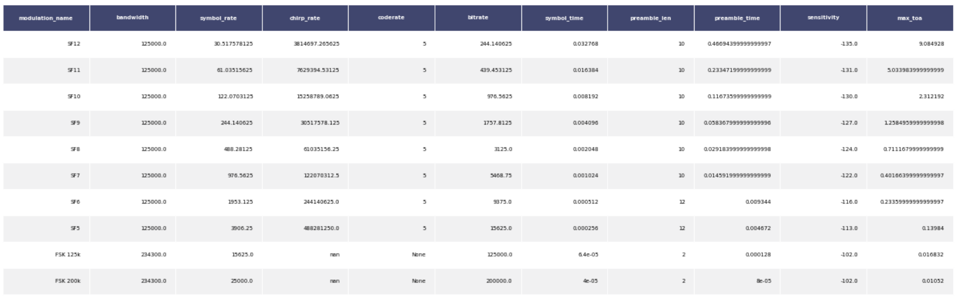
```
In [4]: %matplotlib inline
        #%matplotlib notebook

        #from IPython.display import set_matplotlib_formats
        #set_matplotlib_formats('pdf', 'svg')

        import pandas as pd
        import numpy as np
        from IPython.display import HTML, display
        import matplotlib
        import matplotlib.pyplot as plt
        import six
```

```
In [5]: def render_mpl_table(data, col_width=3.0, row_height=0.625, font_size=8,
                             header_color='#40466e', row_colors=['#f1f1f2', 'w'], edge_color='
                             bbox=[0, 0, 1, 1], header_columns=0,
                             ax=None, **kwargs):
            if ax is None:
                size = (np.array(data.shape[::-1]) + np.array([0, 1])) * np.array([col_width,
        fig, ax = plt.subplots(figsize=size)
        ax.axis('off')

    mpl_table = ax.table(cellText=data.values, bbox=bbox, colLabels=data.columns, **kwa

    mpl_table.auto_set_font_size(True)
    #mpl_table.set_fontsize(font_size)

    for k, cell in six.iteritems(mpl_table._cells):
        cell.set_edgecolor(edge_color)
        if k[0] == 0 or k[1] < header_columns:
            cell.set_text_props(weight='bold', color='w')
            cell.set_facecolor(header_color)
        else:
            cell.set_facecolor(row_colors[k[0]%len(row_colors) ])
    return ax
```

2 Gloria Radio Configurations

The following table shows the configurations under observation. They have been chosen in that way to cover a broad range of different bitrates, while still using moderate channel bandwidths. The settings are defined as constants in `flora_tools/radio_configuration.py`.

```
In [6]: %autoreload

In [7]: from flora_tools.radio_configuration import RadioConfiguration
        from flora_tools.radio_math import RadioMath

        df = pd.DataFrame(columns=['modulation_name', 'bandwidth', 'symbol_rate', 'chirp_rate'
        df.index.name = 'modulation'

        for i in range(0, 10):
            config = RadioConfiguration(i)
            math = RadioMath(config)
            df.loc[i] = [config.modulation_name, config.real_bandwidth, config.symbol_rate, co

        def colorize(row):
            color = matplotlib.colors.to_hex(RadioConfiguration(row.name).color)
            return ['background-color: {}; color: white;'.format(color) for v in row]

        df.style.apply(colorize, axis=1).set_precision(9)

Out[7]: <pandas.io.formats.style.Styler at 0x21946446f28>

In [8]: render_mpl_table(df, header_columns=0, col_width=2.0)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x219463cab38>
```

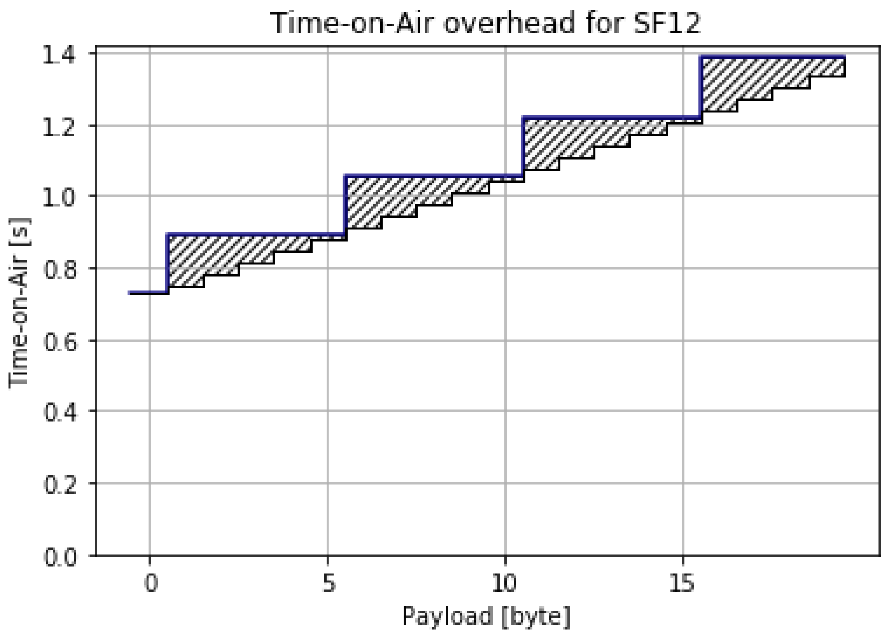


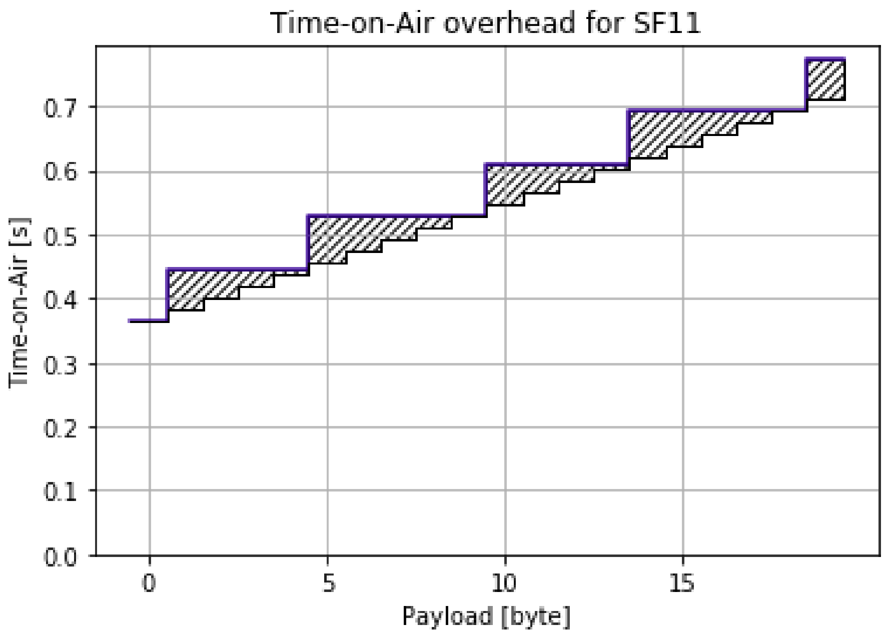
modulation_name	bandwidth	symbol_rate	chirp_rate	coderate	bitrate	symbol_time	preamble_len	preamble_time	sensitivity	max_toa
SF12	125000.0	30.517578125	3814697.265625	5	244.140625	0.032768	10	0.46694399999999997	-135.0	9.084928
SF11	125000.0	61.03515625	7629394.53125	5	439.453125	0.016384	10	0.23347199999999999	-131.0	5.033983999999999
SF10	125000.0	122.0703125	15258789.0625	5	976.5625	0.008192	10	0.11673599999999999	-130.0	2.312192
SF9	125000.0	244.140625	30517578.125	5	1757.8125	0.004096	10	0.058367999999999996	-127.0	1.2584959999999998
SF8	125000.0	488.28125	61035156.25	5	3125.0	0.002048	10	0.029183999999999998	-124.0	0.7111679999999999
SF7	125000.0	976.5625	122070312.5	5	5468.75	0.001024	10	0.014591999999999999	-122.0	0.40166399999999997
SF6	125000.0	1953.125	244140625.0	5	9375.0	0.000512	12	0.009344	-116.0	0.23359999999999997
SF5	125000.0	3906.25	488281250.0	5	15625.0	0.000256	12	0.004672	-113.0	0.13984
FSK 125k	234300.0	15625.0	nan	None	125000.0	6.4e-05	2	0.000128	-102.0	0.016832
FSK 200k	234300.0	25000.0	nan	None	200000.0	4e-05	2	8e-05	-102.0	0.01052

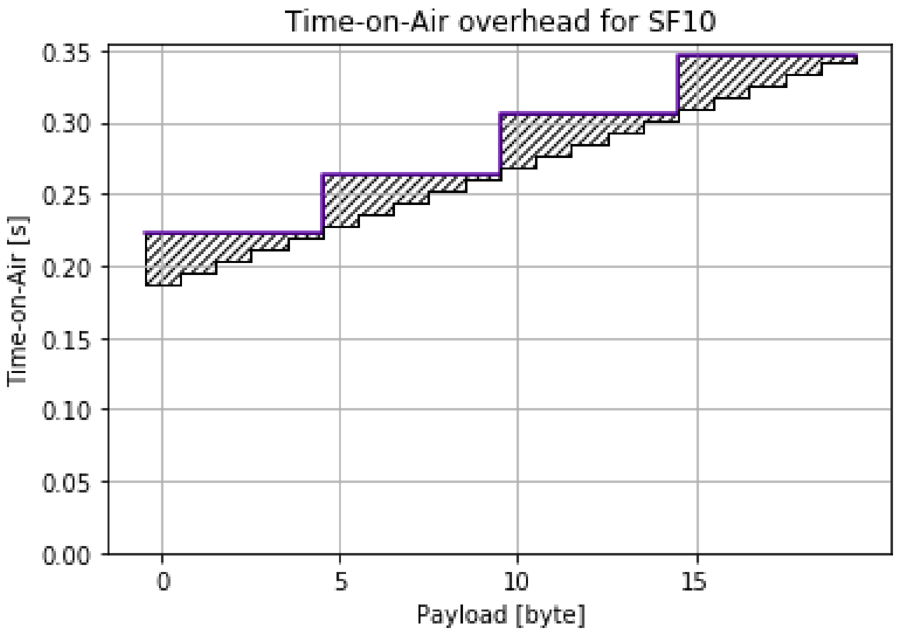
3 ToA Payload overhead

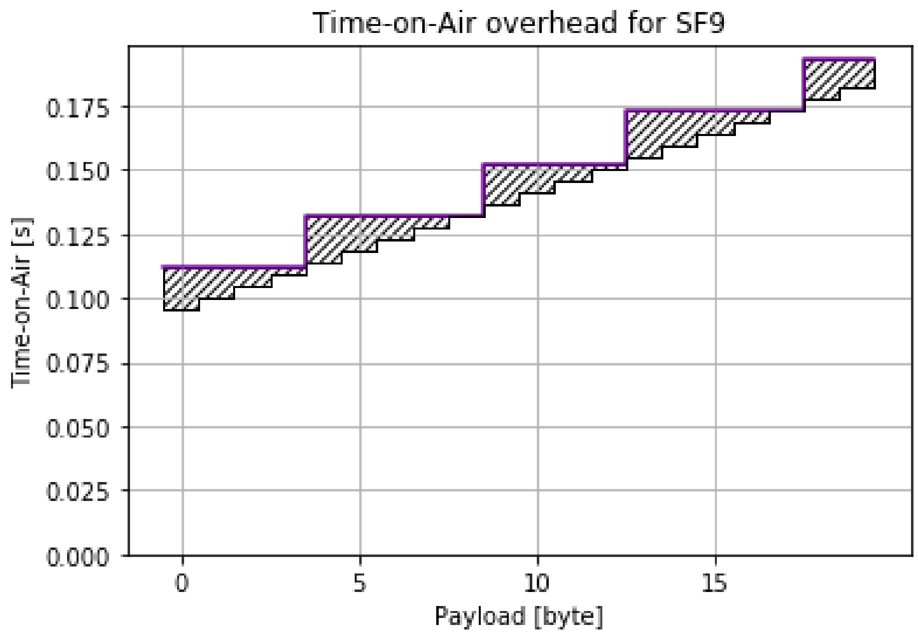
LoRa is not optimal regarding symbol fragmentation. The plots show how much overhead is induced due to the ceiling operation and various payload lenghts.

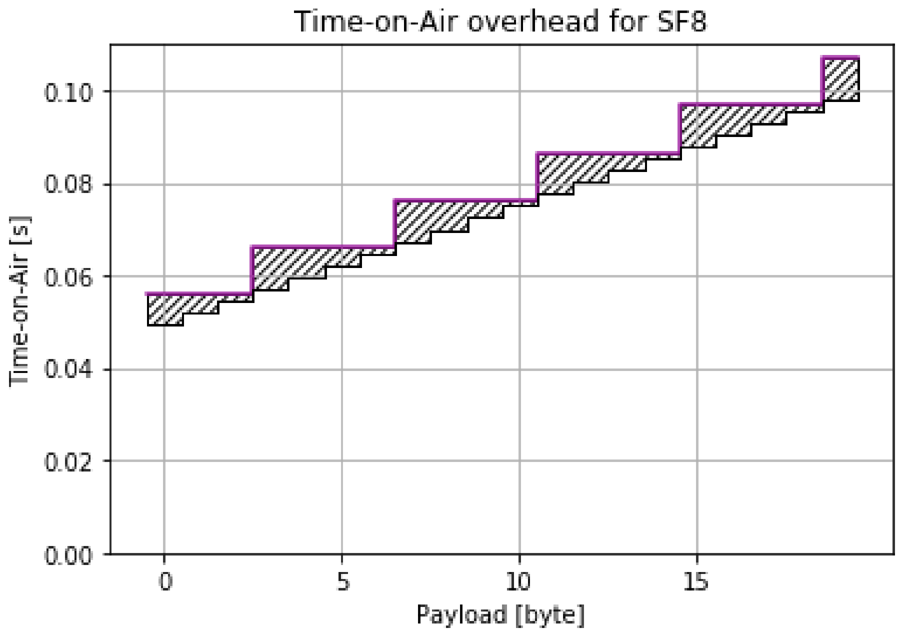
```
In [9]: MAX_PAYLOAD = 20
        payloads = np.arange(MAX_PAYLOAD)
        offset_payloads = np.insert(payloads, 0, -1) + 0.5

        for i in range(10):
            config = RadioConfiguration(i)
            math = RadioMath(config)

            toas = [math.get_message_toa(j) for j in payloads]
            toas = np.insert(toas, 0, math.get_message_toa(0))

            toas_without_overhead = [math.get_message_toa(j, ceil_overhead=False) for j in pay
            toas_without_overhead = np.insert(toas_without_overhead, 0, math.get_message_toa(0

            plt.step(offset_payloads, toas, c=config.color)
            plt.step(offset_payloads, toas_without_overhead, ":",  c='k', linewidth=1)

            plt.xticks(np.arange(0, MAX_PAYLOAD, step=5))

            ax = plt.gca()
            ax.grid(True)

            plt.ylim(ymin=0)

            ax.fill_between(offset_payloads, toas, toas_without_overhead, step='pre', hatch='/


            plt.title("Time-on-Air overhead for {}".format(config.modulation_name))
            plt.xlabel("Payload [byte]")
            plt.ylabel("Time-on-Air [s]")

            plt.show()
```

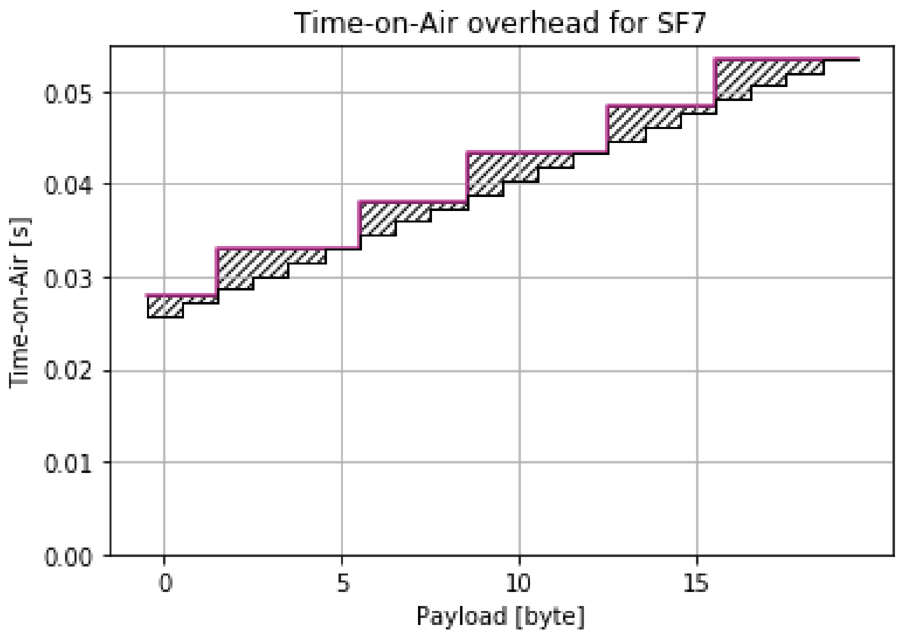


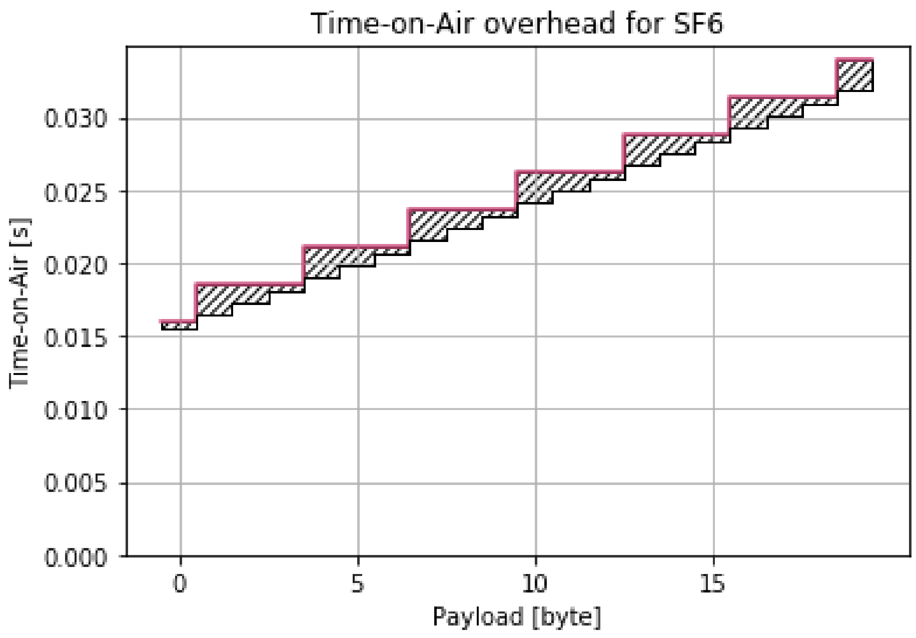

Time-on-Air overhead for SF11
Time-on-Air [s]
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
5
10
15
Payload [byte]

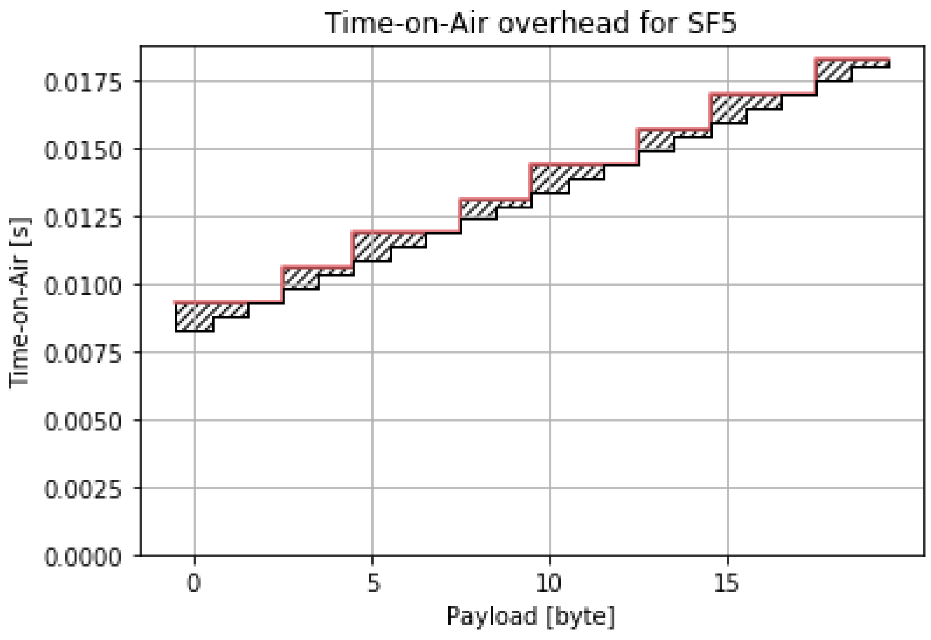

Time-on-Air overhead for SF10
Time-on-Air [s]
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0
5
10
15
Payload [byte]

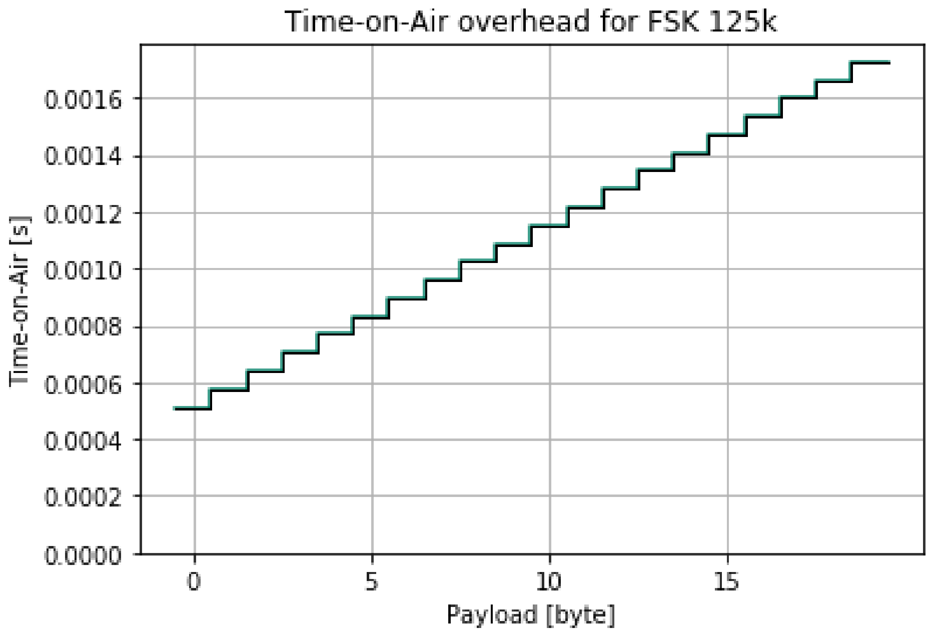

Time-on-Air overhead for SF9
Time-on-Air [s]
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
5
10
15
Payload [byte]

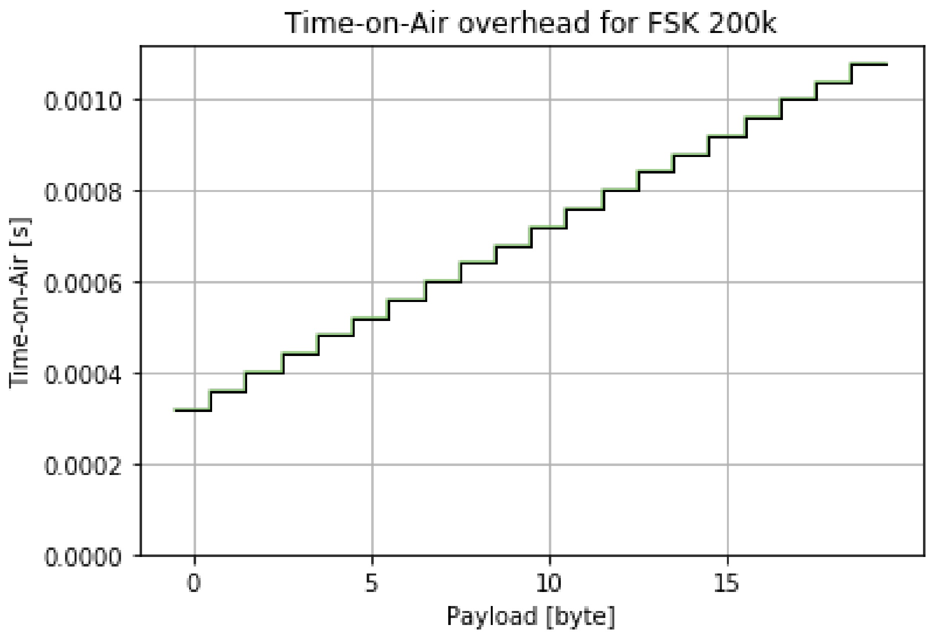

Time-on-Air overhead for SF8
Time-on-Air [s]
0.00
0.02
0.04
0.06
0.08
0.10
0
5
10
15
Payload [byte]


Time-on-Air overhead for SF7
Time-on-Air [s]
0.00
0.01
0.02
0.03
0.04
0.05
0
5
10
15
Payload [byte]


Time-on-Air overhead for SF6
Time-on-Air [s]
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0
5
10
15
Payload [byte]


Time-on-Air overhead for SF5
Time-on-Air [s]
Payload [byte]
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0
5
10
15

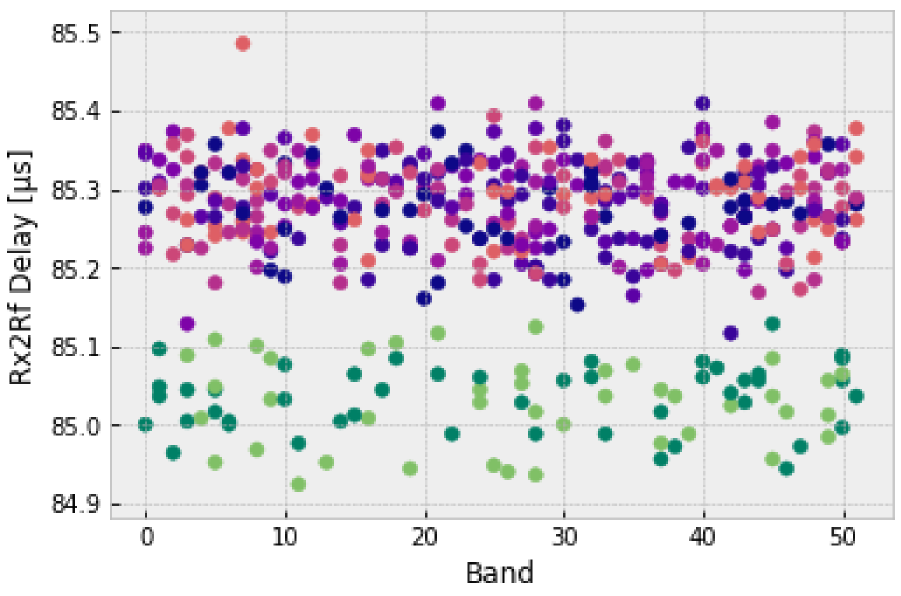

Time-on-Air overhead for FSK 125k
Time-on-Air [s]
Payload [byte]
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014
0.0016
0
5
10
15



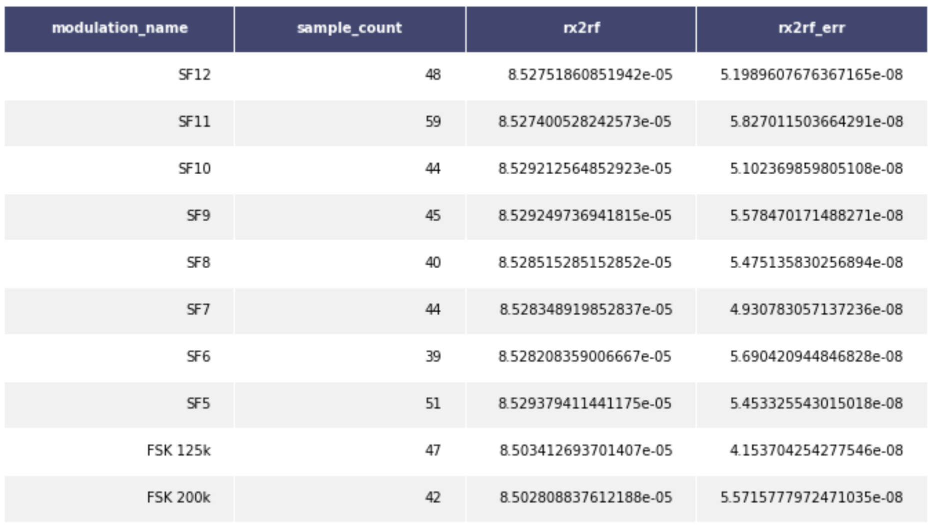
4 Rx 2 Rf

The measurements depict the the time from t_{ex} to t_{proc} when changing from Standby in Rx mode. Note the different value for LoRa (purple) and GFSK (green). For this experiment, `NSS` and `BUSY` signal of one development kit were required.

```
In [10]: from flora_tools.experiments.measure_time_rx2rf import MeasureTimeRx2Rf

         rx2rf = MeasureTimeRx2Rf()
         df = pd.read_csv("../../data/{}.csv".format(rx2rf.name), index_col="sample")
         mod_delays, band_delays = rx2rf.analyze(df)
```



```
In [11]: %matplotlib inline
         mod_delays.style.set_precision(9)

Out[11]: <pandas.io.formats.style.Styler at 0x2194a111e48>

In [12]: render_mpl_table(mod_delays, header_columns=0, col_width=3.0)

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x2194a2d0080>
```

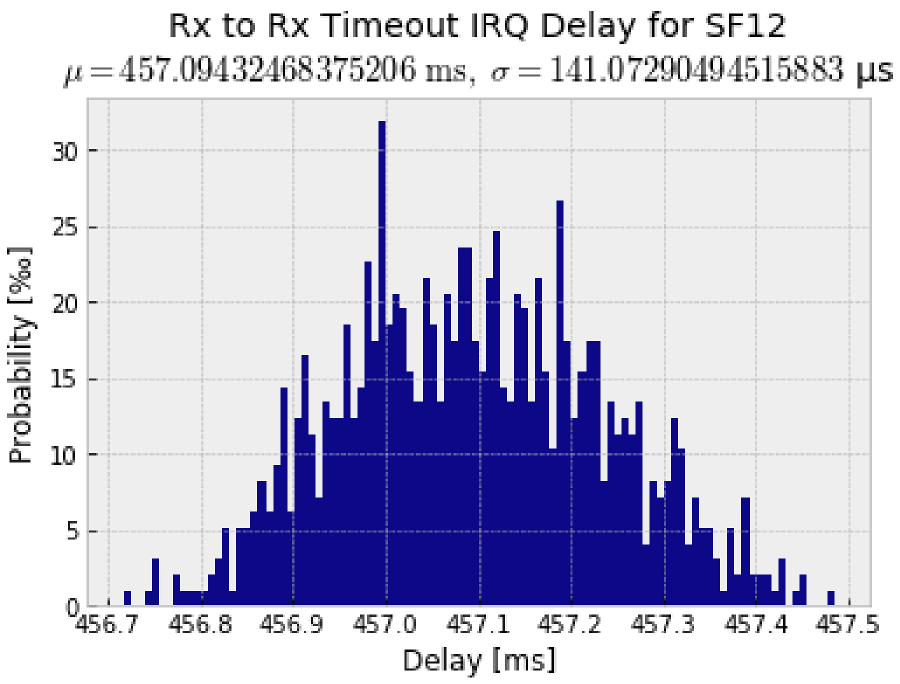


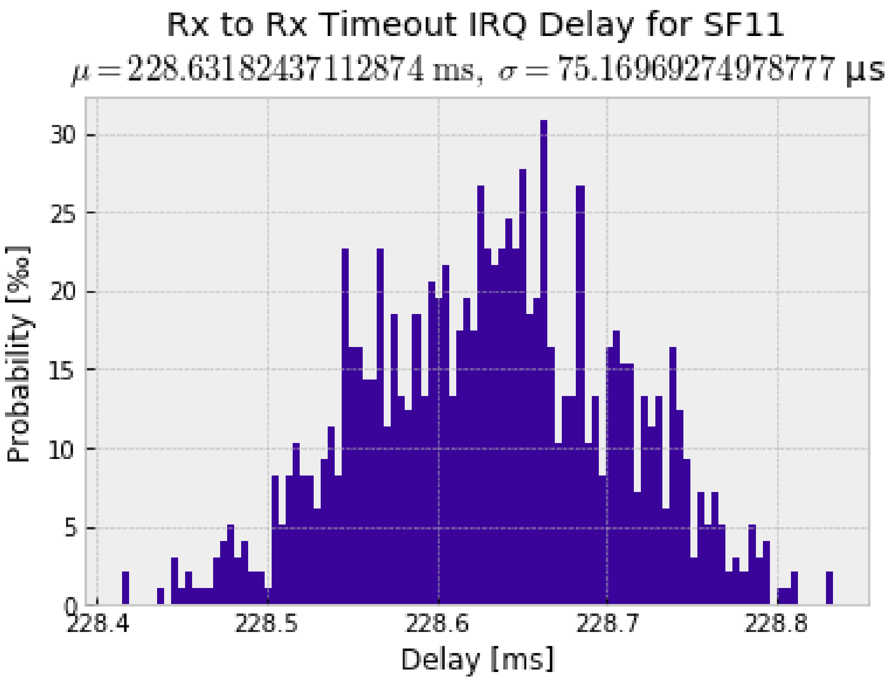
modulation_name	sample_count	rx2rf	rx2rf_err
SF12	48	8.52751860851942e-05	5.1989607676367165e-08
SF11	59	8.527400528242573e-05	5.827011503664291e-08
SF10	44	8.529212564852923e-05	5.102369859805108e-08
SF9	45	8.529249736941815e-05	5.578470171488271e-08
SF8	40	8.528515285152852e-05	5.475135830256894e-08
SF7	44	8.528348919852837e-05	4.930783057137236e-08
SF6	39	8.528208359006667e-05	5.690420944846828e-08
SF5	51	8.529379411441175e-05	5.453325543015018e-08
FSK 125k	47	8.503412693701407e-05	4.153704254277546e-08
FSK 200k	42	8.502808837612188e-05	5.5715777972471035e-08

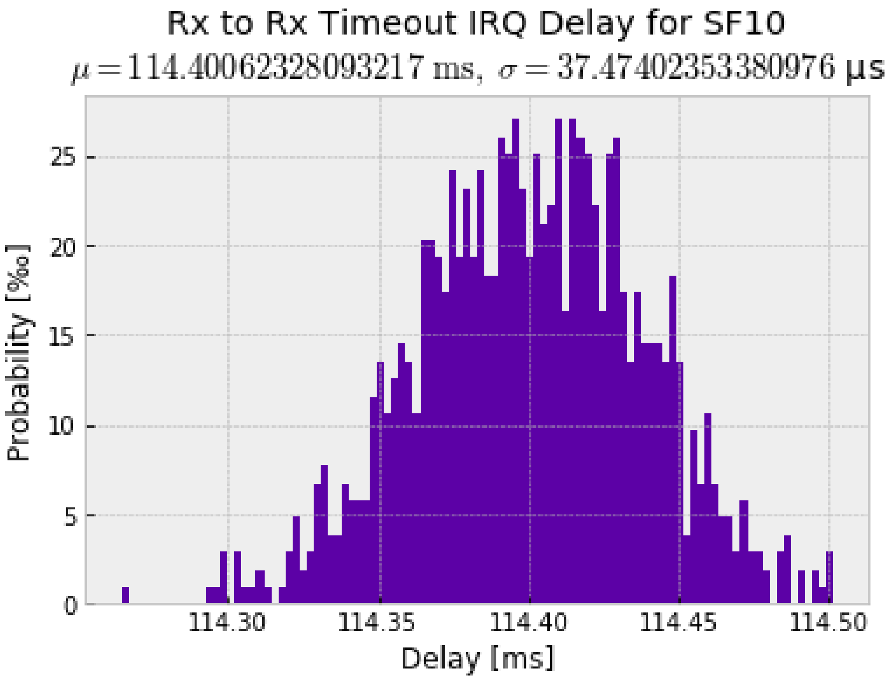
5 Rx 2 Rx Timeout

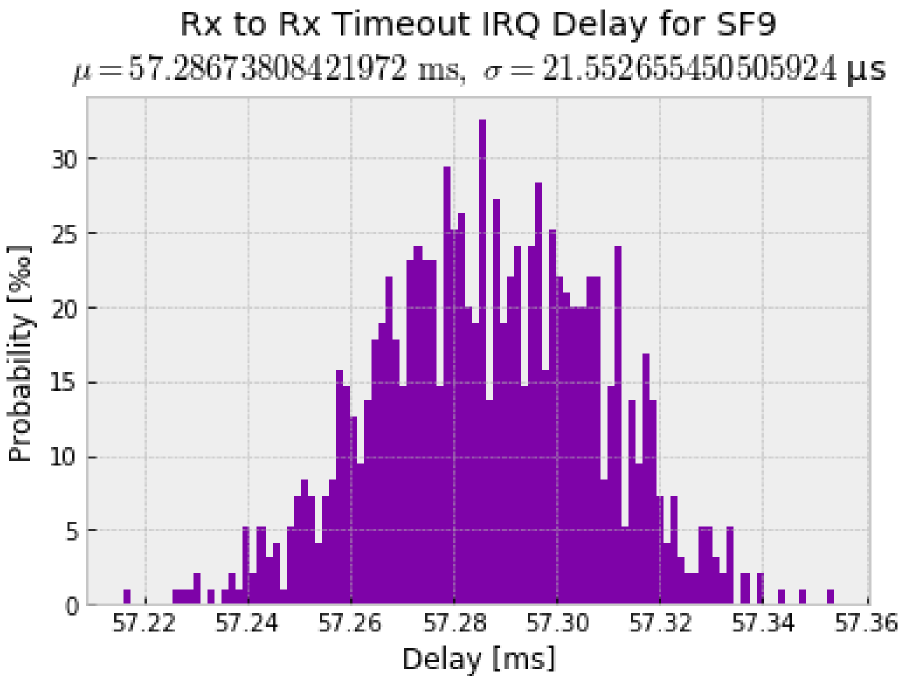
The measurements depict the the time from t_{ex} to t_{irq} when changing in Rx mode and letting it timeout. For this experiment, NSS and DIO1 signal of one development kit were required.

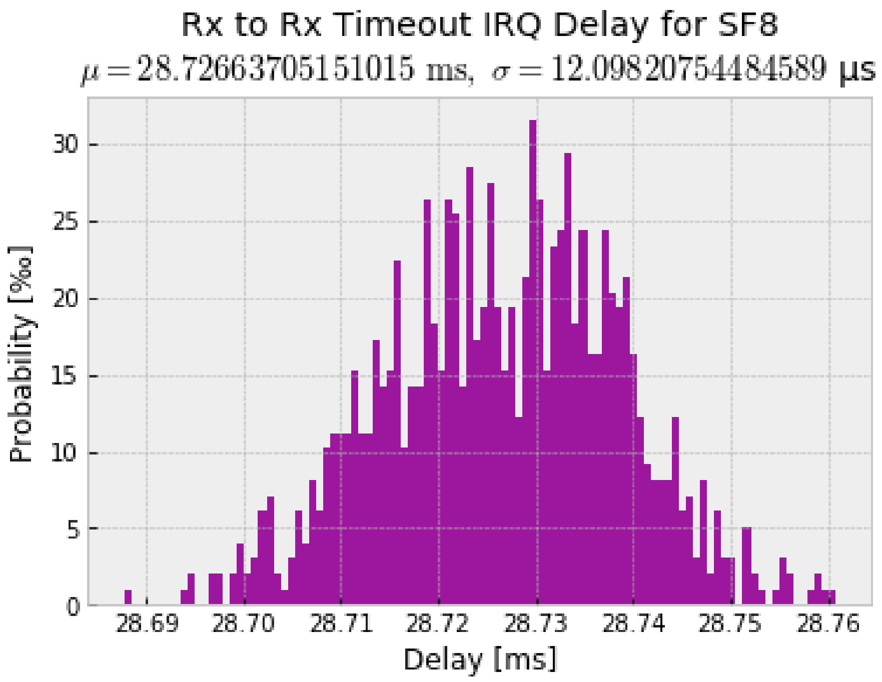
```
In [13]: from flora_tools.experiments.measure_time_rx2rxtimeout import MeasureTimeRx2RxTimeout

         rx2rxtimeout = MeasureTimeRx2RxTimeout()
         df = pd.read_csv("../../data/{}.csv".format(rx2rxtimeout.name), index_col="sample")
         delays = rx2rxtimeout.analyze(df)
```



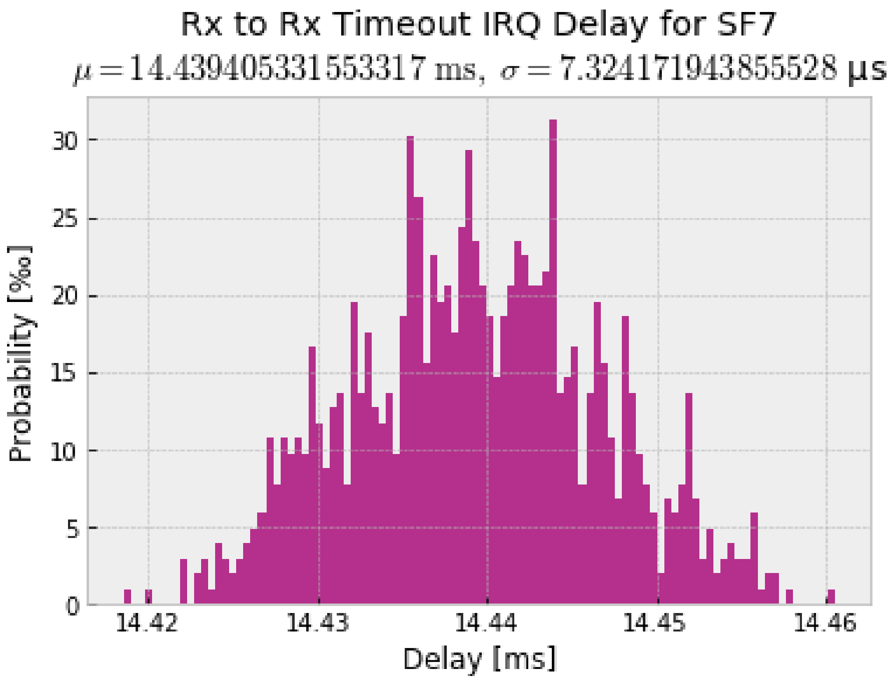

Rx to Rx Timeout IRQ Delay for SF11
$\mu = 228.63182437112874\,\mathrm{ms},\ \sigma = 75.16969274978777$ µs
Probability [‰]
Delay [ms]
0
5
10
15
20
25
30
228.4
228.5
228.6
228.7
228.8


Rx to Rx Timeout IRQ Delay for SF10
$\mu = 114.40062328093217\,\mathrm{ms},\ \sigma = 37.47402353380976$ µs
Probability [‰]
Delay [ms]
0
5
10
15
20
25
114.30
114.35
114.40
114.45
114.50

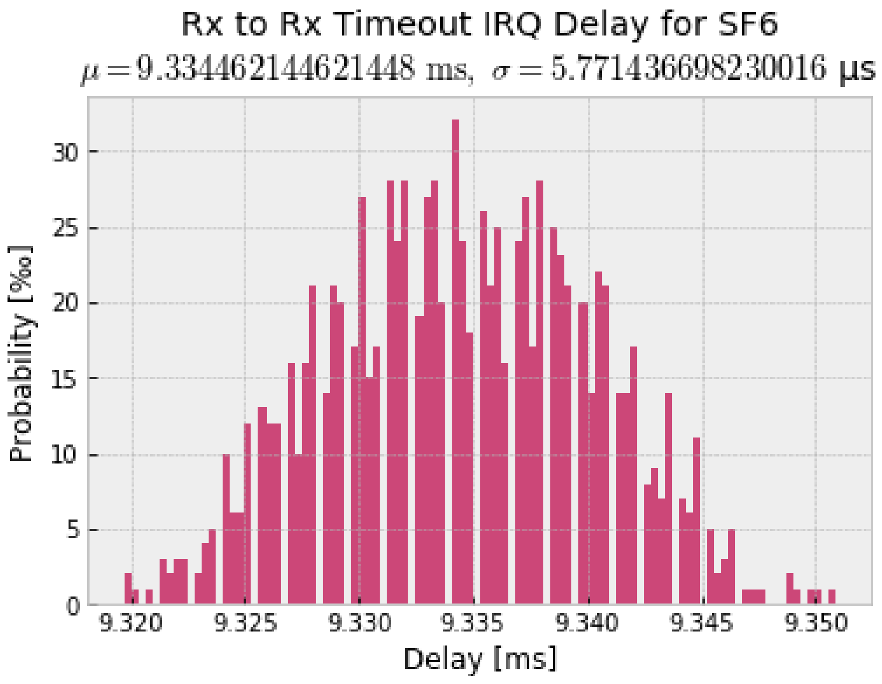

Rx to Rx Timeout IRQ Delay for SF9
$\mu = 57.28673808421972$ ms, $\sigma = 21.552655450505924$ µs
Probability [‰]
0
5
10
15
20
25
30
57.22
57.24
57.26
57.28
57.30
57.32
57.34
57.36
Delay [ms]

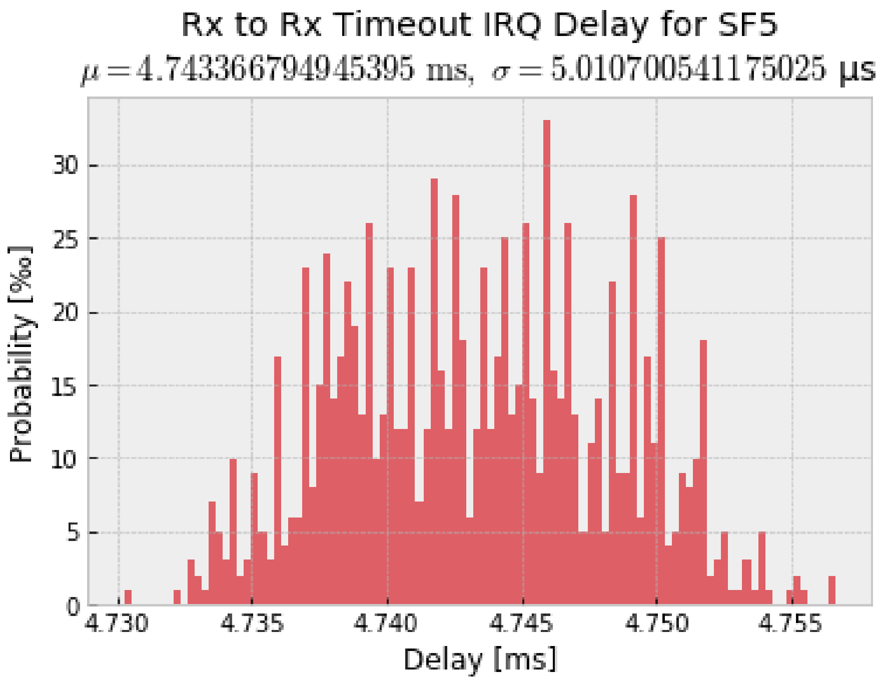


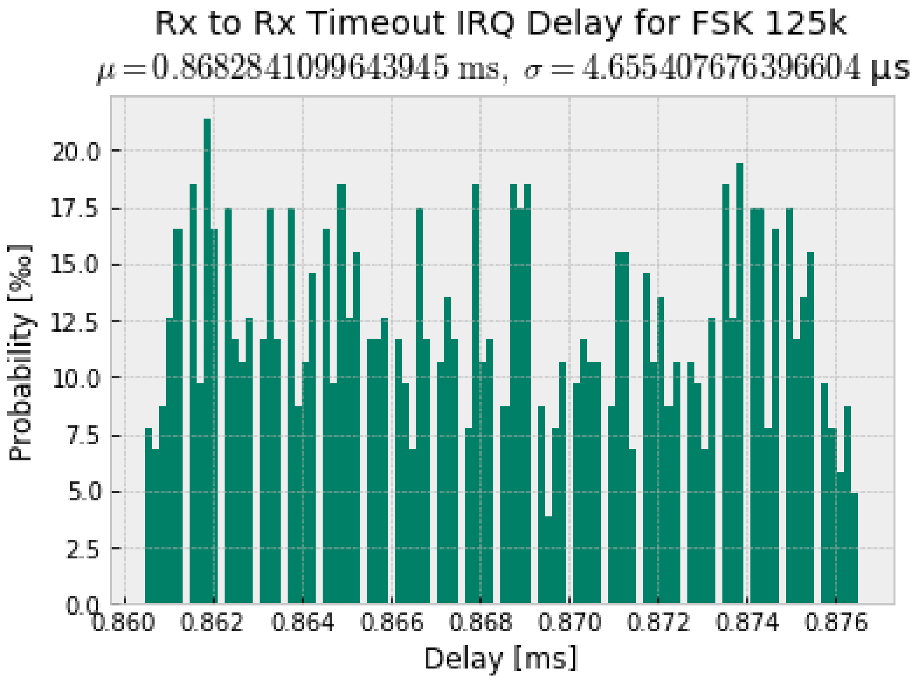
Rx to Rx Timeout IRQ Delay for SF8
$\mu = 28.72663705151015$ ms, $\sigma = 12.09820754484589$ µs
Probability [‰]
0
5
10
15
20
25
30
28.69
28.70
28.71
28.72
28.73
28.74
28.75
28.76
Delay [ms]

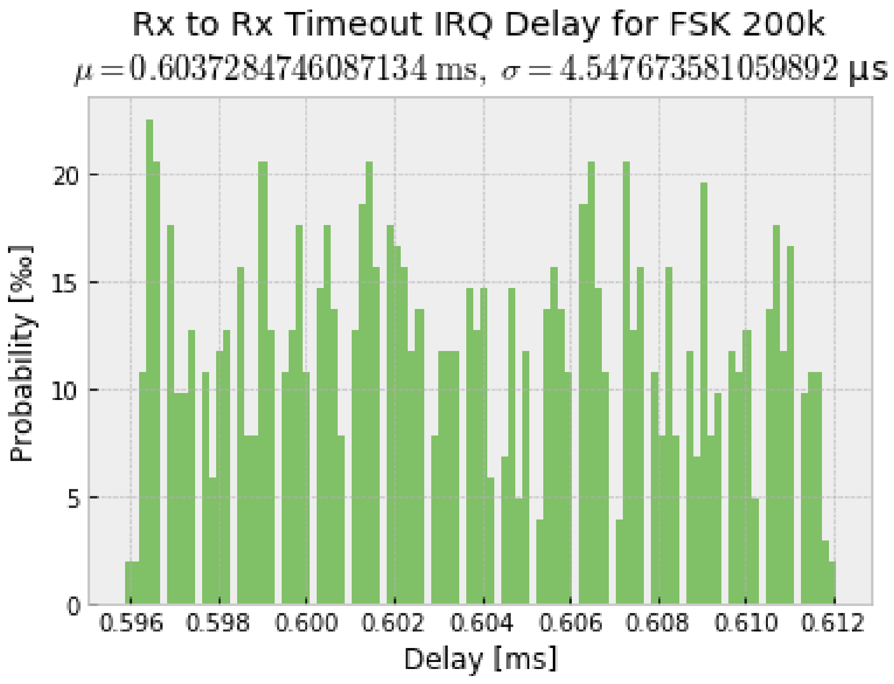


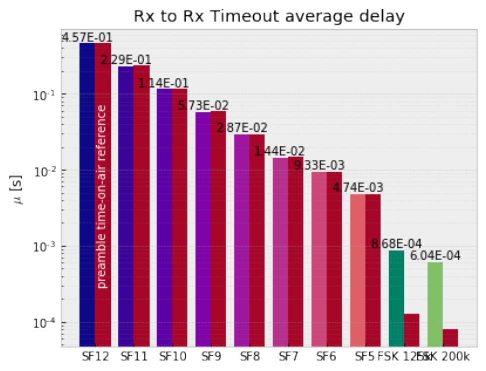
Rx to Rx Timeout IRQ Delay for SF7
$\mu = 14.439405331553317$ ms, $\sigma = 7.324171943855528$ µs
Probability [‰]
0
5
10
15
20
25
30
14.42
14.43
14.44
14.45
14.46
Delay [ms]

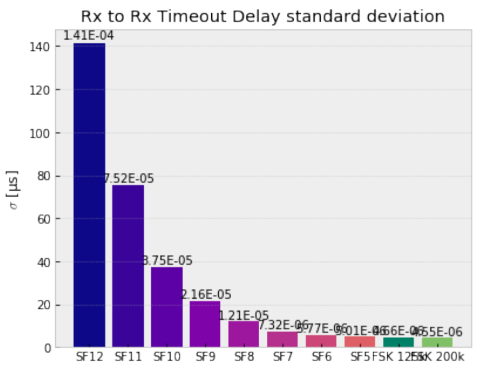

Rx to Rx Timeout IRQ Delay for SF6
$\mu = 9.334462144621448$ ms, $\sigma = 5.771436698230016$ µs
Probability [‰]
0
5
10
15
20
25
30
9.320
9.325
9.330
9.335
9.340
9.345
9.350
Delay [ms]


Rx to Rx Timeout IRQ Delay for SF5
$\mu = 4.743366794945395$ ms, $\sigma = 5.010700541175025$ µs
Probability [‰]
Delay [ms]
0
5
10
15
20
25
30
4.730
4.735
4.740
4.745
4.750
4.755


Rx to Rx Timeout IRQ Delay for FSK 125k
$\mu = 0.8682841099643945$ ms, $\sigma = 4.655407676396604$ µs
Probability [‰]
Delay [ms]
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.860
0.862
0.864
0.866
0.868
0.870
0.872
0.874
0.876



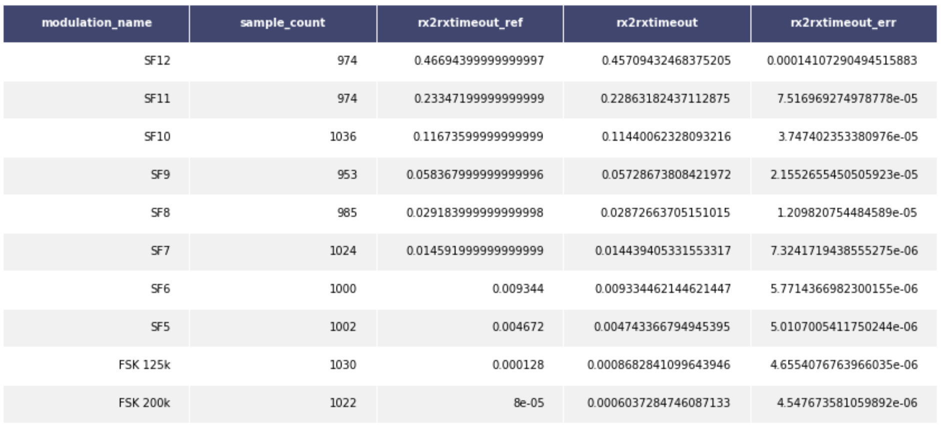




```
In [14]: %matplotlib inline
         delays.style.bar(subset=['rx2rxtimeout', 'rx2rxtimeout_err'], color='lightgray').set_

Out[14]: <pandas.io.formats.style.Styler at 0x2194a06f668>
```

```
In [15]: render_mpl_table(delays, header_columns=0, col_width=3.0)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x2194ab40f28>
```

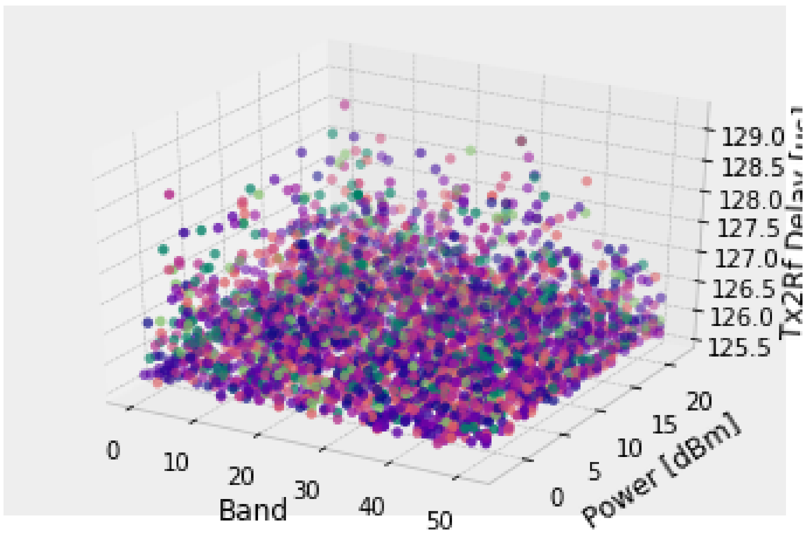


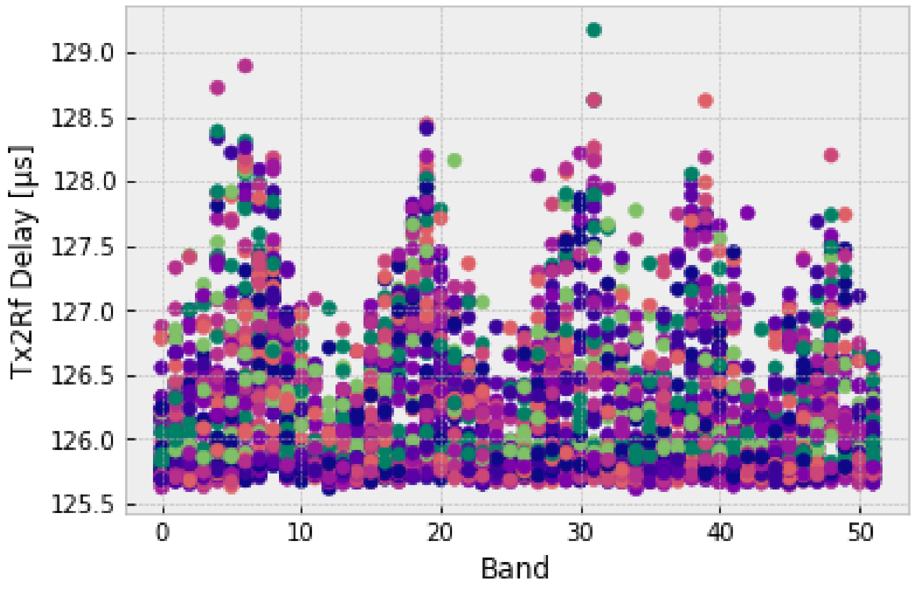
modulation_name	sample_count	rx2rxtimeout_ref	rx2rxtimeout	rx2rxtimeout_err
SF12	974	0.46694399999999997	0.45709432468375205	0.00014107290494515883
SF11	974	0.23347199999999999	0.22863182437112875	7.516969274978778e-05
SF10	1036	0.11673599999999999	0.11440062328093216	3.747402353380976e-05
SF9	953	0.058367999999999996	0.05728673808421972	2.1552655450505923e-05
SF8	985	0.029183999999999998	0.02872663705151015	1.209820754484589e-05
SF7	1024	0.014591999999999999	0.014439405331553317	7.3241719438555275e-06
SF6	1000	0.009344	0.009334462144621447	5.7714366982300155e-06
SF5	1002	0.004672	0.004743366794945395	5.0107005411750244e-06
FSK 125k	1030	0.000128	0.0008682841099643946	4.6554076763966035e-06
FSK 200k	1022	8e-05	0.0006037284746087133	4.547673581059892e-06

6 Tx 2 Rf

The measurements depict the the time from t_{ex} to t_{proc} when changing into Tx mode. For this experiment, NSS and BUSY signal of one development kit were required. Note the dependcy of variation in delay on the selected band.

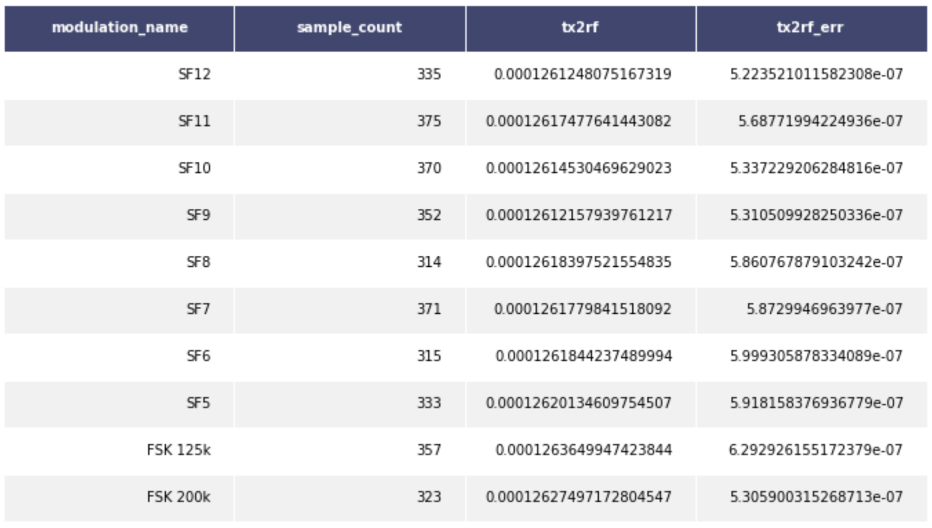
```
In [16]: from flora_tools.experiments.measure_time_tx2rf import MeasureTimeTx2Rf

         tx2rf = MeasureTimeTx2Rf()
         df = pd.read_csv("../../data/{}.csv".format(tx2rf.name), index_col="sample")
         mod_delays, band_delays = tx2rf.analyze(df)
```





```
In [17]: %matplotlib inline
         mod_delays.style.bar(subset=['tx2rf', 'tx2rf_err'], color='lightgray').set_precision(

Out[17]: <pandas.io.formats.style.Styler at 0x2194abf95c0>
```

```
In [18]: render_mpl_table(mod_delays, header_columns=0, col_width=3.0)

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x21948ece5f8>
```



modulation_name	sample_count	tx2rf	tx2rf_err
SF12	335	0.0001261248075167319	5.223521011582308e-07
SF11	375	0.00012617477641443082	5.68771994224936e-07
SF10	370	0.00012614530469629023	5.337229206284816e-07
SF9	352	0.00012612157939761217	5.310509928250336e-07
SF8	314	0.00012618397521554835	5.860767879103242e-07
SF7	371	0.0001261779841518092	5.8729946963977e-07
SF6	315	0.0001261844237489994	5.999305878334089e-07
SF5	333	0.00012620134609754507	5.918158376936779e-07
FSK 125k	357	0.0001263649947423844	6.292926155172379e-07
FSK 200k	323	0.00012627497172804547	5.305900315268713e-07

```
In [19]: band_delays.style.bar(subset=['tx2rf', 'tx2rf_err'], color='lightgray').set_precision

Out[19]: <pandas.io.formats.style.Styler at 0x2194a044080>

In [20]: render_mpl_table(band_delays, header_columns=0, col_width=3.0)

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2194a6b6198>
```

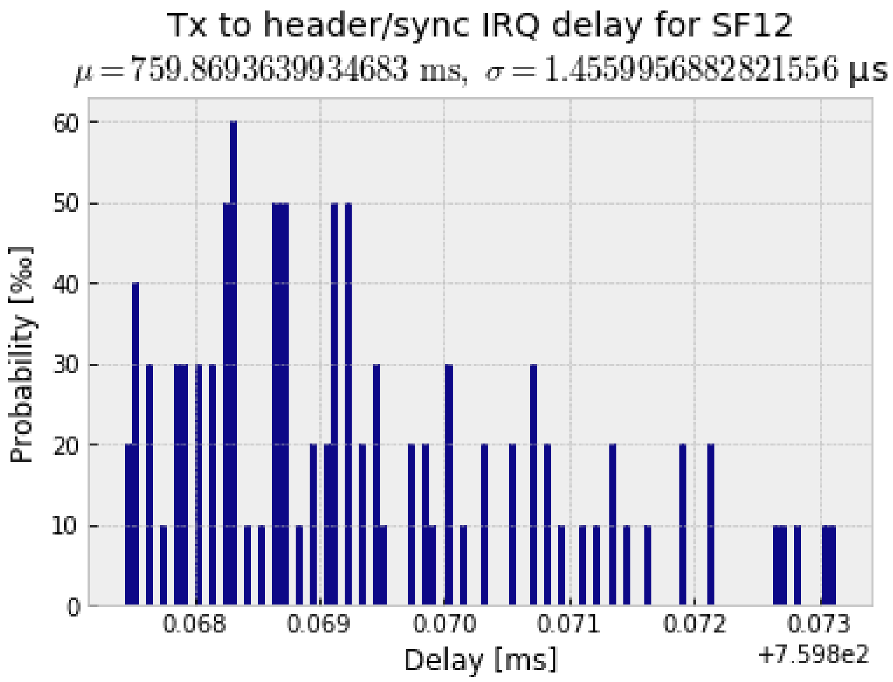
sample_count	tx2rf	tx2rf_err
65.0	0.0001259594749793652	2.8225982496420607e-07
70.0	0.0001260106886783154	3.732033161789936e-07
63.0	0.0001260628479300666	4.2794249605851696e-07
64.0	0.0001261268237682377	4.540888584556173e-07
72.0	0.0001263868194237498	7.666430607713772e-07
65.0	0.00012637141756032945	6.085005701586448e-07
73.0	0.00012670460951184854	8.606827647342731e-07
70.0	0.00012665435225780831	6.630906582584675e-07
62.0	0.00012684810719074932	6.096795432107102e-07
60.0	0.0001262871962052954	4.3146945230329233e-07
53.0	0.0001261491860201621	4.083911480975127e-07
56.0	0.0001259341879133077	2.730072826251357e-07
54.0	0.00012586599940073476	2.3779483424403985e-07
76.0	0.00012591720654048645	2.4933268375641023e-07
65.0	0.00012590212055966712	2.280050128272747e-07
65.0	0.00012607012223968394	3.9821905743741607e-07
69.0	0.00012623227681552178	4.631846404554811e-07
66.0	0.00012641520354597492	5.795072221306071e-07
65.0	0.00012664397413204905	7.226257659078355e-07
75.0	0.0001268078814121475	8.15871553039728e-07
65.0	0.00012654218849880806	5.065673862380583e-07
68.0	0.0001261874383449717	4.853661728708015e-07
64.0	0.00012614832398323984	4.1099948999442246e-07
74.0	0.00012592520519799793	2.750258323228481e-07
71.0	0.00012586390652638923	1.9219580461612416e-07
55.0	0.00012591115002059115	2.7313963050838447e-07
60.0	0.00012599379327126606	2.9985377875268324e-07
64.0	0.00012617726177261774	4.660837467077896e-07
84.0	0.00012637473993787555	4.950980899898628e-07
67.0	0.0001264702796281694	7.357801630791739e-07
53.0	0.00012633424447452025	7.092632167771449e-07
62.0	0.00012668294424879735	9.36933264875469e-07
64.0	0.00012616663666636667	5.481531704704523e-07
67.0	0.00012610251475649086	4.5541257441962017e-07
84.0	0.0001259611643735485	3.8764622867636463e-07
54.0	0.00012602118613778732	3.845651419825976e-07
81.0	0.00012599834640321711	3.9607113296658064e-07
70.0	0.00012610246102461029	5.214139414058113e-07
75.0	0.00012628734287342878	6.834608161349821e-07
66.0	0.00012648659819931534	7.659363833691479e-07
57.0	0.0001264545277031718	5.280519006982369e-07
50.0	0.00012621214212142124	5.332140495547263e-07
66.0	0.00012593271387259327	3.3783930497089614e-07
74.0	0.0001258797452839393	2.0764634835775143e-07
60.0	0.000125929859298593	2.889831659752034e-07
69.0	0.0001260163326270944	4.2875631787360253e-07
82.0	0.0001260117967033329	3.9431635283733474e-07
69.0	0.00012608265213086918	4.7587959458266656e-07
57.0	0.0001266260381902065	5.914030194854522e-07
59.0	0.0001261750922593972	5.342816871899246e-07
76.0	0.0001259144696710125	2.8911926064913053e-07
70.0	0.00012591645916459166	2.4747231478276444e-07

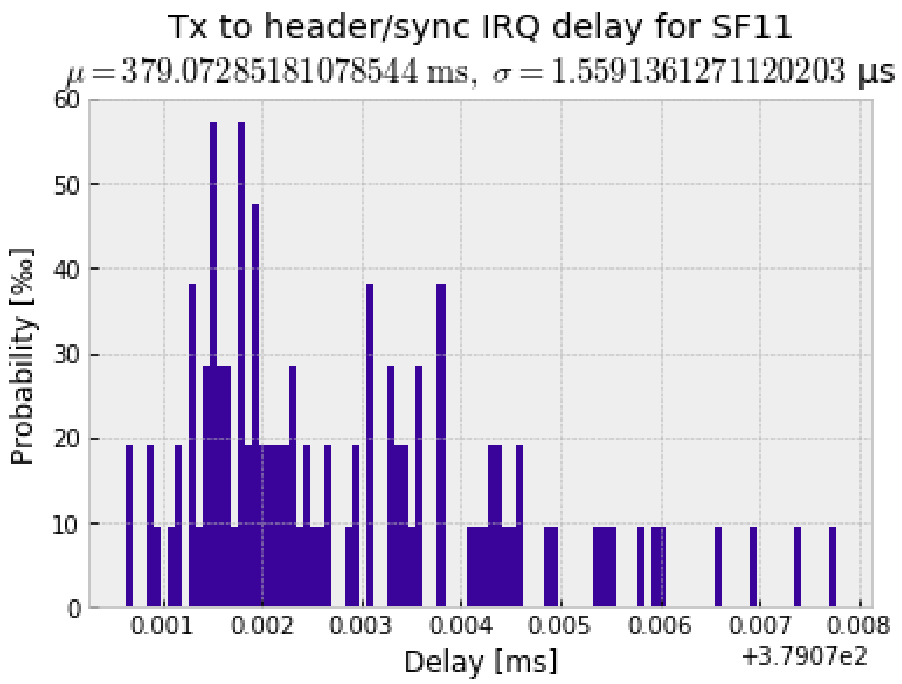
7 TX 2 Sync

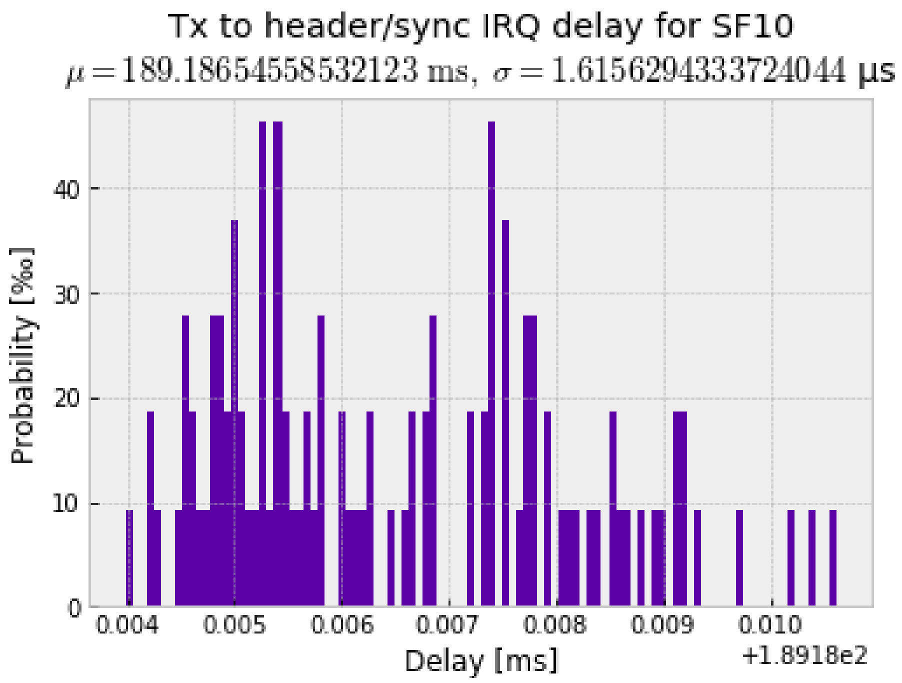
The measurements depict the the time from t_{ex} on DevKit B to t_{irq} on DevKit A when changing in Tx mode with DevKit B and receiving the message on node A. For this experiment, `NSS` signal on DevKit B and `DIO1` signal on DevKit A are required.

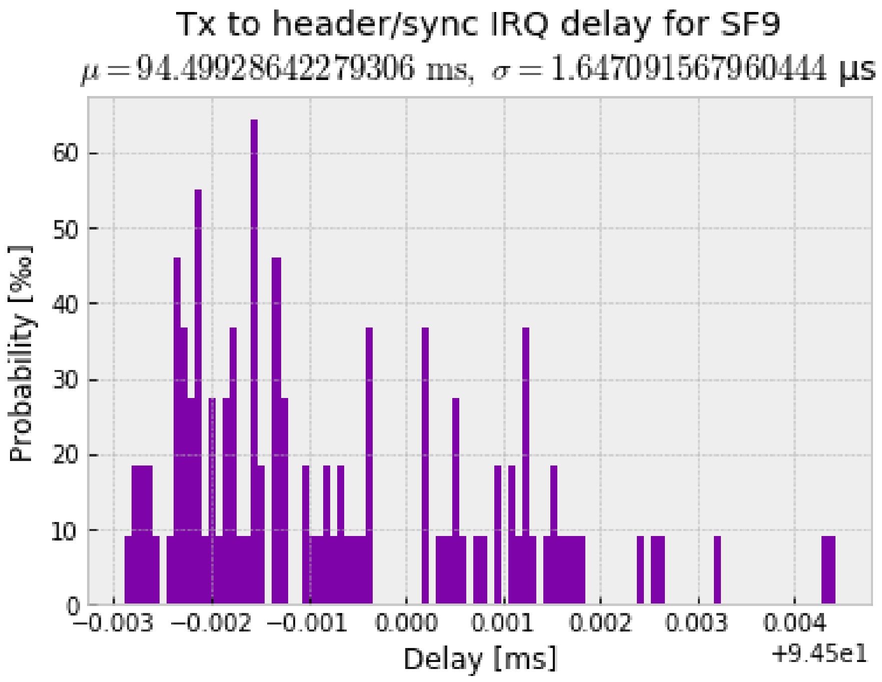
Explicit header (LoRa) and syncword detection (GFSK) interrupt are enabled.

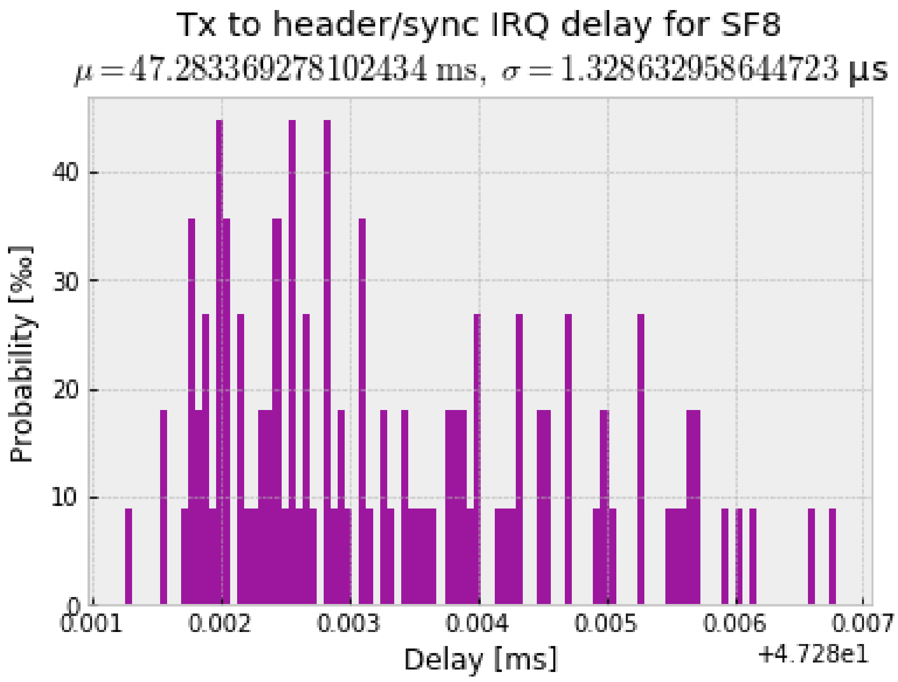
```
In [21]: %matplotlib inline

         from flora_tools.experiments.measure_time_tx2sync import MeasureTimeTx2Sync

         txex2sync = MeasureTimeTx2Sync()
         df = pd.read_csv("../../data/{}.csv".format(txex2sync.name + "-Preambles_LoRa10_FSK8"
         
         delays = txex2sync.analyze(df)
```

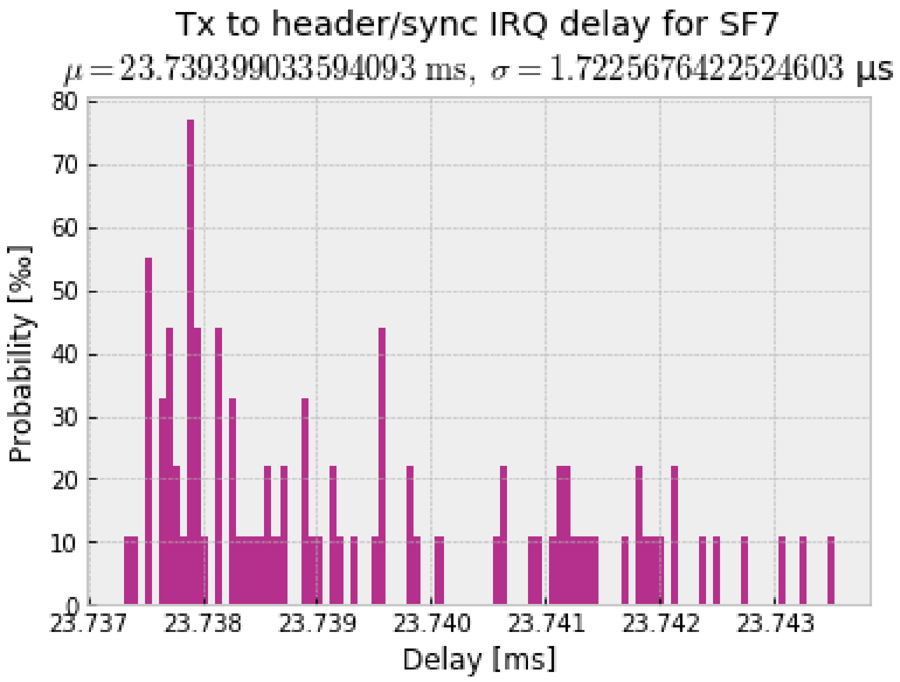


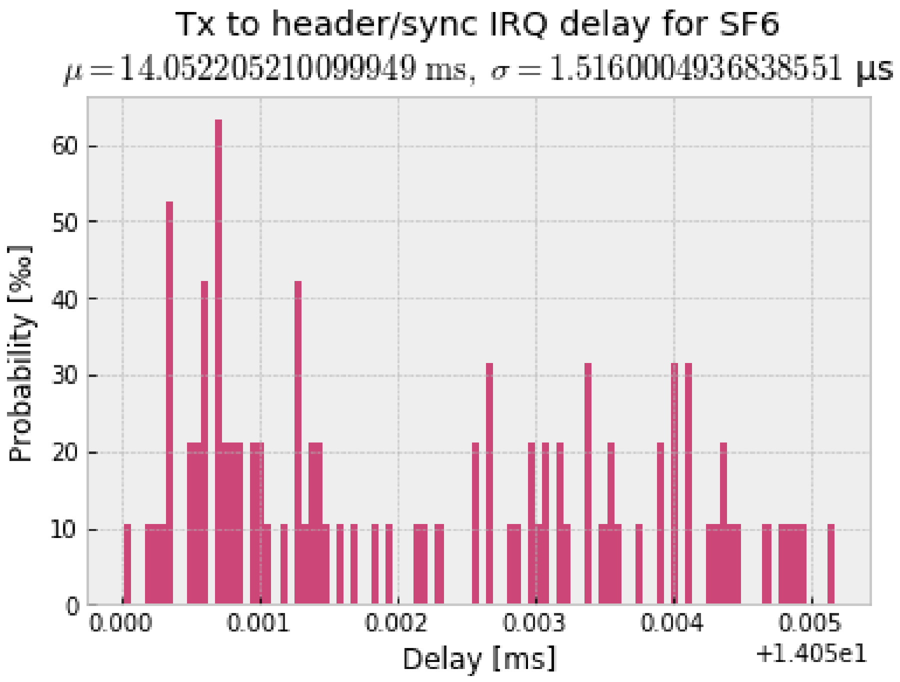

Tx to header/sync IRQ delay for SF11
$\mu = 379.07285181078544\ \mathrm{ms},\ \sigma = 1.5591361271120203$ µs
Probability [‰]
0
10
20
30
40
50
60
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
Delay [ms]
+3.7907e2

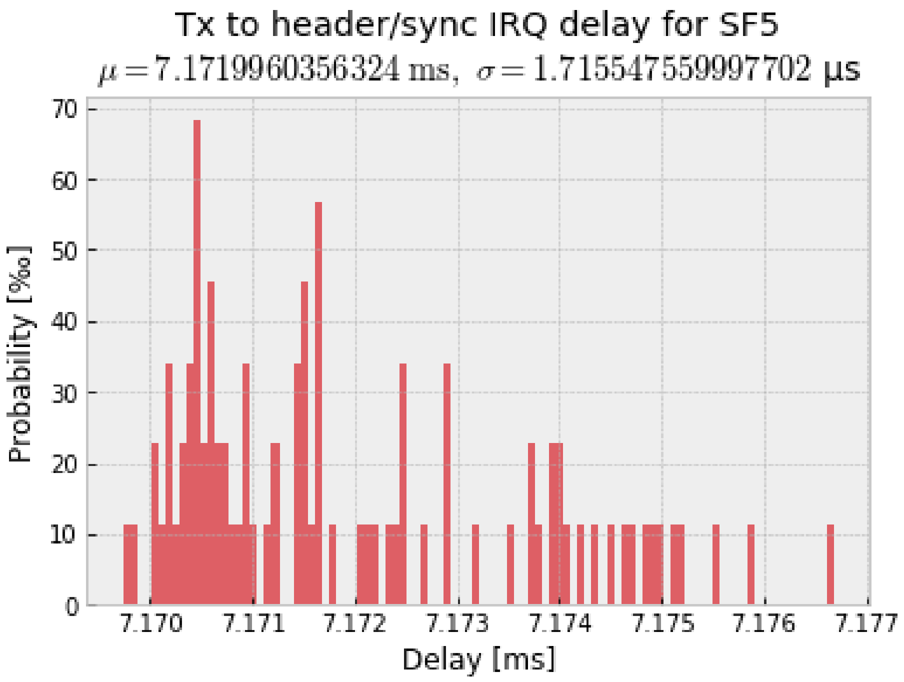

Tx to header/sync IRQ delay for SF10
$\mu = 189.18654558532123\ \mathrm{ms},\ \sigma = 1.6156294333724044$ µs
Probability [‰]
0
10
20
30
40
0.004
0.005
0.006
0.007
0.008
0.009
0.010
Delay [ms]
+1.8918e2

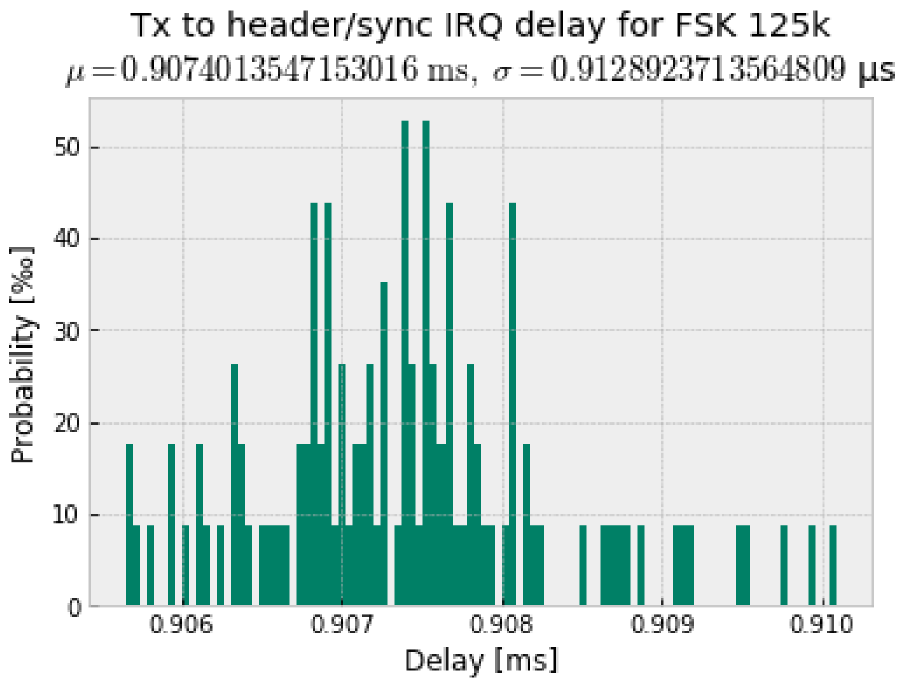

Tx to header/sync IRQ delay for SF9
$\mu = 94.49928642279306$ ms, $\sigma = 1.647091567960444$ µs
Probability [‰]
0
10
20
30
40
50
60
−0.003
−0.002
−0.001
0.000
0.001
0.002
0.003
0.004
+9.45e1
Delay [ms]

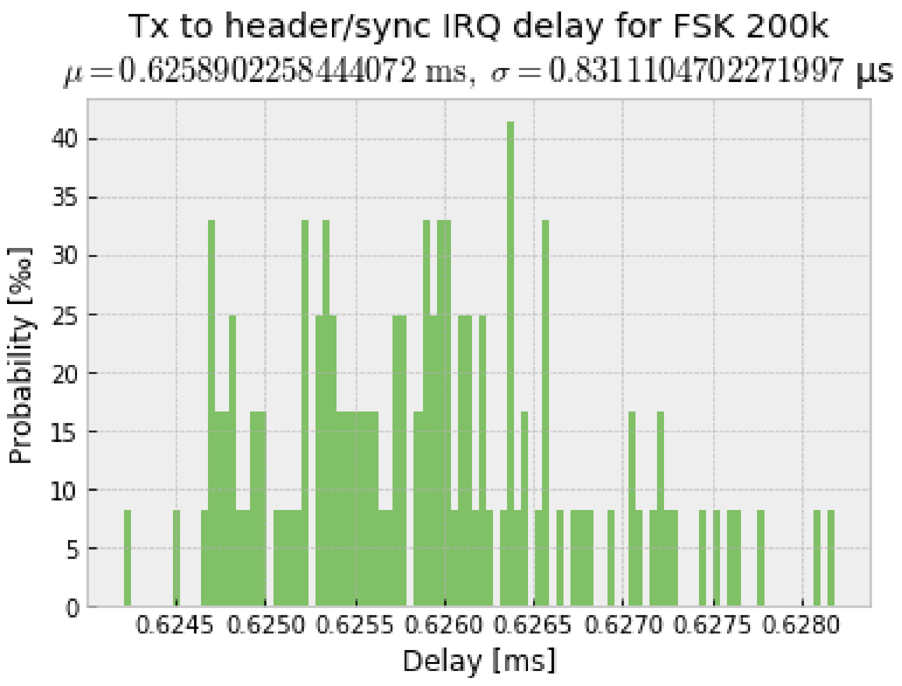

Tx to header/sync IRQ delay for SF8
$\mu = 47.283369278102434$ ms, $\sigma = 1.328632958644723$ µs
Probability [‰]
0
10
20
30
40
0.001
0.002
0.003
0.004
0.005
0.006
0.007
+4.728e1
Delay [ms]

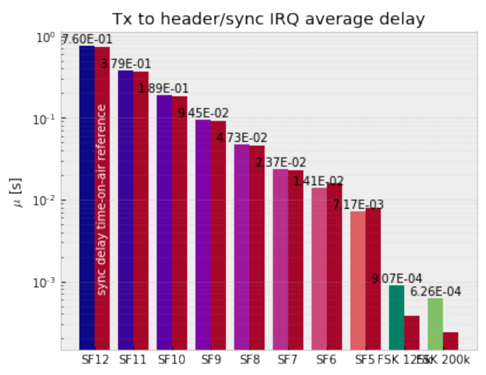

Tx to header/sync IRQ delay for SF7
$\mu = 23.739399033594093$ ms, $\sigma = 1.7225676422524603$ µs
Probability [‰]
0
10
20
30
40
50
60
70
80
23.737
23.738
23.739
23.740
23.741
23.742
23.743
Delay [ms]

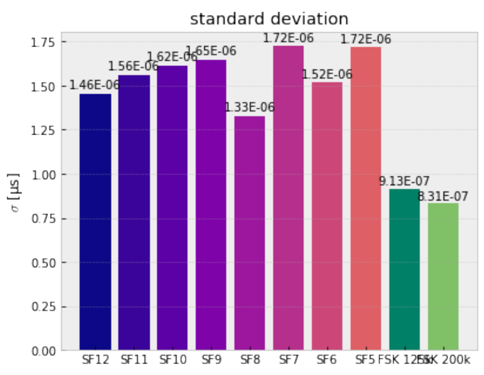

Tx to header/sync IRQ delay for SF6
$\mu = 14.052205210099949$ ms, $\sigma = 1.5160004936838551$ µs
Probability [‰]
0
10
20
30
40
50
60
0.000
0.001
0.002
0.003
0.004
0.005
+1.405e1
Delay [ms]

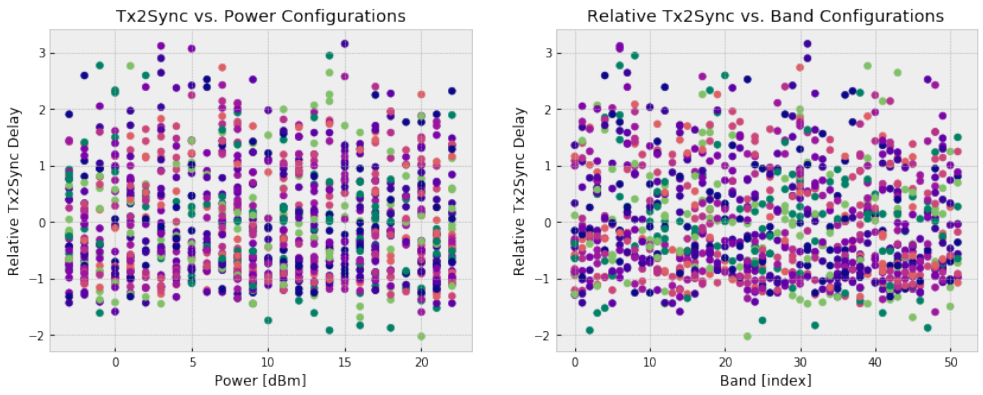

Tx to header/sync IRQ delay for SF5
$\mu = 7.1719960356324\,\mathrm{ms},\ \sigma = 1.715547559997702$ µs
Probability [‰]
Delay [ms]
0
10
20
30
40
50
60
70
7.170
7.171
7.172
7.173
7.174
7.175
7.176
7.177


Tx to header/sync IRQ delay for FSK 125k
$\mu = 0.9074013547153016\,\mathrm{ms},\ \sigma = 0.9128923713564809$ µs
Probability [‰]
Delay [ms]
0
10
20
30
40
50
0.906
0.907
0.908
0.909
0.910

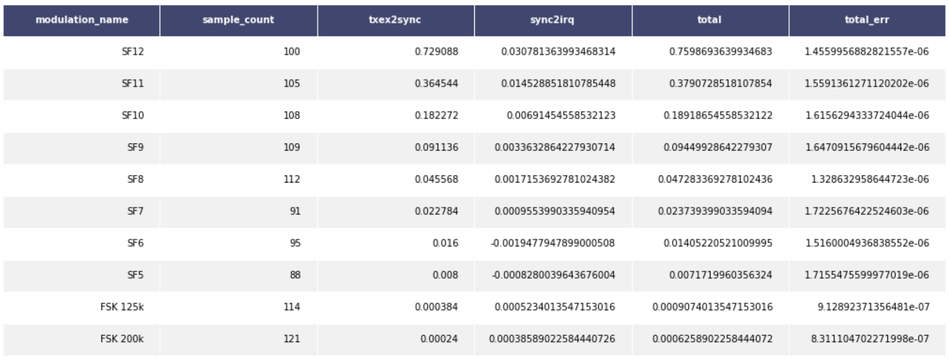

Tx to header/sync IRQ delay for FSK 200k
$\mu = 0.6258902258444072$ ms, $\sigma = 0.8311104702271997$ µs
Probability [‰]
Delay [ms]
0.6245 0.6250 0.6255 0.6260 0.6265 0.6270 0.6275 0.6280
0 5 10 15 20 25 30 35 40


Tx to header/sync IRQ average delay
μ [s]
sync delay time-on-air reference
7.60E-01
3.79E-01
1.89E-01
9.45E-02
4.73E-02
2.37E-02
1.41E-02
7.17E-03
9.07E-04
6.26E-04
SF12 SF11 SF10 SF9 SF8 SF7 SF6 SF5 FSK 125k FSK 200k


standard deviation
σ [µs]
1.46E-06
1.56E-06
1.62E-06
1.65E-06
1.33E-06
1.72E-06
1.52E-06
1.72E-06
9.13E-07
8.31E-07
SF12 SF11 SF10 SF9 SF8 SF7 SF6 SF5 FSK 125k FSK 200k



```
In [22]: delays.style.bar(subset='total', color='lightgray').set_precision(9)

Out[22]: <pandas.io.formats.style.Styler at 0x2194bbd1f98>

In [23]: render_mpl_table(delays, header_columns=0, col_width=3.0)

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x21948fec208>
```

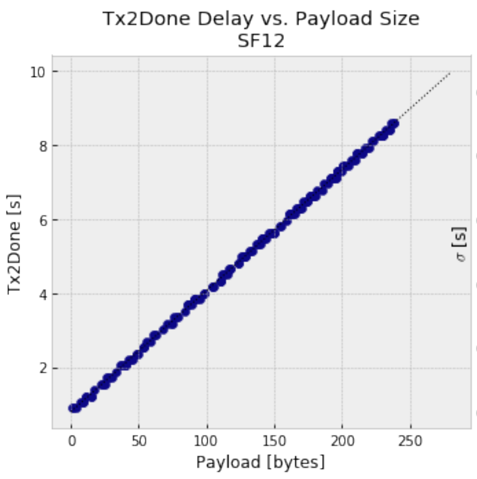


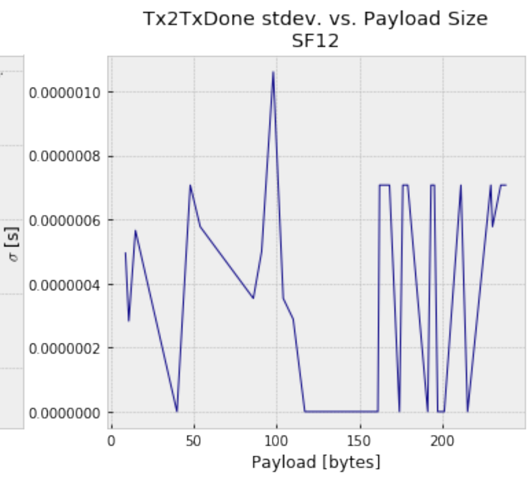
modulation_name	sample_count	txex2sync	sync2irq	total	total_err
SF12	100	0.729088	0.030781363993468314	0.7598693639934683	1.4559956882821557e-06
SF11	105	0.364544	0.014528851810785448	0.3790728518107854	1.5591361271120202e-06
SF10	108	0.182272	0.00691454558532123	0.18918654558532122	1.6156294333724044e-06
SF9	109	0.091136	0.0033632864227930714	0.09449928642279307	1.6470915679604442e-06
SF8	112	0.045568	0.0017153692781024382	0.047283369278102436	1.328632958644723e-06
SF7	91	0.022784	0.0009553990335940954	0.023739399033594094	1.7225676422524603e-06
SF6	95	0.016	-0.0019477947899000508	0.01405220521009995	1.5160004936838552e-06
SF5	88	0.008	-0.0008280039643676004	0.0071719960356324	1.7155475599977019e-06
FSK 125k	114	0.000384	0.0005234013547153016	0.0009074013547153016	9.12892371356481e-07
FSK 200k	121	0.00024	0.00038589022584440726	0.0006258902258444072	8.311104702271998e-07

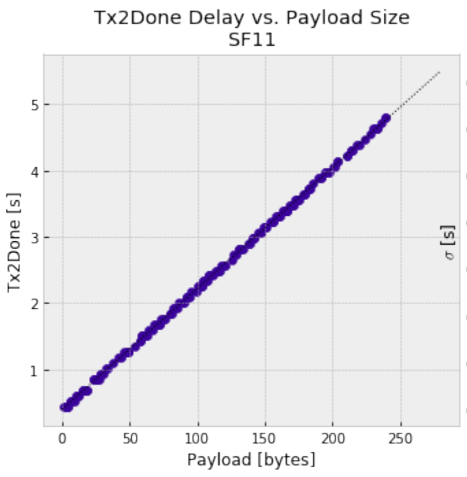
8 Tx 2 Tx Done

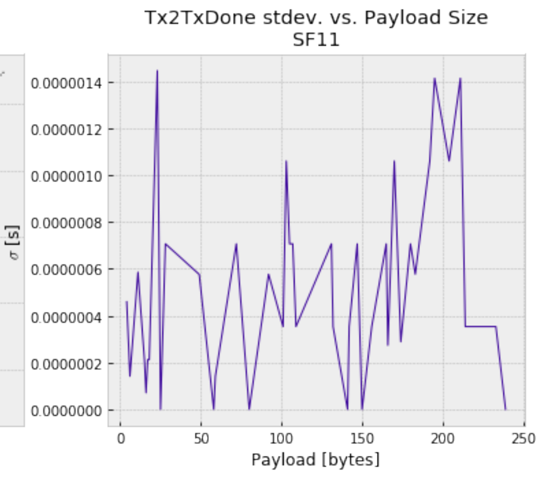
The measurements depict the the time from t_{ex} to t_{irq} when changing in Tx mode and finishing transmission for different payload lengths on a single node. For this experiment, NSS and DIO1 signals are required.

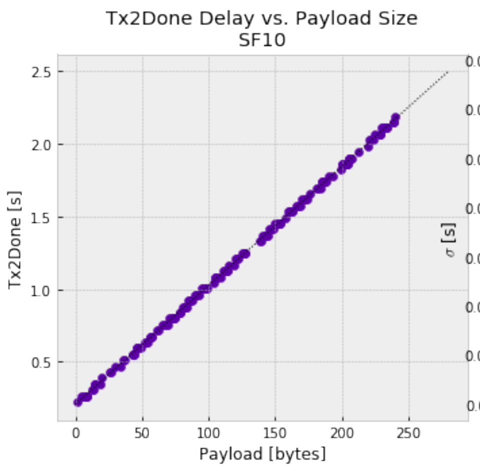
```
In [24]: %matplotlib inline

         from flora_tools.experiments.measure_time_tx2txdone import MeasureTimeTx2TxDone

         tx2txdone = MeasureTimeTx2TxDone()
         df = pd.read_csv("../../data/{}.csv".format(tx2txdone.name), index_col="sample")
         delays = tx2txdone.analyze(df)
```

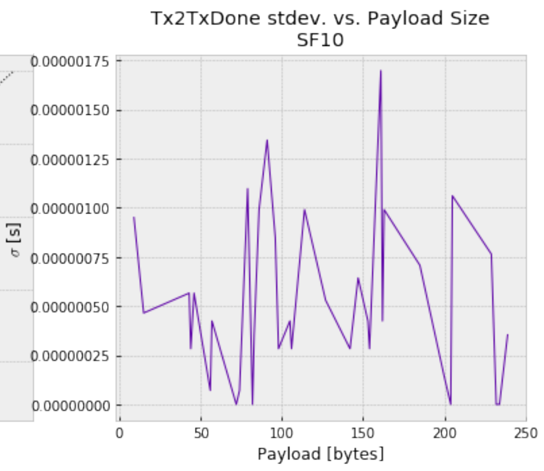


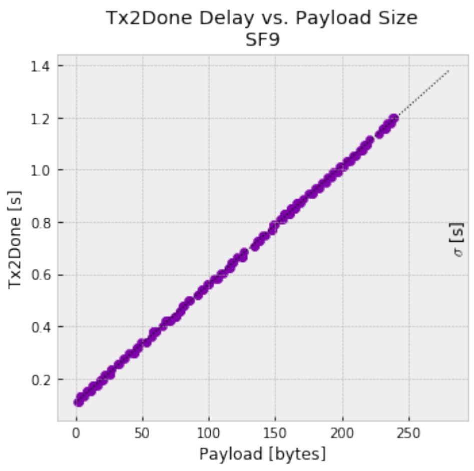


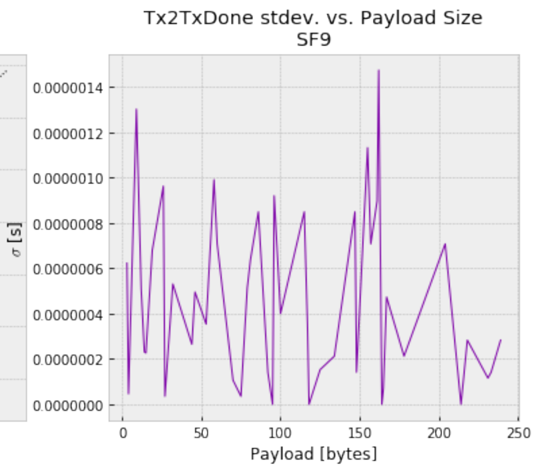


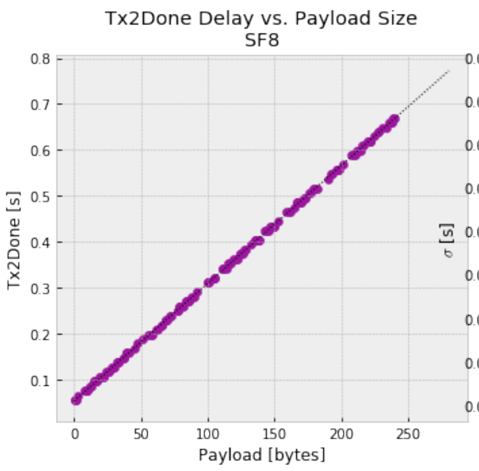


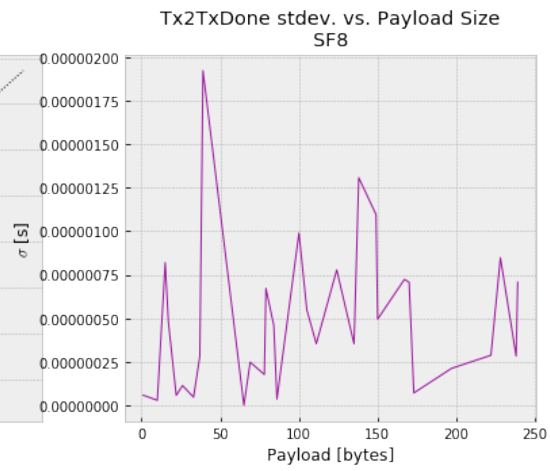

Tx2Done Delay vs. Payload Size
SF10
Tx2Done [s]
Payload [bytes]

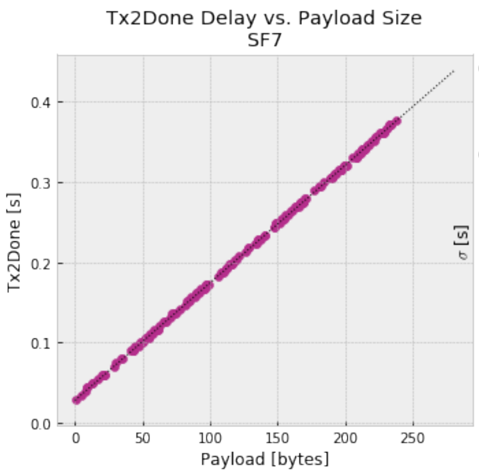

Tx2TxDone stdev. vs. Payload Size
SF10
σ [s]
Payload [bytes]

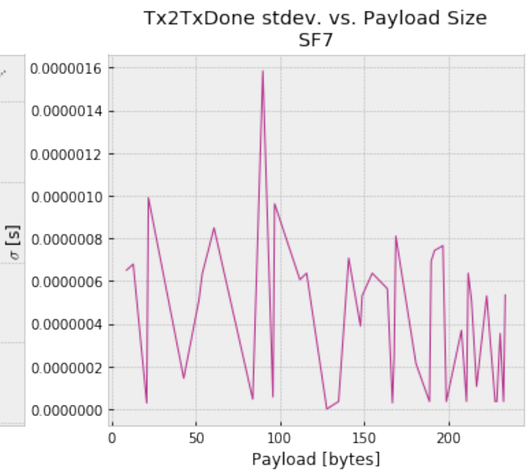

Tx2Done Delay vs. Payload Size
SF9
Tx2Done [s]
Payload [bytes]

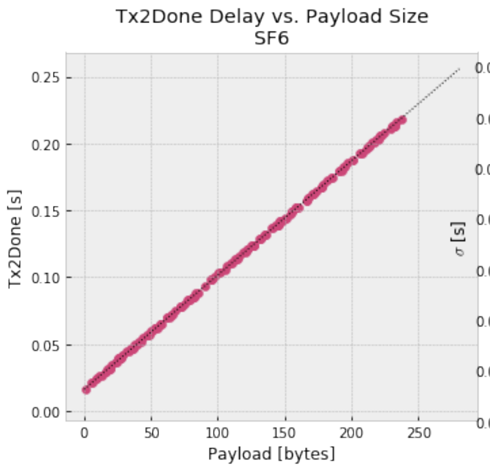

Tx2TxDone stdev. vs. Payload Size
SF9
σ [s]
Payload [bytes]

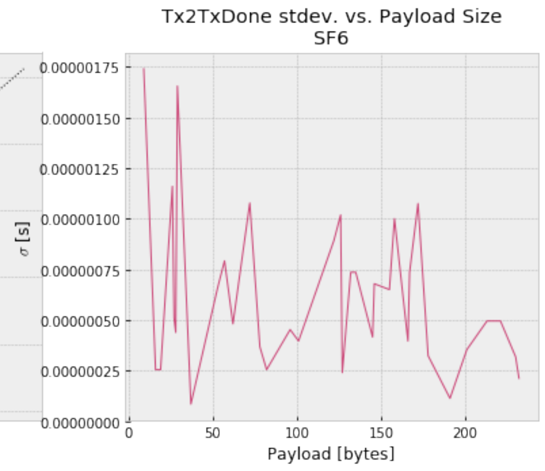

Tx2Done Delay vs. Payload Size
SF8
Tx2Done [s]
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250
Payload [bytes]

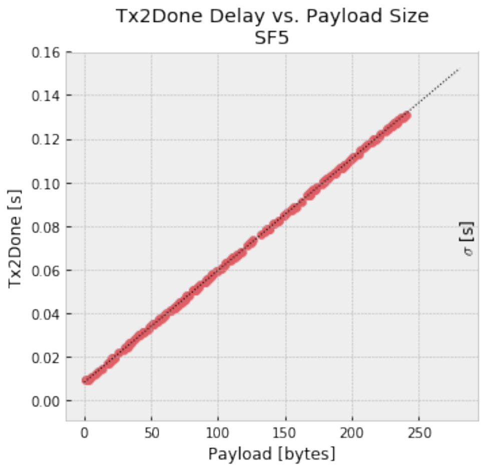

Tx2TxDone stdev. vs. Payload Size
SF8
σ [s]
0.00000000
0.00000025
0.00000050
0.00000075
0.00000100
0.00000125
0.00000150
0.00000175
0.00000200
0
50
100
150
200
250
Payload [bytes]

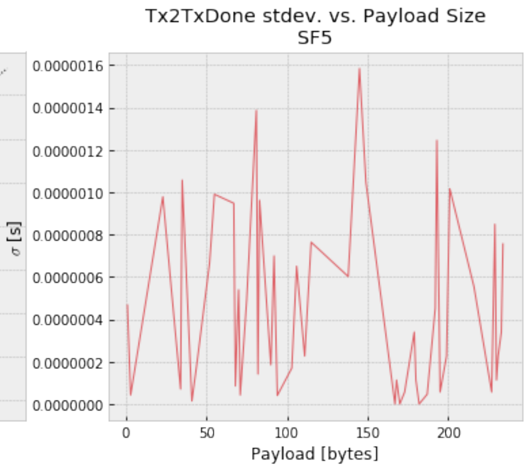

Tx2Done Delay vs. Payload Size
SF7
Tx2Done [s]
0.0
0.1
0.2
0.3
0.4
0
50
100
150
200
250
Payload [bytes]

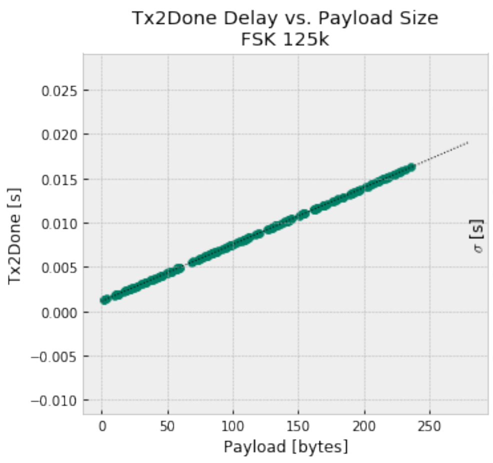

Tx2TxDone stdev. vs. Payload Size
SF7
σ [s]
0.0000000
0.0000002
0.0000004
0.0000006
0.0000008
0.0000010
0.0000012
0.0000014
0.0000016
0
50
100
150
200
Payload [bytes]

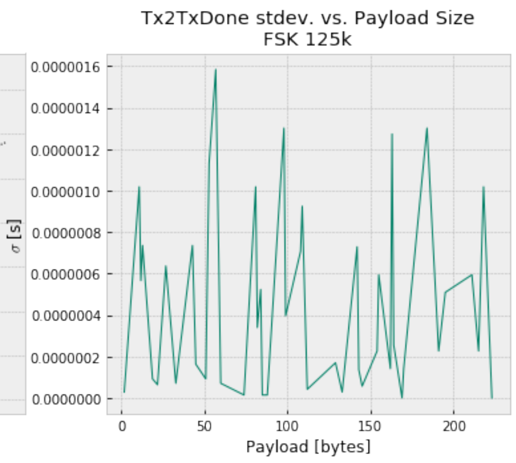

Tx2Done Delay vs. Payload Size
SF6
Tx2Done [s]
0.00
0.05
0.10
0.15
0.20
0.25
0
50
100
150
200
250
Payload [bytes]

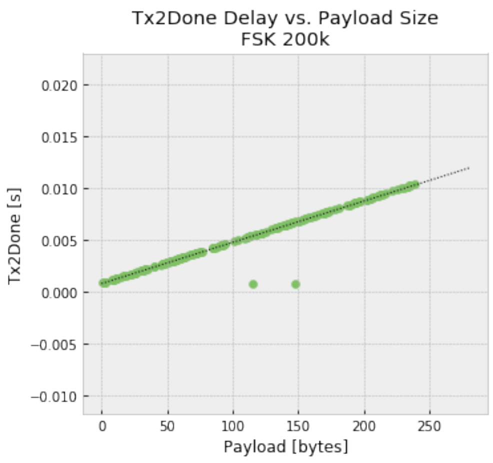

Tx2TxDone stdev. vs. Payload Size
SF6
σ [s]
0.00000000
0.00000025
0.00000050
0.00000075
0.00000100
0.00000125
0.00000150
0.00000175
0
50
100
150
200
Payload [bytes]

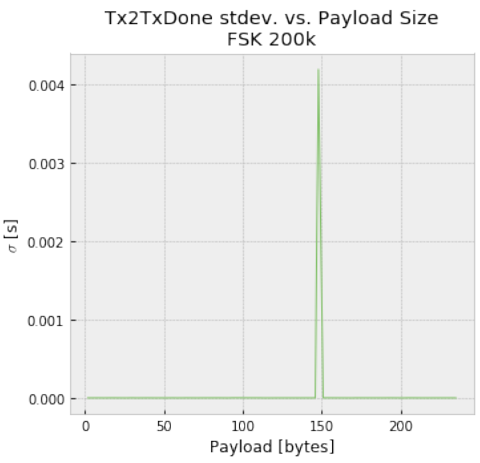

Tx2Done Delay vs. Payload Size
SF5
Tx2Done [s]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0
50
100
150
200
250
Payload [bytes]


Tx2TxDone stdev. vs. Payload Size
SF5
σ [s]
0.0000000
0.0000002
0.0000004
0.0000006
0.0000008
0.0000010
0.0000012
0.0000014
0.0000016
0
50
100
150
200
Payload [bytes]









```
In [25]: delays.style.set_precision(9)

Out[25]: <pandas.io.formats.style.Styler at 0x219490142e8>

In [26]: render_mpl_table(delays, header_columns=0, col_width=3.0)

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x21948f380b8>
```

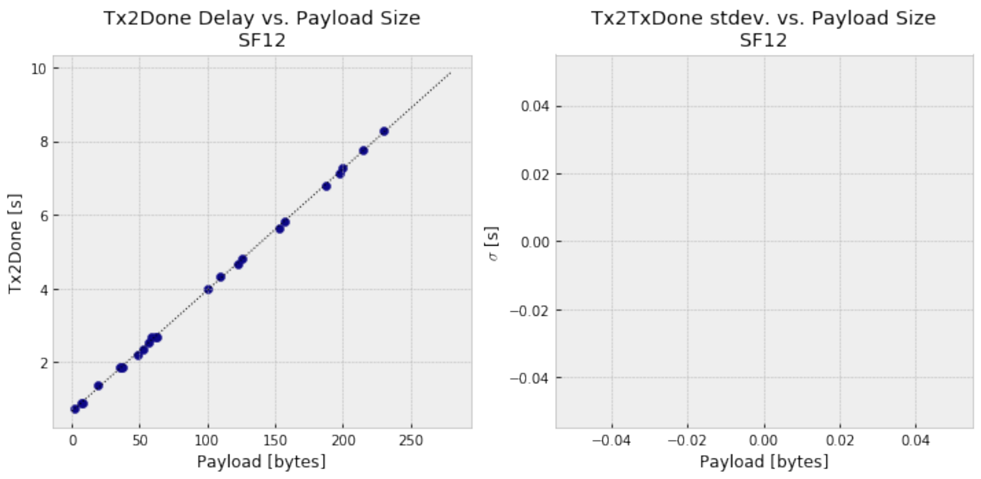
modulation_name	sample_count	offset	offset_err	m	b	fit_err
SF12	171	0.00015965547244141927	2.1934146802477286e-05	0.03274083902129352	0.7966077407722048	0.04741798212141947
SF11	182	0.00018539660341494597	1.2081904233801324e-05	0.018197642138599587	0.3995077428585395	0.023071513650039368
SF10	167	0.00019778579077071444	5.456367111552045e-06	0.008190770925146676	0.20645966694910461	0.011956680412277999
SF9	178	0.0002032071666072593	3.2209015528442857e-06	0.0045438581249091494	0.10573953025324803	0.006146405531816942
SF8	154	0.00020599794802310568	1.928248013849077e-06	0.0025605044878913496	0.0543421286042968	0.0027447257986775685
SF7	179	0.00020736585526492493	1.1601847377556683e-06	0.0014638742966620118	0.02797077027196037	0.0014976388285217095
SF6	168	-0.0005838649831839338	0.0011870724512460496	0.0008539473853982078	0.016214671614345467	0.000715076886270702
SF5	207	-0.00030438683908191937	0.0006287396394741577	0.0005125235364445632	0.008411608090449425	0.00036205383545861455
FSK 125k	180	0.0006286426984090606	5.990258907312815e-07	6.399859429095223e-05	0.001140804487710964	5.917182255111897e-07
FSK 200k	178	0.0004135825685370173	0.0005603187813445632	3.9700854760470685e-05	0.0007662465397476468	0.0005599312949472764

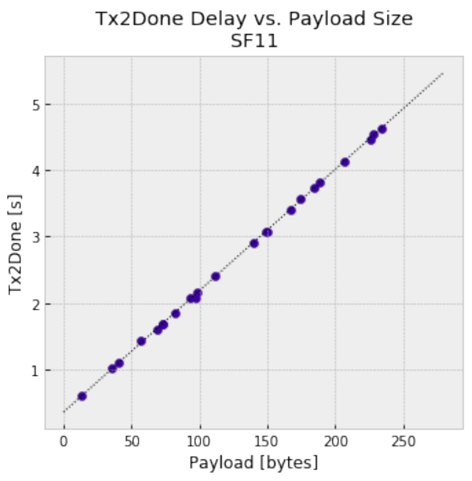
8.1 Tx 2 Tx Done Implicit

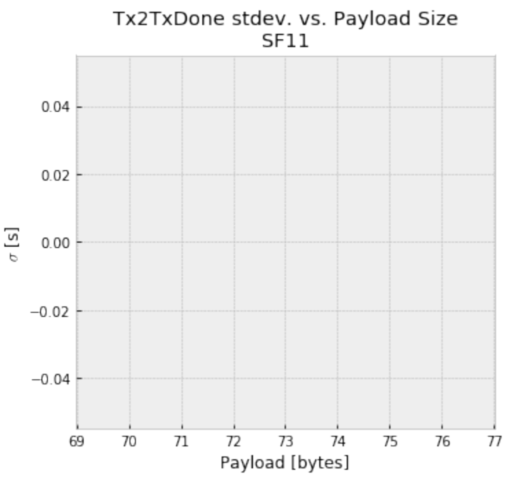
The measurements depict the the time from t_{ex} to t_{irq} when changing in Tx mode and finishing transmission on a single node. For this experiment, `NSS` and `DIO1` signals are required.

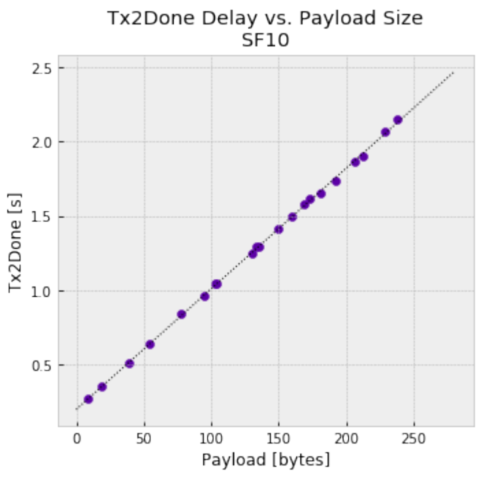
Implicit header mode is enabled (to isolate the impact of the packet engine's header logic).

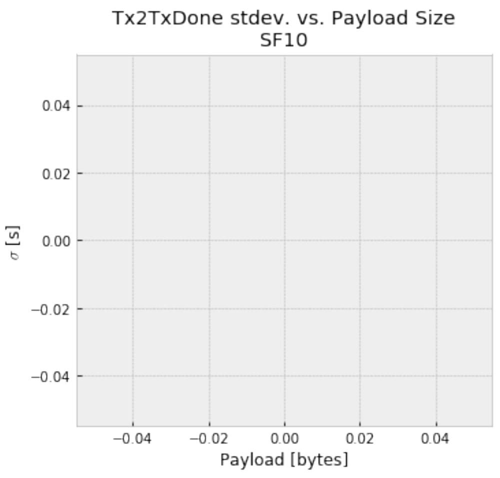
```
In [27]: %matplotlib inline

         from flora_tools.experiments.measure_time_tx2txdone_implicit import MeasureTimeTx2TxD

         tx2txdone_implicit = MeasureTimeTx2TxDoneImplicit()
         df = pd.read_csv("../../data/{}.csv".format(tx2txdone_implicit.name), index_col="samp
         delays = tx2txdone_implicit.analyze(df)
```

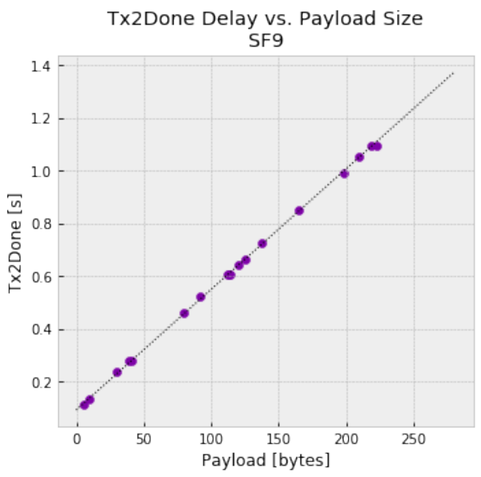


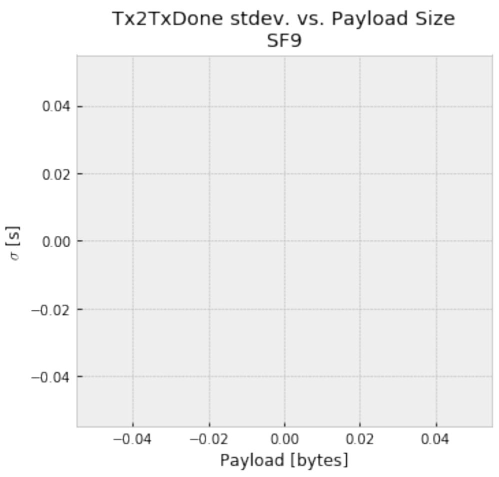

Tx2Done Delay vs. Payload Size
SF11
Tx2Done [s]
Payload [bytes]
0
50
100
150
200
250
1
2
3
4
5

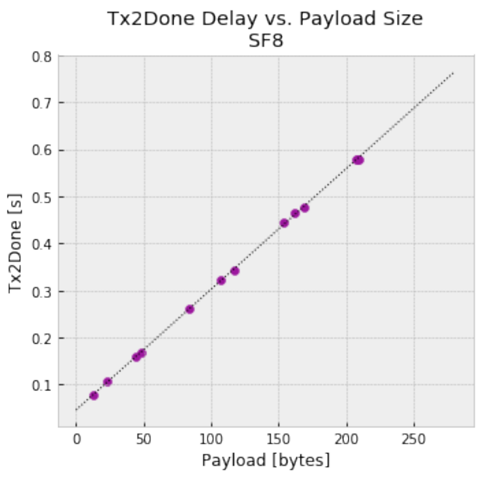

Tx2TxDone stdev. vs. Payload Size
SF11
σ [s]
Payload [bytes]
0.04
0.02
0.00
−0.02
−0.04
69
70
71
72
73
74
75
76
77

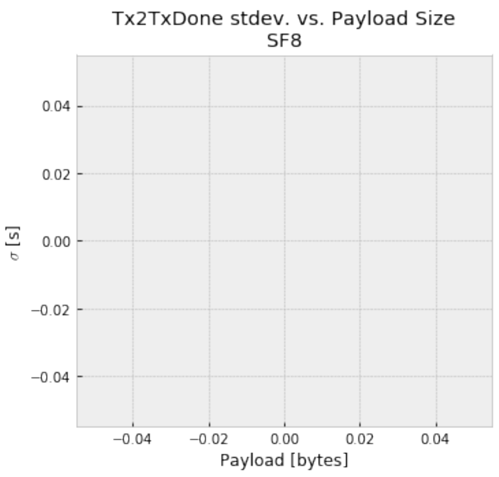

Tx2Done Delay vs. Payload Size
SF10
Tx2Done [s]
Payload [bytes]
0
50
100
150
200
250
0.5
1.0
1.5
2.0
2.5

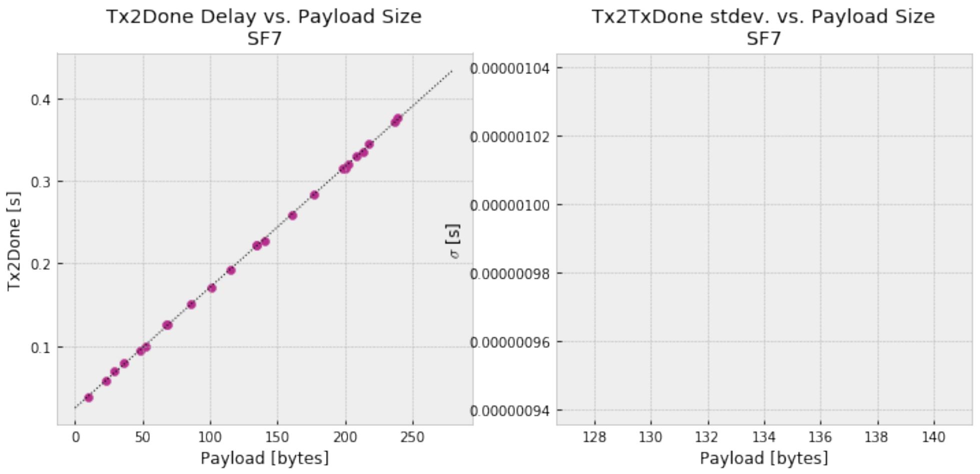

Tx2TxDone stdev. vs. Payload Size
SF10
σ [s]
Payload [bytes]
0.04
0.02
0.00
−0.02
−0.04
−0.04
−0.02
0.00
0.02
0.04

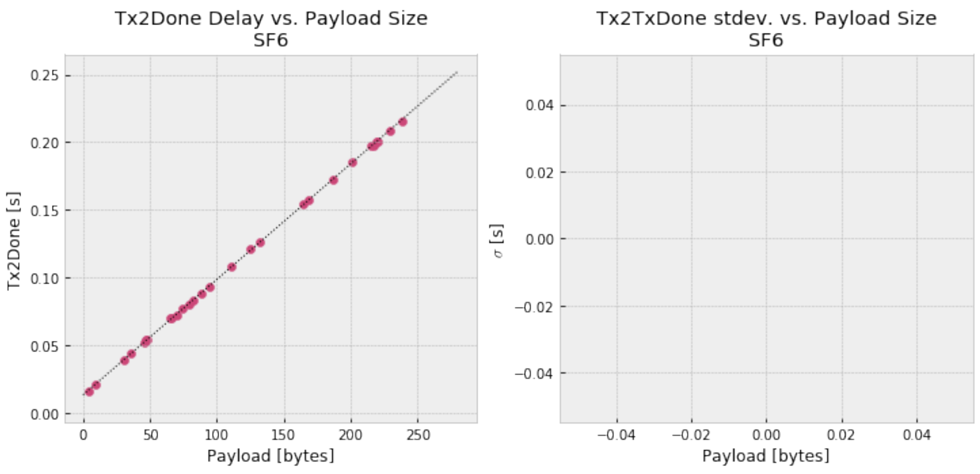

Tx2Done Delay vs. Payload Size
SF9
Tx2Done [s]
Payload [bytes]

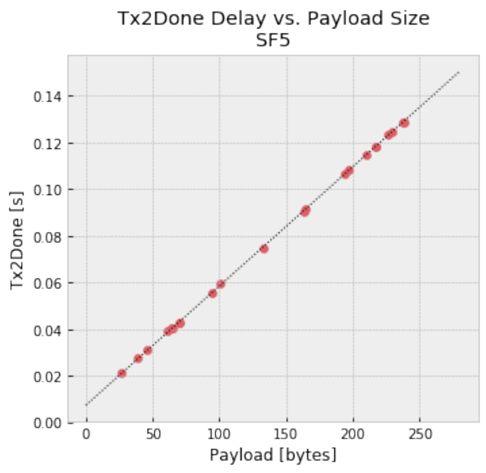

Tx2TxDone stdev. vs. Payload Size
SF9
σ [s]
Payload [bytes]

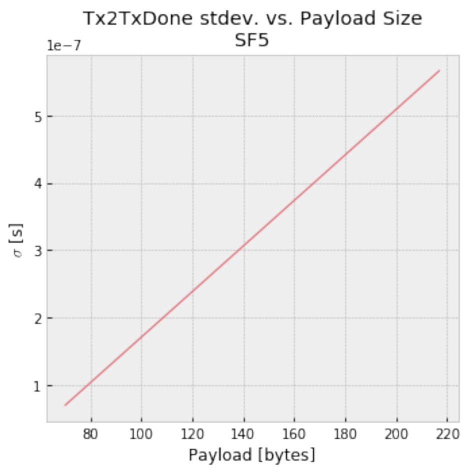

Tx2Done Delay vs. Payload Size
SF8
Tx2Done [s]
Payload [bytes]

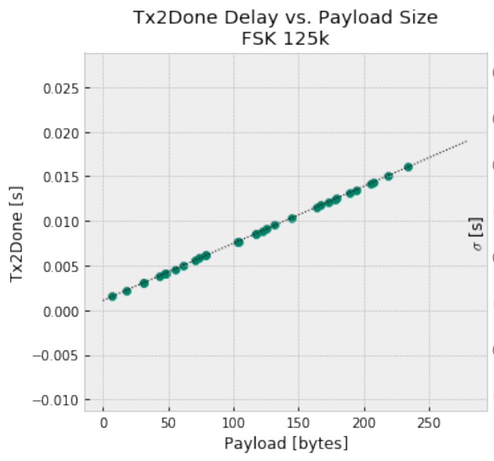

Tx2TxDone stdev. vs. Payload Size
SF8
σ [s]
Payload [bytes]

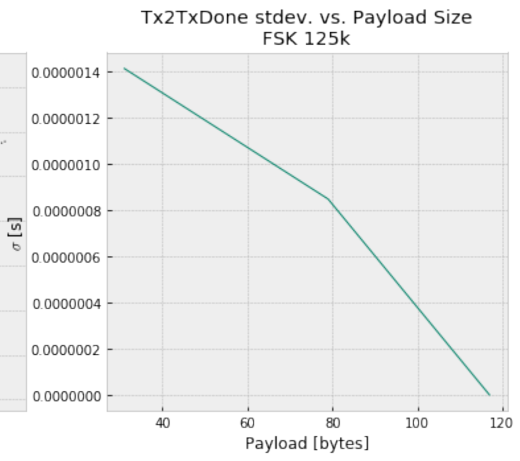

Tx2Done Delay vs. Payload Size
SF7
Tx2Done [s]
Payload [bytes]
Tx2TxDone stdev. vs. Payload Size
SF7
σ [s]
Payload [bytes]

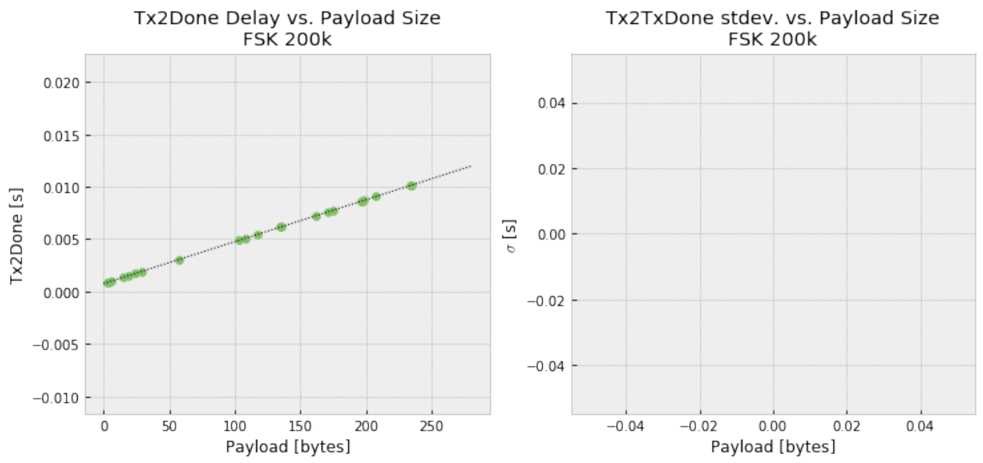

Tx2Done Delay vs. Payload Size
SF6
Tx2Done [s]
Payload [bytes]
Tx2TxDone stdev. vs. Payload Size
SF6
σ [s]
Payload [bytes]


Tx2Done Delay vs. Payload Size
SF5
Tx2Done [s]
Payload [bytes]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
50
100
150
200
250


Tx2TxDone stdev. vs. Payload Size
SF5
1e−7
σ [s]
Payload [bytes]
1
2
3
4
5
80
100
120
140
160
180
200
220


Tx2Done Delay vs. Payload Size
FSK 125k
Tx2Done [s]
Payload [bytes]
−0.010
−0.005
0.000
0.005
0.010
0.015
0.020
0.025
0
50
100
150
200
250


Tx2TxDone stdev. vs. Payload Size
FSK 125k
σ [s]
Payload [bytes]
0.0000000
0.0000002
0.0000004
0.0000006
0.0000008
0.0000010
0.0000012
0.0000014
40
60
80
100
120



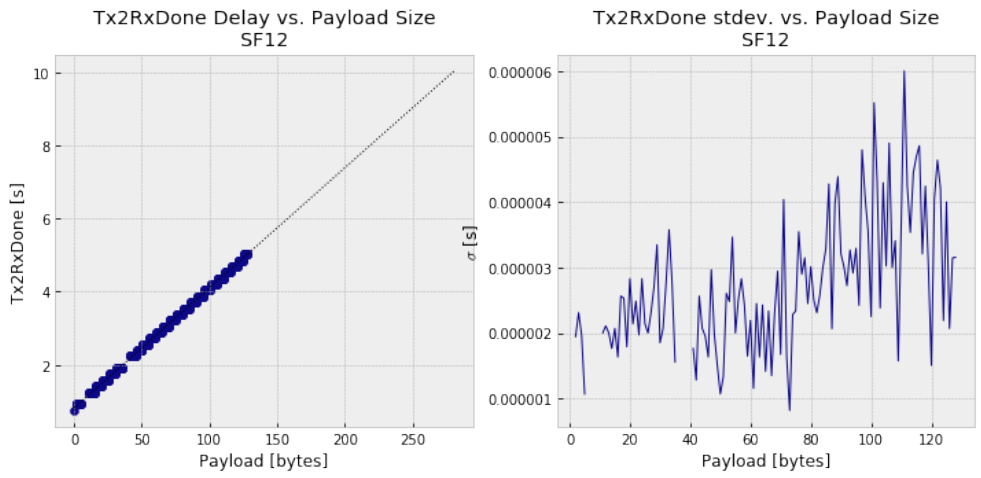
```
In [28]: delays.style.set_precision(9)

Out[28]: <pandas.io.formats.style.Styler at 0x2194a69b278>

In [29]: render_mpl_table(delays, header_columns=0, col_width=3.0)

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x2194a789e48>
```

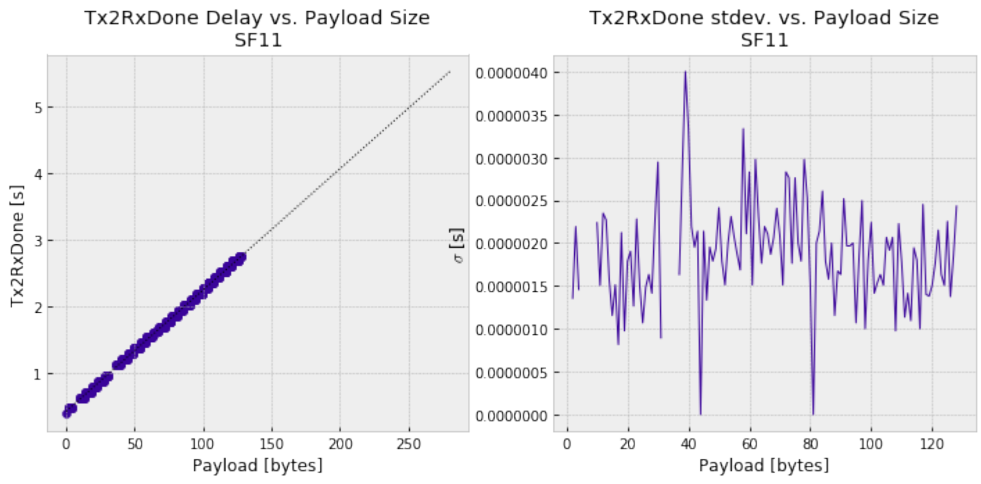
modulation_name	sample_count	offset	offset_err	m	b	fit_err
SF12	24	-0.10904986577510316	0.07889025753104796	0.03285026900213505	0.6750142008550453	0.04687423765807106
SF11	23	-0.04255349928233635	0.0418400386455385	0.018227625455403983	0.35864961191402456	0.019653935567751787
SF10	21	-0.01735443971830838	0.020772118933983064	0.008138604782354448	0.19390513963247138	0.009584118219772822
SF9	18	-0.008896278682169304	0.010471268919846349	0.004576980372209381	0.09145710265353885	0.006415321373568383 6
SF8	13	-0.008456190751588238	0.0038447858772002338	0.0025643037825555074	0.04585570586008411	0.002777862513692482
SF7	24	-0.0032037587486515528	0.0024653558921697615	0.0014623167208250088	0.024952857871520247	0.001362038177984718
SF6	28	-0.0035380228220045604	0.0013002614552888533	0.0008528838507919366	0.013489628148149904	0.0006813819983636348
SF5	22	-0.0015345629095678548	0.0006303199723987404	0.0005113653742539391	0.007298521574985943	0.0003403238279962411 4
FSK 125k	32	0.0005668707591396352	6.279234899748021e-07	6.399722592899622e-05	0.001079193244893825	6.009271861875342e-07
FSK 200k	22	0.00043463248639523513	5.288635674713368e-07	4.000004715401512e-05	0.0007546270572534034	5.288493272559411e-07

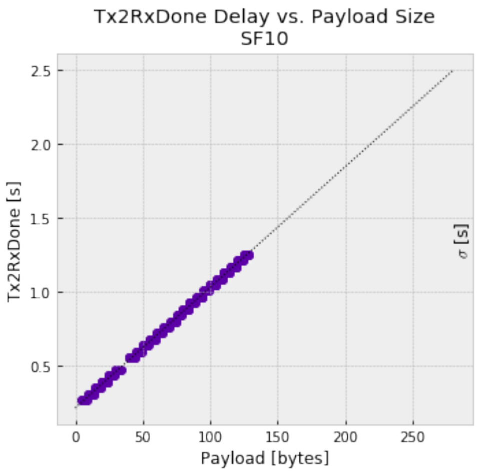
9 Tx 2 Rx Done

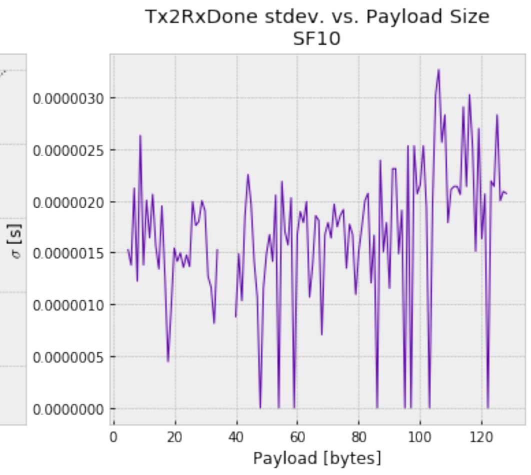
The measurements depict the the time from t_{ex} to t_{irq} when transmitting a message from DevKit B and Rx completion on DevKit A. For this experiment, `NSS` on DevKit B and `DIO1` on DevKit B are required.

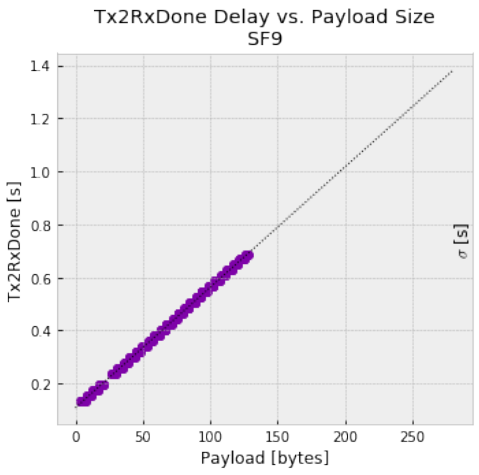
```
In [30]: %matplotlib inline

         from flora_tools.experiments.measure_time_tx2rxdone import MeasureTimeTx2RxDone

         tx2rxdone = MeasureTimeTx2RxDone()
         df = pd.read_csv("../../data/{}.csv".format(tx2rxdone.name), index_col="sample")
         delays = tx2rxdone.analyze(df)
```

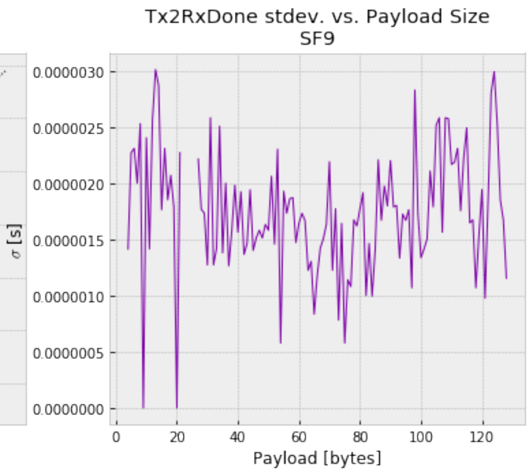


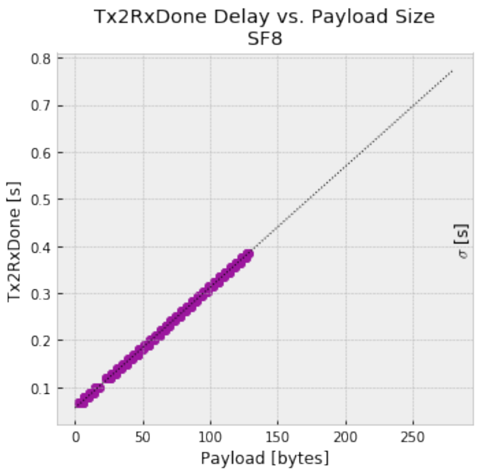


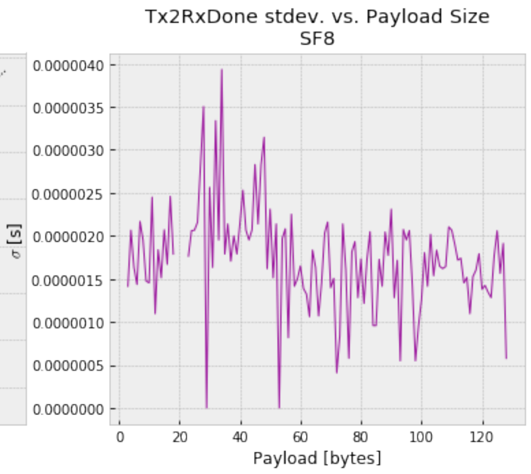

Tx2RxDone Delay vs. Payload Size
SF10
Tx2RxDone [s]
Payload [bytes]

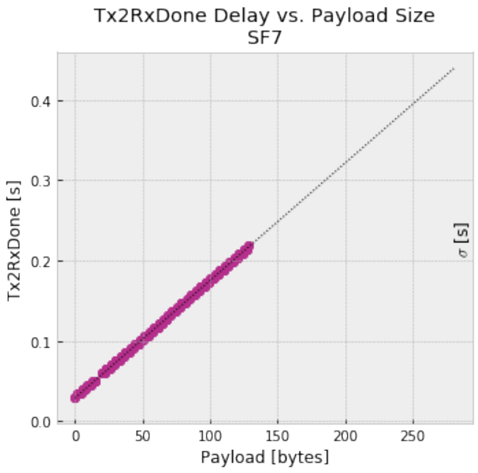

Tx2RxDone stdev. vs. Payload Size
SF10
σ [s]
Payload [bytes]

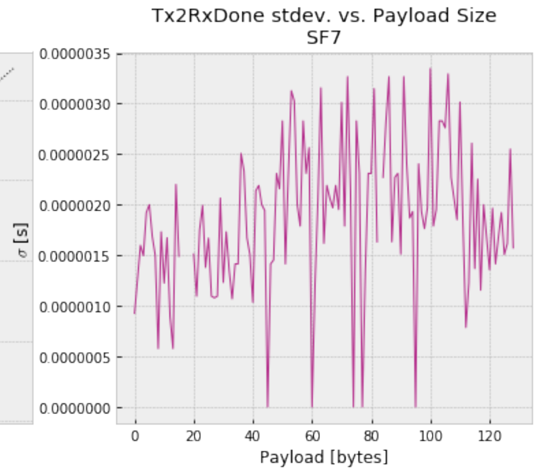

Tx2RxDone Delay vs. Payload Size
SF9
Tx2RxDone [s]
Payload [bytes]

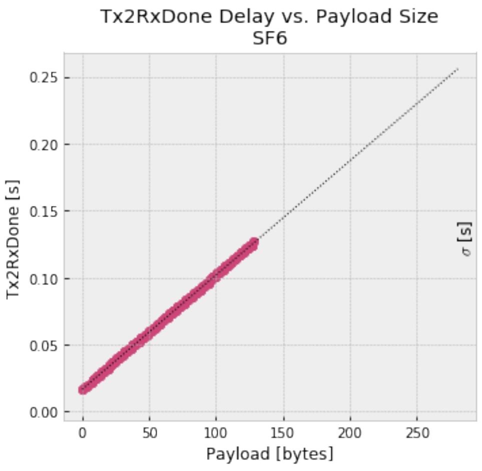
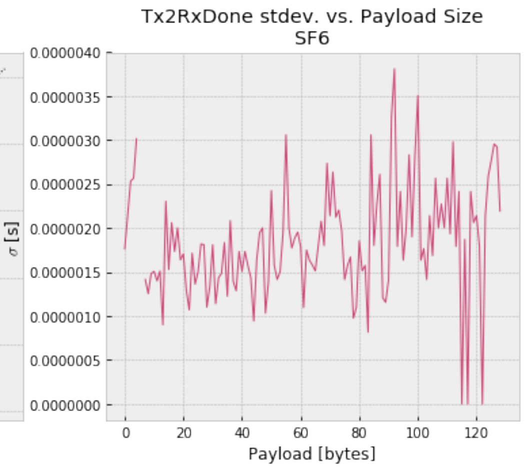
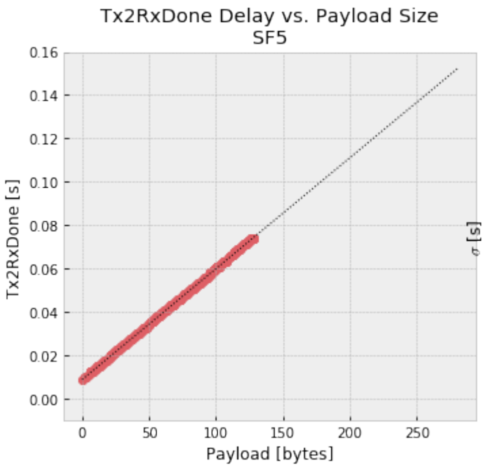
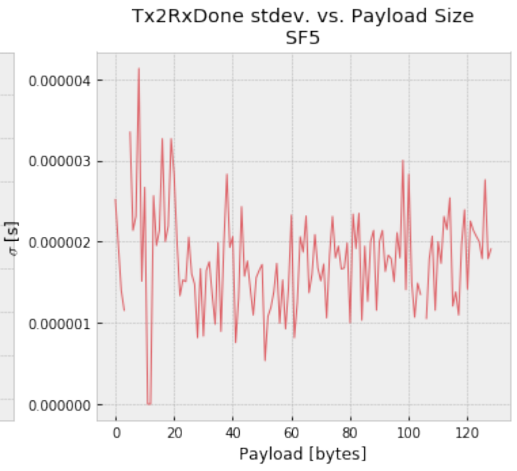

Tx2RxDone stdev. vs. Payload Size
SF9
σ [s]
Payload [bytes]

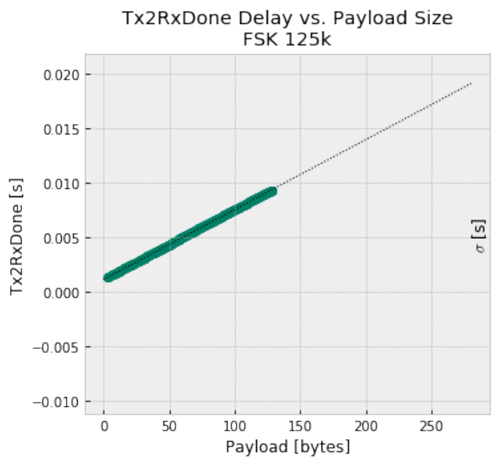

Tx2RxDone Delay vs. Payload Size
SF8
Tx2RxDone [s]
Payload [bytes]
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250

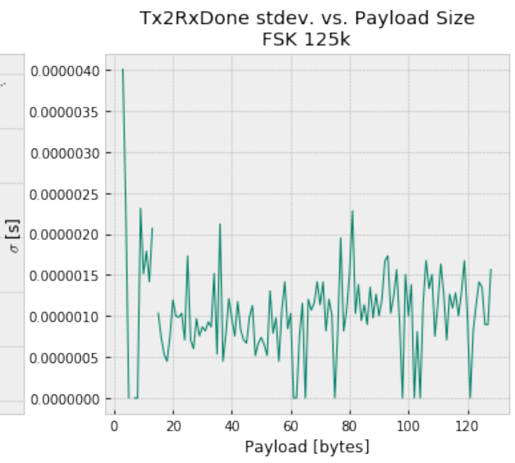

Tx2RxDone stdev. vs. Payload Size
SF8
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
0.0000040
0
20
40
60
80
100
120

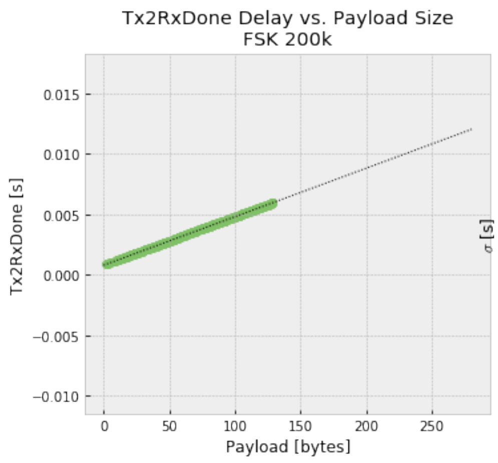

Tx2RxDone Delay vs. Payload Size
SF7
Tx2RxDone [s]
Payload [bytes]
0.0
0.1
0.2
0.3
0.4
0
50
100
150
200
250

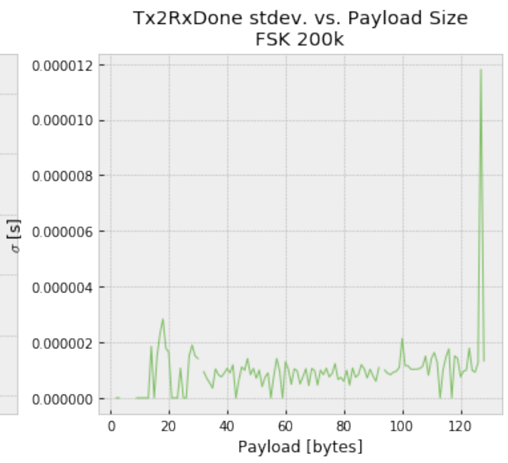

Tx2RxDone stdev. vs. Payload Size
SF7
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
0
20
40
60
80
100
120


Tx2RxDone Delay vs. Payload Size
SF6
Tx2RxDone [s]
Payload [bytes]
0.00
0.05
0.10
0.15
0.20
0.25
0
50
100
150
200
250

Tx2RxDone stdev. vs. Payload Size
SF6
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
0.0000040
0
20
40
60
80
100
120

Tx2RxDone Delay vs. Payload Size
SF5
Tx2RxDone [s]
Payload [bytes]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0
50
100
150
200
250

Tx2RxDone stdev. vs. Payload Size
SF5
σ [s]
Payload [bytes]
0.000000
0.000001
0.000002
0.000003
0.000004
0
20
40
60
80
100
120

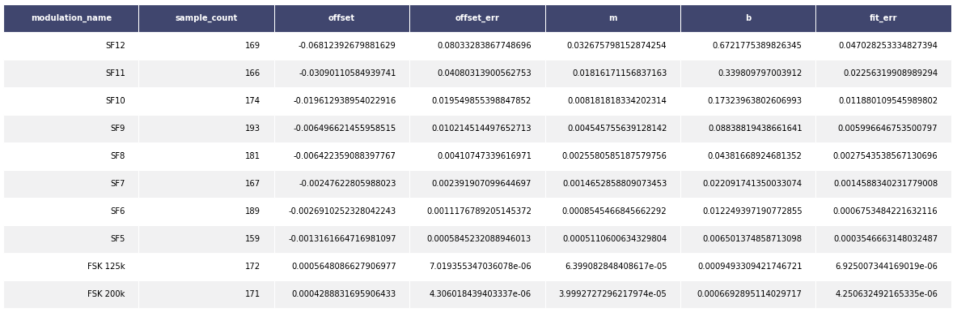








```
In [31]: delays.style.set_precision(9)

Out[31]: <pandas.io.formats.style.Styler at 0x2194ac3d0b8>

In [78]: render_mpl_table(delays, header_columns=0, col_width=3.0)

Out[78]: <matplotlib.axes._subplots.AxesSubplot at 0x2194f6cfd30>
```



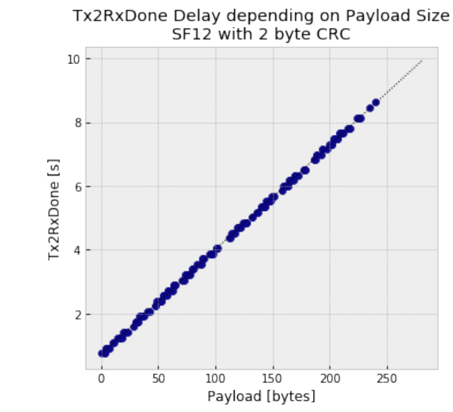
modulation_name	sample_count	offset	offset_err	m	b	fit_err
SF12	169	-0.06812392679881629	0.08033283867748696	0.032675798152874254	0.6721775389826345	0.047028253334827394
SF11	166	-0.03090110584939741	0.04080313900562753	0.01816171156837163	0.339809797003912	0.02256319908989294
SF10	174	-0.019612938954022916	0.019549855398847852	0.008181818334202314	0.17323963802606993	0.011880109545989802
SF9	193	-0.006496621455958515	0.010214514497652713	0.004545755639128142	0.08838819438661641	0.005996646753500797
SF8	181	-0.006422359088397767	0.00410747339616971	0.0025580585187579756	0.04381668924681352	0.0027543538567130696
SF7	167	-0.00247622805988023	0.002391907099644697	0.0014652858809073453	0.022091741350033074	0.0014588340231779008
SF6	189	-0.0026910252328042243	0.0011176789205145372	0.0008545466845662292	0.012249397190772855	0.0006753484221632116
SF5	159	-0.0013161664716981097	0.0005845232088946013	0.0005110600634329804	0.006501374858713098	0.0003546663148032487
FSK 125k	172	0.0005648086627906977	7.019355347036078e-06	6.399082848408617e-05	0.0009493309421746721	6.925007344169019e-06
FSK 200k	171	0.0004288831695906433	4.306018439403337e-06	3.9992727296217974e-05	0.0006692895114029717	4.250632492165335e-06

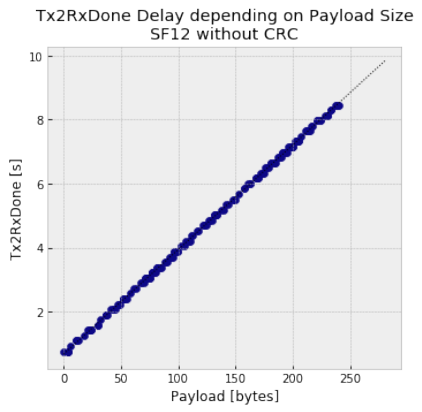
10 Tx 2 Rx Done Implicit

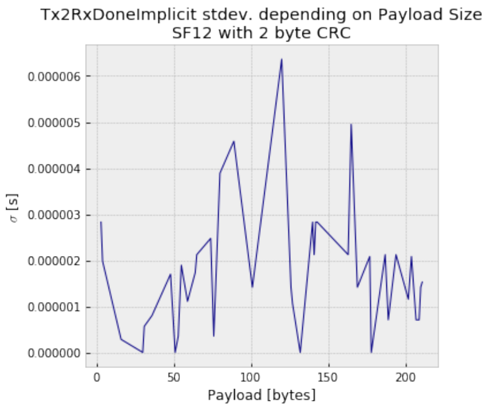
The measurements depict the the time from t_{ex} to t_{irq} when transmitting a message from DevKit B and Rx completion on DevKit A. For this experiment, `NSS` on DevKit B and `DIO1` on DevKit B are required.

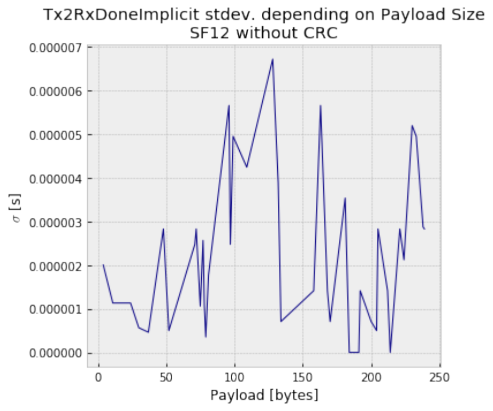
Implicit header mode is enabled (to isolate the impact of the packet engine's header logic).

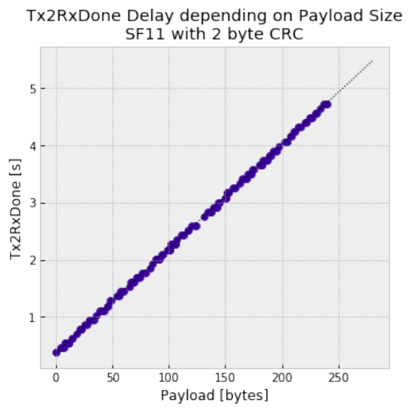
```
In [32]: %matplotlib inline

         from flora_tools.experiments.measure_time_tx2rxdone_implicit import MeasureTimeTx2RxD

         tx2rxdone_implicit = MeasureTimeTx2RxDoneImplicit()
         df = pd.read_csv("../../data/{}.csv".format(tx2rxdone_implicit.name), index_col="samp
         delays = tx2rxdone_implicit.analyze(df)
```

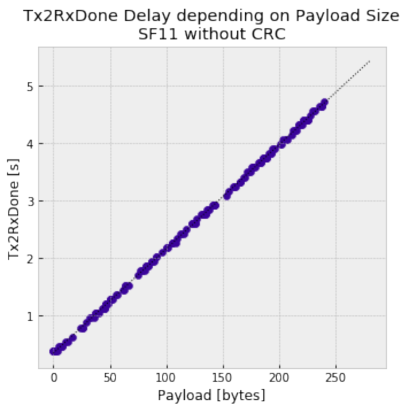

Tx2RxDone Delay depending on Payload Size
SF12 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]

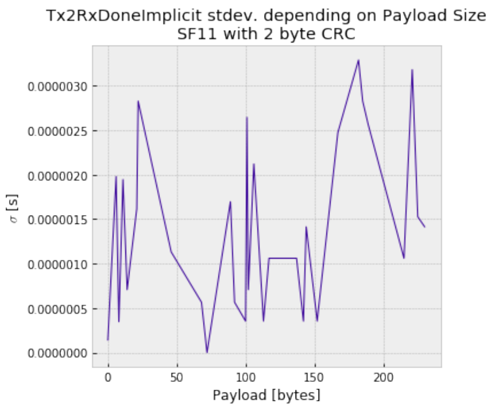

Tx2RxDone Delay depending on Payload Size
SF12 without CRC
Tx2RxDone [s]
Payload [bytes]

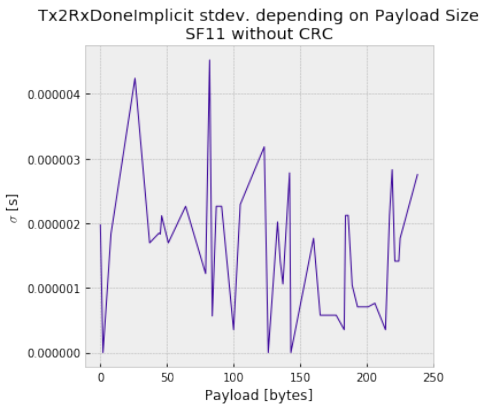

Tx2RxDoneImplicit stdev. depending on Payload Size
SF12 with 2 byte CRC
σ [s]
Payload [bytes]

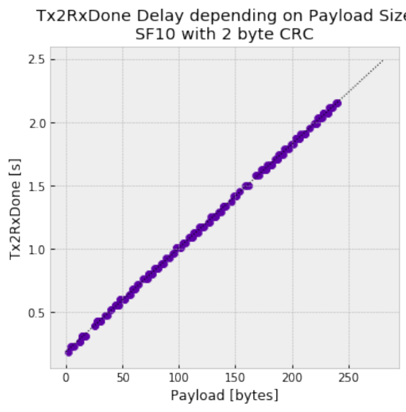

Tx2RxDoneImplicit stdev. depending on Payload Size
SF12 without CRC
σ [s]
Payload [bytes]

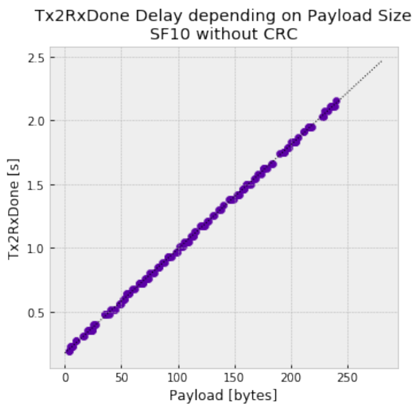

Tx2RxDone Delay depending on Payload Size
SF11 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]
1
2
3
4
5
0
50
100
150
200
250

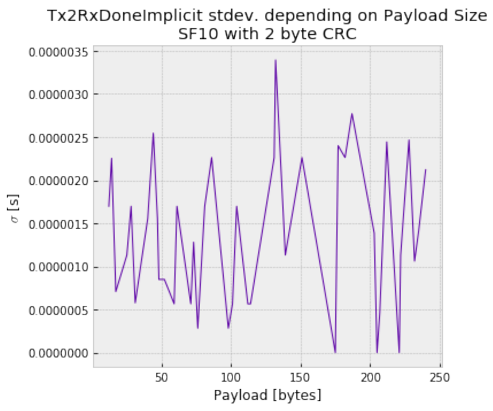

Tx2RxDone Delay depending on Payload Size
SF11 without CRC
Tx2RxDone [s]
Payload [bytes]
1
2
3
4
5
0
50
100
150
200
250

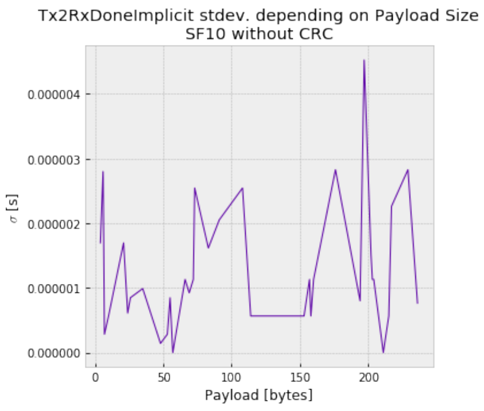

Tx2RxDoneImplicit stdev. depending on Payload Size
SF11 with 2 byte CRC
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0
50
100
150
200

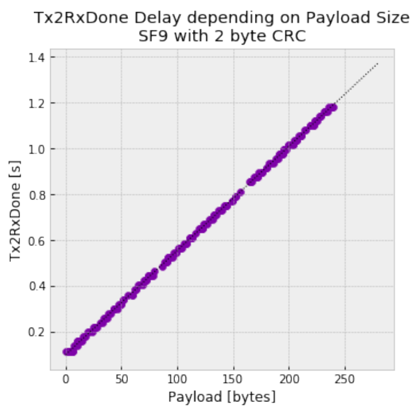

Tx2RxDoneImplicit stdev. depending on Payload Size
SF11 without CRC
σ [s]
Payload [bytes]
0.000000
0.000001
0.000002
0.000003
0.000004
0
50
100
150
200
250

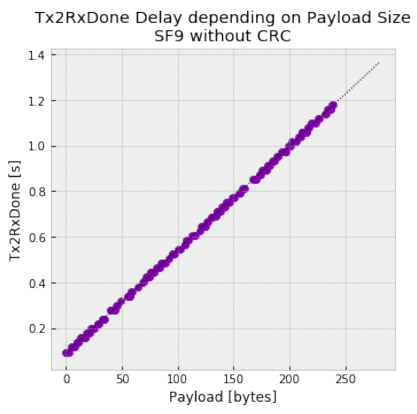

Tx2RxDone Delay depending on Payload Size
SF10 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250

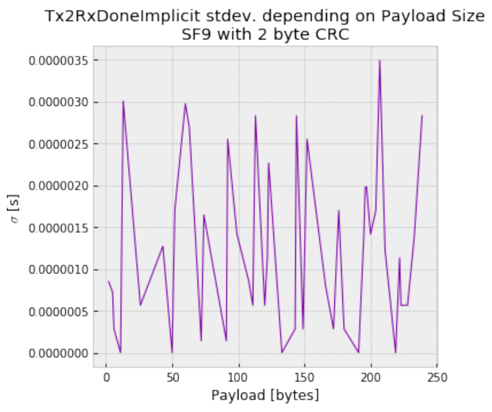

Tx2RxDone Delay depending on Payload Size
SF10 without CRC
Tx2RxDone [s]
Payload [bytes]
0.5
1.0
1.5
2.0
2.5
0
50
100
150
200
250

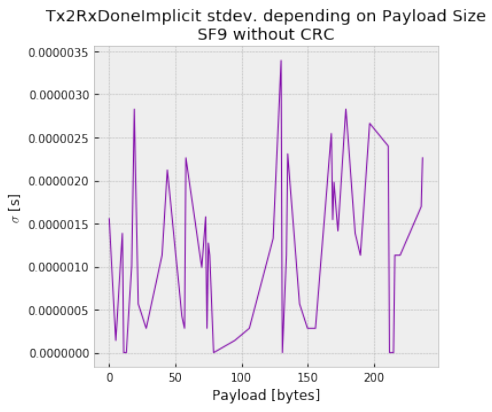

Tx2RxDoneImplicit stdev. depending on Payload Size
SF10 with 2 byte CRC
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
50
100
150
200
250

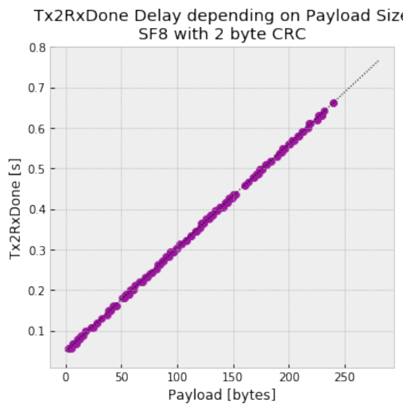

Tx2RxDoneImplicit stdev. depending on Payload Size
SF10 without CRC
σ [s]
Payload [bytes]
0.000000
0.000001
0.000002
0.000003
0.000004
0
50
100
150
200


Tx2RxDone Delay depending on Payload Size
SF9 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]

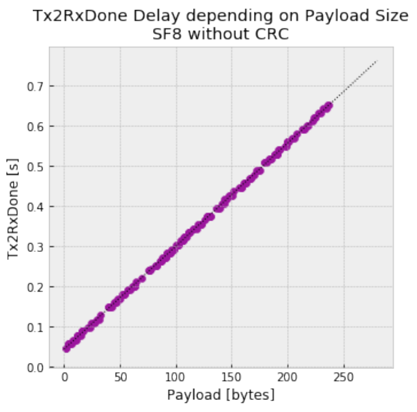

Tx2RxDone Delay depending on Payload Size
SF9 without CRC
Tx2RxDone [s]
Payload [bytes]


Tx2RxDoneImplicit stdev. depending on Payload Size
SF9 with 2 byte CRC
σ [s]
Payload [bytes]

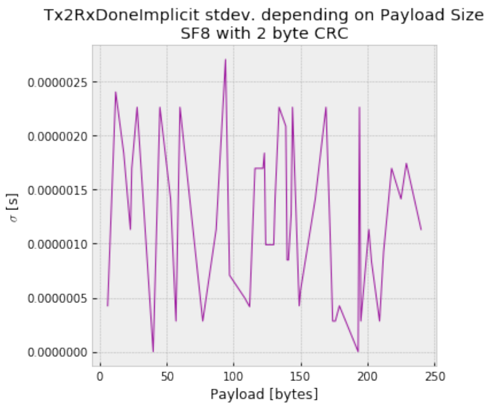

Tx2RxDoneImplicit stdev. depending on Payload Size
SF9 without CRC
σ [s]
Payload [bytes]



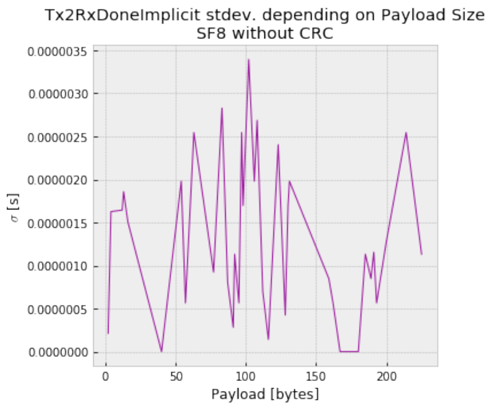
Tx2RxDone Delay depending on Payload Size
SF8 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]



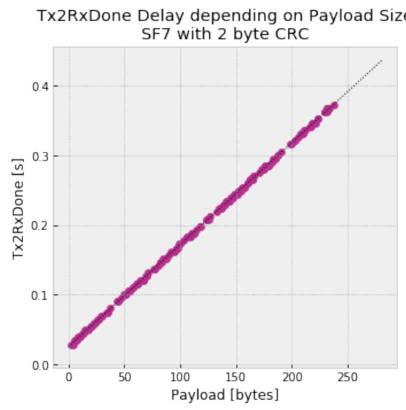
Tx2RxDone Delay depending on Payload Size
SF8 without CRC
Tx2RxDone [s]
Payload [bytes]

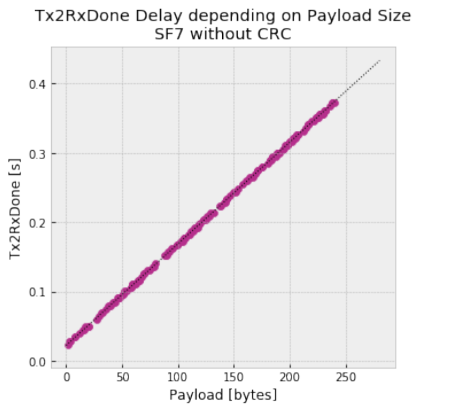


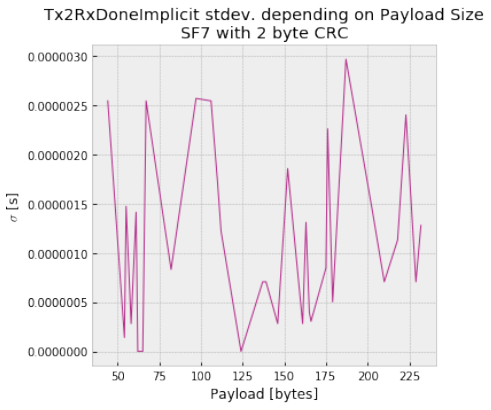
Tx2RxDoneImplicit stdev. depending on Payload Size
SF8 with 2 byte CRC
σ [s]
Payload [bytes]

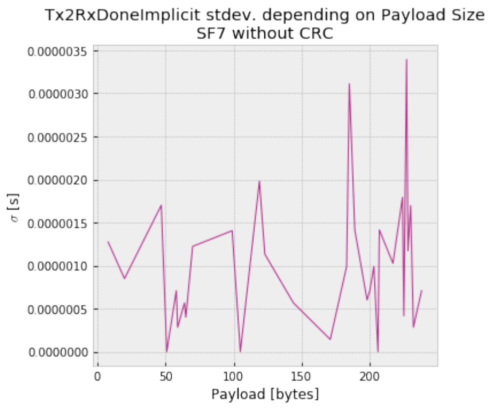


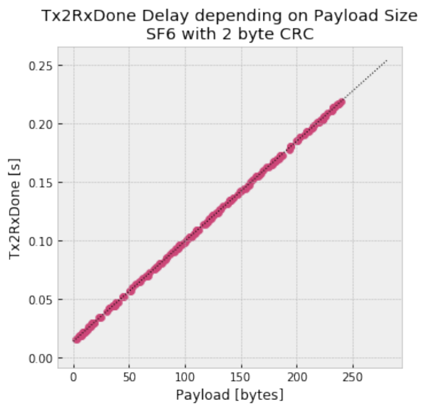
Tx2RxDoneImplicit stdev. depending on Payload Size
SF8 without CRC
σ [s]
Payload [bytes]

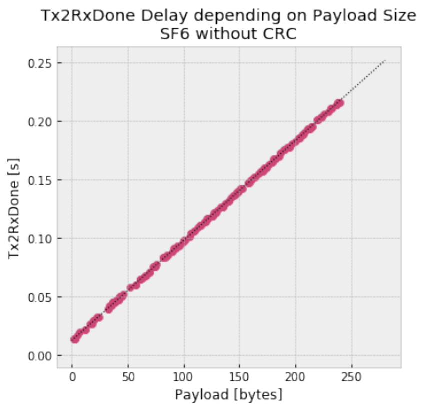

Tx2RxDone Delay depending on Payload Size
SF7 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]
0.0
0.1
0.2
0.3
0.4
0
50
100
150
200
250

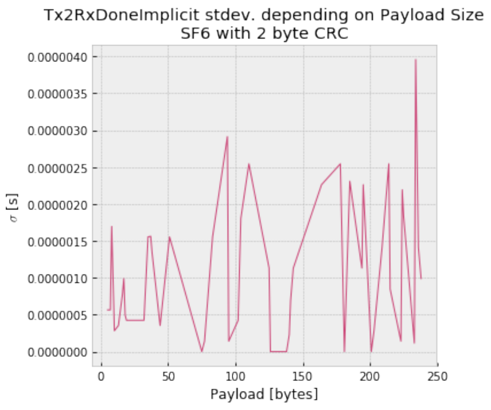

Tx2RxDone Delay depending on Payload Size
SF7 without CRC
Tx2RxDone [s]
Payload [bytes]
0.0
0.1
0.2
0.3
0.4
0
50
100
150
200
250

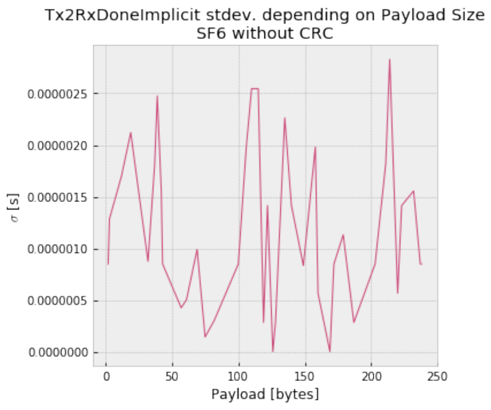

Tx2RxDoneImplicit stdev. depending on Payload Size
SF7 with 2 byte CRC
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
50
75
100
125
150
175
200
225

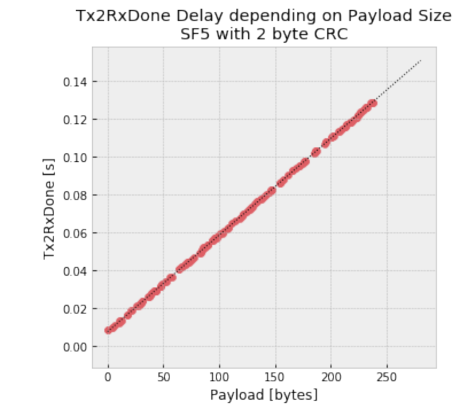

Tx2RxDoneImplicit stdev. depending on Payload Size
SF7 without CRC
σ [s]
Payload [bytes]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
0
50
100
150
200

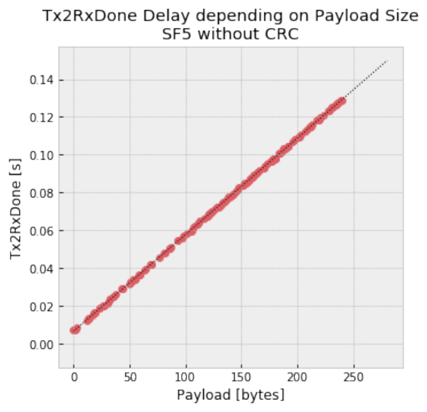

Tx2RxDone Delay depending on Payload Size
SF6 with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]

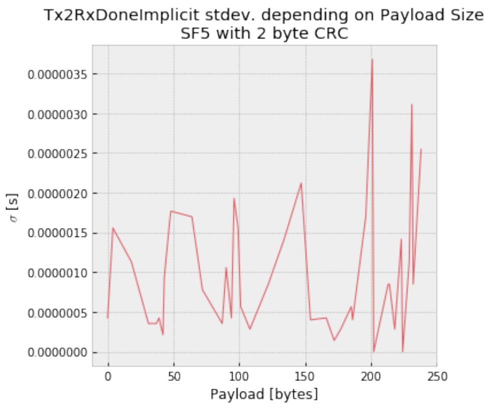

Tx2RxDone Delay depending on Payload Size
SF6 without CRC
Tx2RxDone [s]
Payload [bytes]

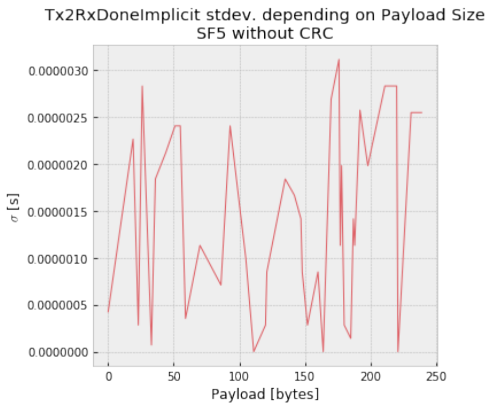

Tx2RxDoneImplicit stdev. depending on Payload Size
SF6 with 2 byte CRC
σ [s]
Payload [bytes]

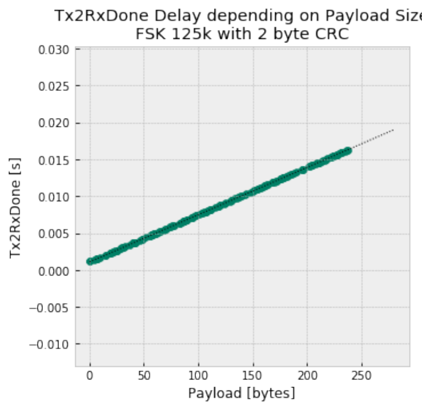

Tx2RxDoneImplicit stdev. depending on Payload Size
SF6 without CRC
σ [s]
Payload [bytes]

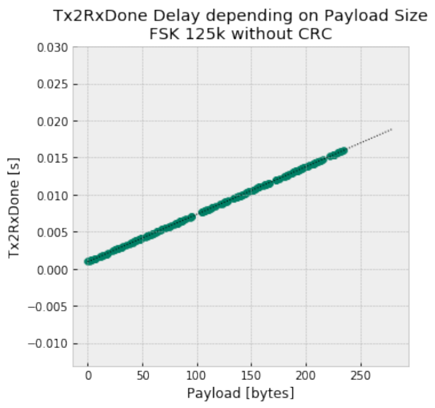

Tx2RxDone Delay depending on Payload Size
SF5 with 2 byte CRC
Tx2RxDone [s]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
50
100
150
200
250
Payload [bytes]

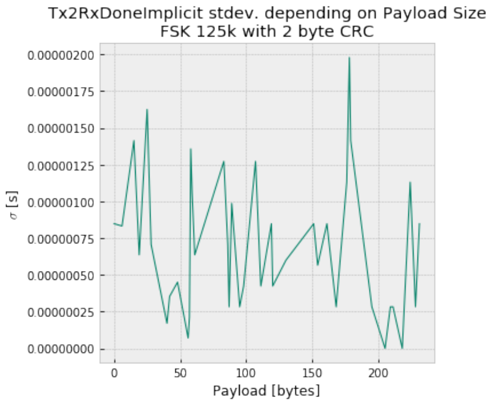

Tx2RxDone Delay depending on Payload Size
SF5 without CRC
Tx2RxDone [s]
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0
50
100
150
200
250
Payload [bytes]

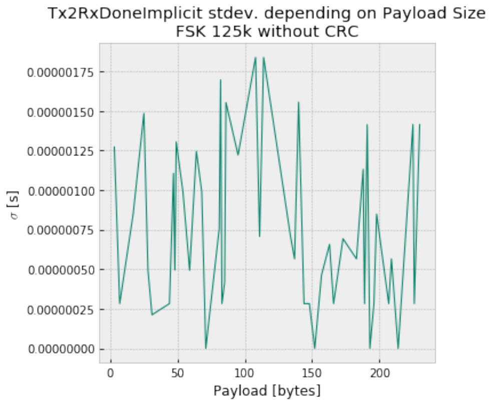

Tx2RxDoneImplicit stdev. depending on Payload Size
SF5 with 2 byte CRC
σ [s]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0.0000035
0
50
100
150
200
250
Payload [bytes]

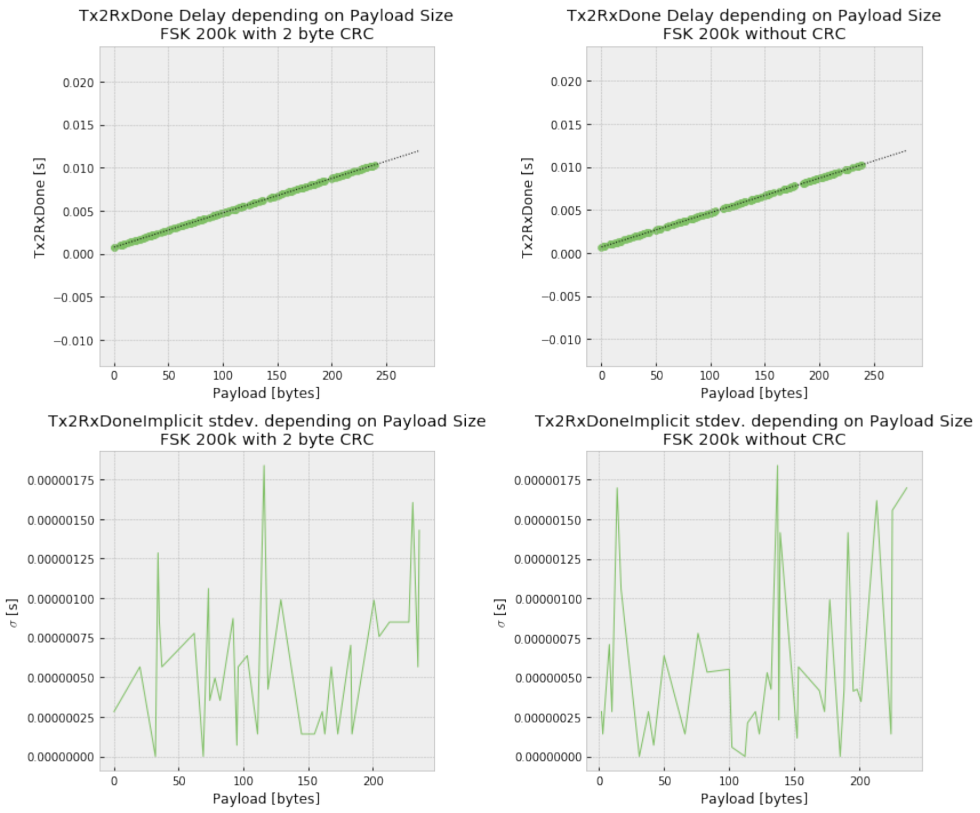

Tx2RxDoneImplicit stdev. depending on Payload Size
SF5 without CRC
σ [s]
0.0000000
0.0000005
0.0000010
0.0000015
0.0000020
0.0000025
0.0000030
0
50
100
150
200
250
Payload [bytes]


Tx2RxDone Delay depending on Payload Size
FSK 125k with 2 byte CRC
Tx2RxDone [s]
Payload [bytes]

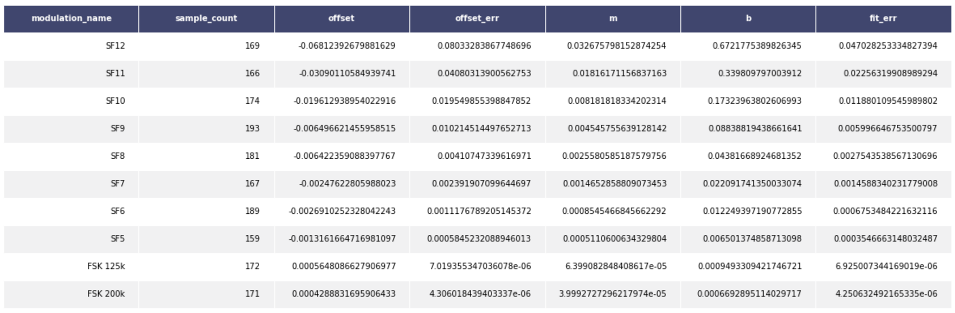

Tx2RxDone Delay depending on Payload Size
FSK 125k without CRC
Tx2RxDone [s]
Payload [bytes]


Tx2RxDoneImplicit stdev. depending on Payload Size
FSK 125k with 2 byte CRC
σ [s]
Payload [bytes]


Tx2RxDoneImplicit stdev. depending on Payload Size
FSK 125k without CRC
σ [s]
Payload [bytes]



```
In [33]: delays.style.set_precision(9)

Out[33]: <pandas.io.formats.style.Styler at 0x2194c8a6c18>

In [34]: render_mpl_table(delays, header_columns=0, col_width=3.0)

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x2194a82d2e8>
```



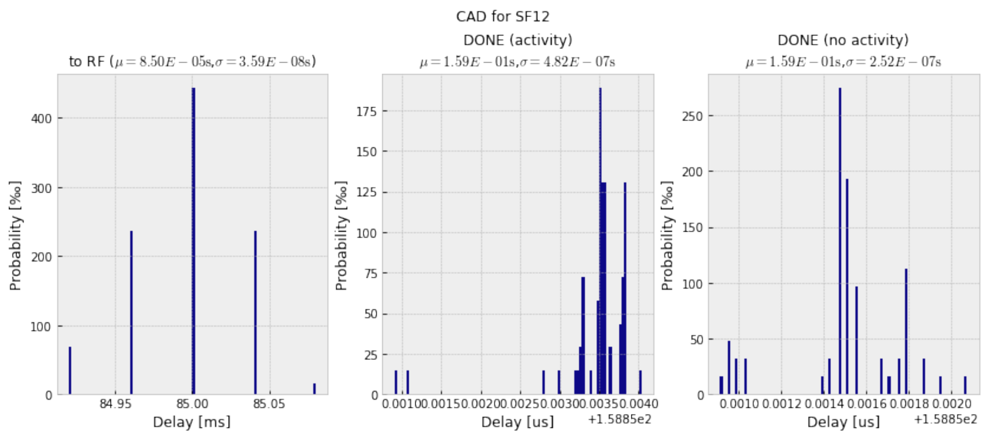
modulation_name	sample_count	offset	offset_err	m	b	fit_err
SF12	169	-0.06812392679881629	0.08033283867748696	0.032675798152874254	0.6721775389826345	0.047028253334827394
SF11	166	-0.03090110584939741	0.04080313900562753	0.01816171156837163	0.339809797003912	0.02256319908989294
SF10	174	-0.019612938954022916	0.019549855398847852	0.008181818334202314	0.17323963802606993	0.011880109545989802
SF9	193	-0.006496621455958515	0.010214514497652713	0.004545755639128142	0.08838819438661641	0.005996646753500797
SF8	181	-0.006422359088397767	0.00410747339616971	0.0025580585187579756	0.04381668924681352	0.0027543538567130696
SF7	167	-0.00247622805988023	0.002391907099644697	0.0014652858809073453	0.022091741350033074	0.0014588340231779008
SF6	189	-0.0026910252328042243	0.0011176789205145372	0.0008545466845662292	0.012249397190772855	0.0006753484221632116
SF5	159	-0.0013161664716981097	0.0005845232088946013	0.0005110600634329804	0.006501374858713098	0.0003546663148032487
FSK 125k	172	0.0005648086627906977	7.019355347036078e-06	6.399082848408617e-05	0.0009493309421746721	6.925007344169019e-06
FSK 200k	171	0.0004288831695906433	4.306018439403337e-06	3.9992727296217974e-05	0.0006692895114029717	4.250632492165335e-06

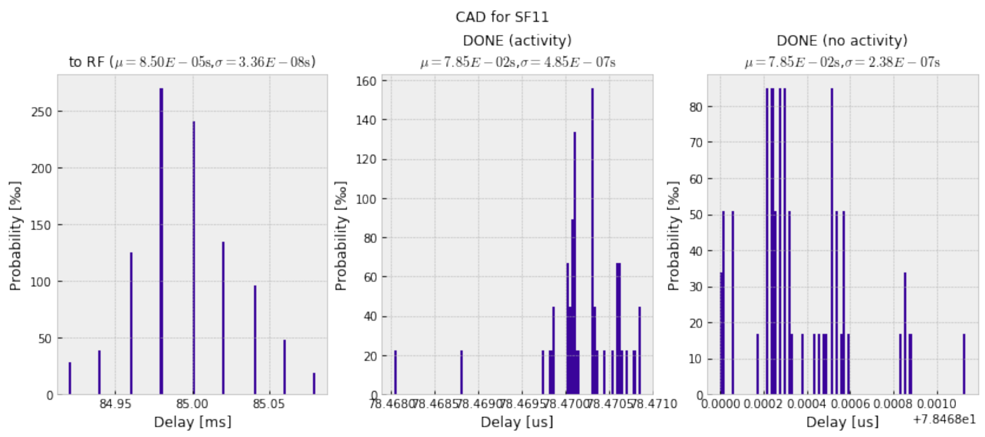
11 CAD 2 Done

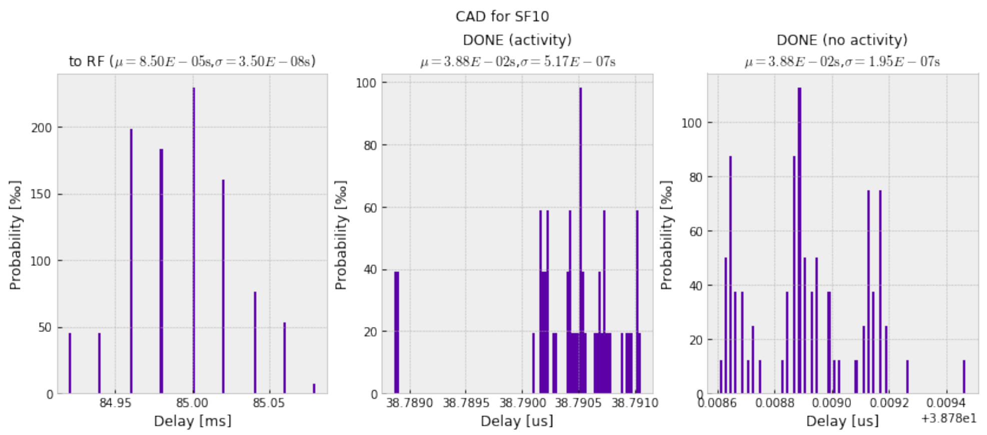
The measurements depict the the times from t_{ex} to t_{busy} and t_{ex} to t_{irq} when initiating a CAD on a single node with different outcomes. For this experiment, NSS, DIO1 and BUSY on a single node are required.

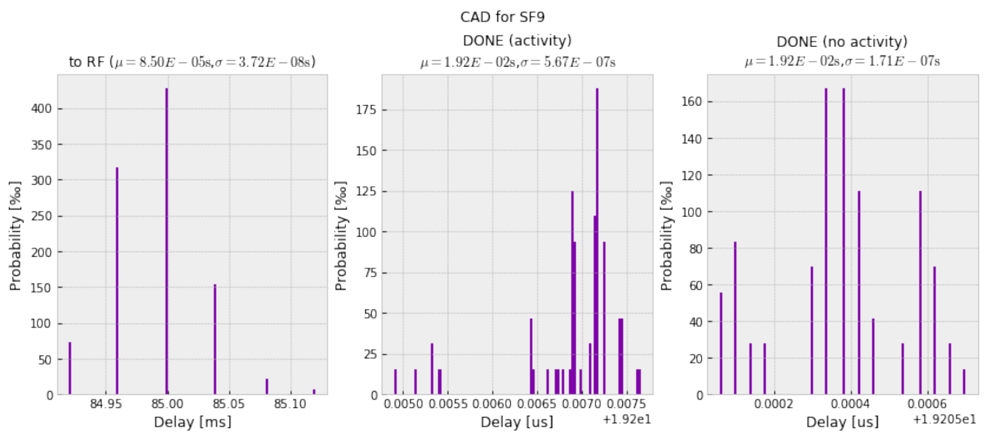
The CAD is defined to be in Rx mode for a duration of 4 symbols.

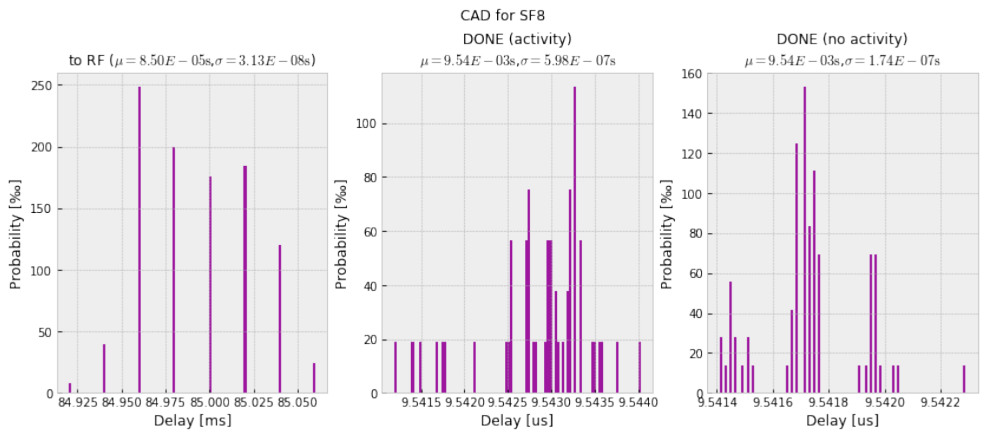
```
In [35]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_cad2done import MeasureTimeCAD2Done

         cad2done = MeasureTimeCAD2Done()
         df = pd.read_csv("../../data/{}.csv".format(cad2done.name), index_col="sample")
         (delays_detected, delays_nodetection) = cad2done.analyze(df)
```

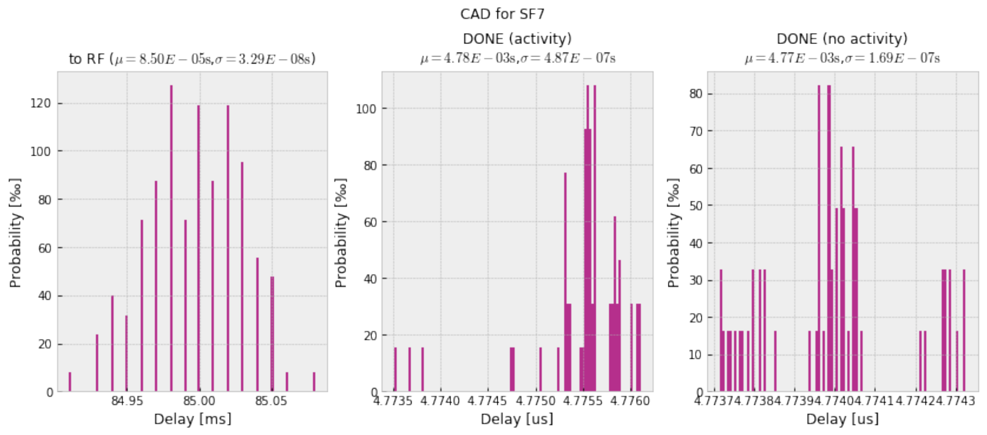


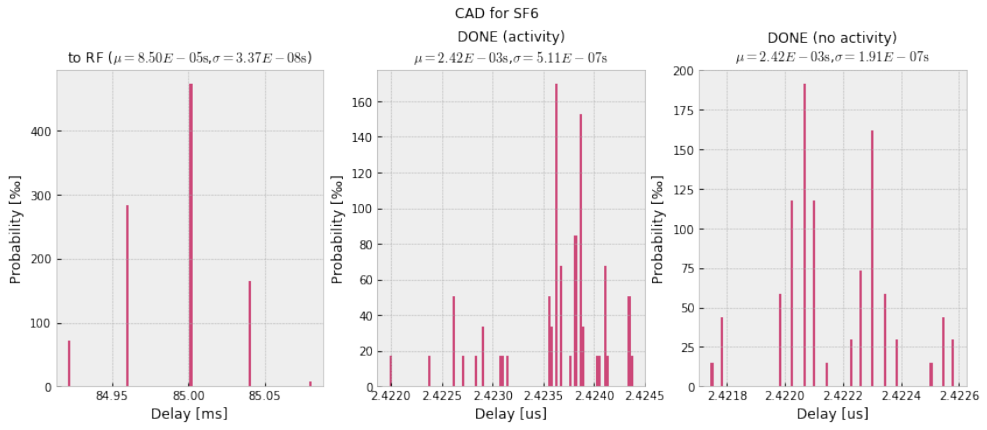


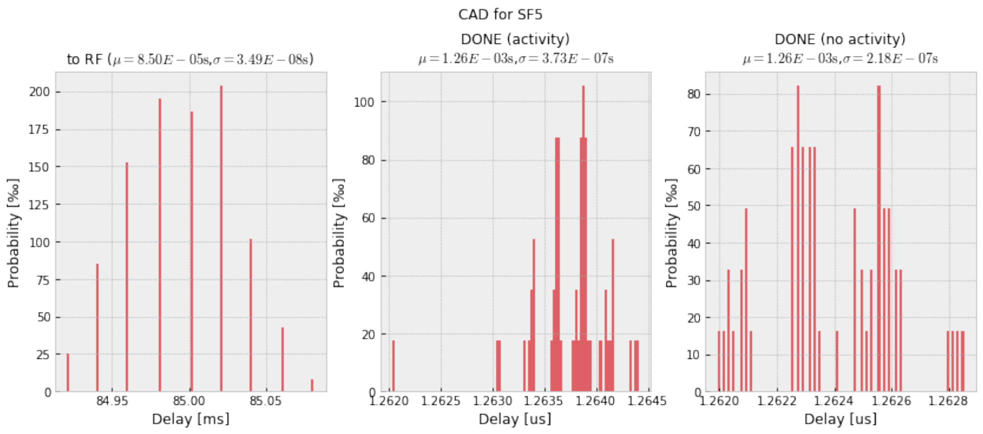

CAD for SF10
to RF ($\mu = 8.50E-05$s,$\sigma = 3.50E-08$s)
DONE (activity)
$\mu = 3.88E-02$s,$\sigma = 5.17E-07$s
DONE (no activity)
$\mu = 3.88E-02$s,$\sigma = 1.95E-07$s
Probability [‰]
Delay [ms]
Delay [us]
+3.878e1

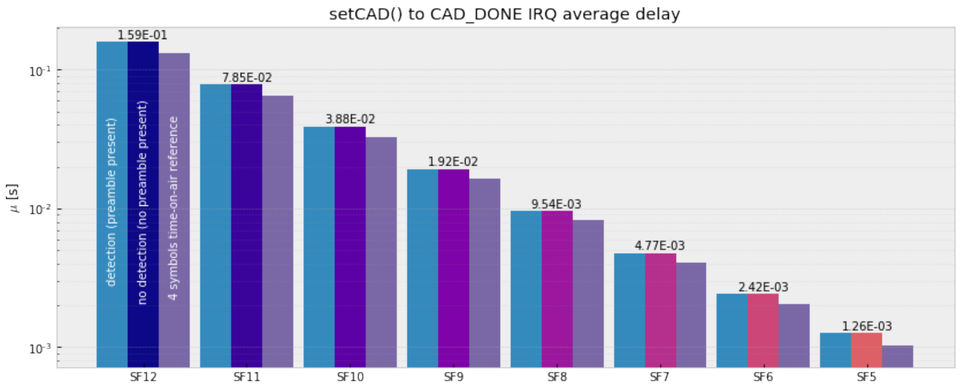

CAD for SF9
to RF ($\mu = 8.50E-05$s,$\sigma = 3.72E-08$s)
DONE (activity)
$\mu = 1.92E-02$s,$\sigma = 5.67E-07$s
DONE (no activity)
$\mu = 1.92E-02$s,$\sigma = 1.71E-07$s
Probability [‰]
Delay [ms]
Delay [us]
+1.92e1
+1.9205e1

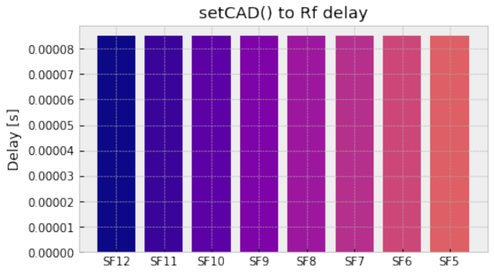

CAD for SF8
to RF ($\mu = 8.50E-05$s,$\sigma = 3.13E-08$s)
DONE (activity)
$\mu = 9.54E-03$s,$\sigma = 5.98E-07$s
DONE (no activity)
$\mu = 9.54E-03$s,$\sigma = 1.74E-07$s
Probability [‰]
Delay [ms]
Delay [us]
Delay [us]

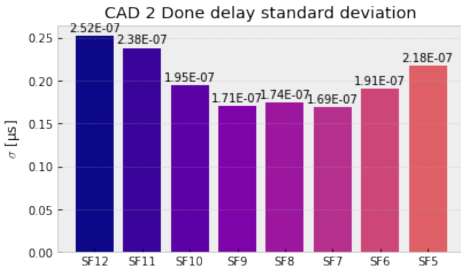

CAD for SF7
to RF ($\mu = 8.50E-05$s,$\sigma = 3.29E-08$s)
DONE (activity)
$\mu = 4.78E-03$s,$\sigma = 4.87E-07$s
DONE (no activity)
$\mu = 4.77E-03$s,$\sigma = 1.69E-07$s
Probability [‰]
Delay [ms]
Delay [us]
Delay [us]


CAD for SF6
to RF ($\mu = 8.50E-05$s,$\sigma = 3.37E-08$s)
DONE (activity)
$\mu = 2.42E-03$s,$\sigma = 5.11E-07$s
DONE (no activity)
$\mu = 2.42E-03$s,$\sigma = 1.91E-07$s
Probability [‰]
Delay [ms]
Delay [us]
Delay [us]



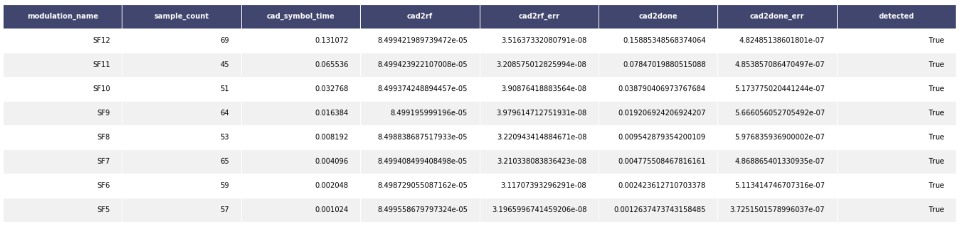






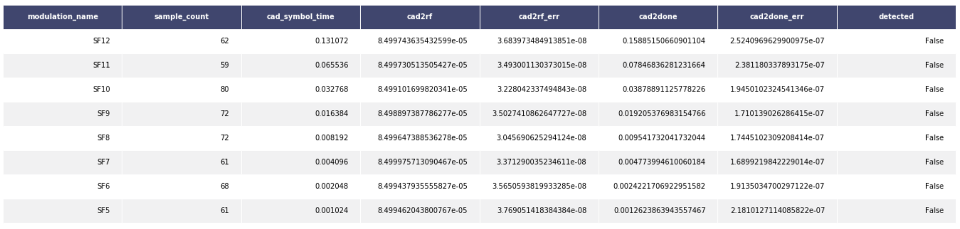
```
In [36]: delays_detected.style.bar(subset='cad2done', color='lightgray').set_precision(9)

Out[36]: <pandas.io.formats.style.Styler at 0x21950cd6da0>
```

```
In [37]: render_mpl_table(delays_detected, header_columns=0, col_width=3.0)

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x2194fc5e940>
```



modulation_name	sample_count	cad_symbol_time	cad2rf	cad2rf_err	cad2done	cad2done_err	detected
SF12	69	0.131072	8.499421989739472e-05	3.51637332080791e-08	0.15885348568374064	4.82485138601801e-07	True
SF11	45	0.065536	8.499423922107008e-05	3.208575012825994e-08	0.07847019880515088	4.853857086470497e-07	True
SF10	51	0.032768	8.499374248894457e-05	3.90876418883564e-08	0.038790406973767684	5.173775020441244e-07	True
SF9	64	0.016384	8.499195999196e-05	3.979614712751931e-08	0.019206924206924207	5.666056052705492e-07	True
SF8	53	0.008192	8.498838687517933e-05	3.220943414884671e-08	0.009542879354200109	5.976835936900002e-07	True
SF7	65	0.004096	8.499408499408498e-05	3.210338083836423e-08	0.004775508467816161	4.868865401330935e-07	True
SF6	59	0.002048	8.498729055087162e-05	3.11707393296291e-08	0.002423612710703378	5.113414746707316e-07	True
SF5	57	0.001024	8.499558679797324e-05	3.1965996741459206e-08	0.0012637473743158485	3.7251501578996037e-07	True

```
In [38]: delays_nodetection.style.bar(subset='cad2done', color='lightgray').set_precision(9)

Out[38]: <pandas.io.formats.style.Styler at 0x2194fcdeb38>

In [39]: render_mpl_table(delays_nodetection, header_columns=0, col_width=3.0)

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x2194fcc3080>
```

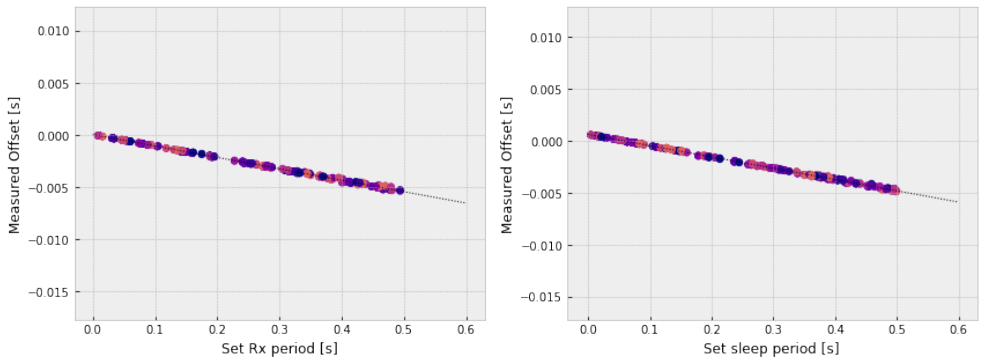


modulation_name	sample_count	cad_symbol_time	cad2rf	cad2rf_err	cad2done	cad2done_err	detected
SF12	62	0.131072	8.4997436354325 99e-05	3.683973484913851e-08	0.15885150660901104	2.5240969629900975e-07	False
SF11	59	0.065536	8.499730513505427e-05	3.493001130373015e-08	0.07846836281231664	2.381180337893175e-07	False
SF10	80	0.032768	8.499101699820341e-05	3.228042337494843e-08	0.03878891125778226	1.9450102324541346e-07	False
SF9	72	0.016384	8.498897387786277e-05	3.5027410862647727e-08	0.019205376983154766	1.710139026286415e-07	False
SF8	72	0.008192	8.499647388536278e-05	3.045690625294124e-08	0.009541732041732044	1.7445102309208414e-07	False
SF7	61	0.004096	8.499975713090467e-05	3.371290035234611e-08	0.004773994610060184	1.6899219842229014e-07	False
SF6	68	0.002048	8.499437935555827e-05	3.5650593819933285e-08	0.0024221706922951582	1.9135034700297122e-07	False
SF5	61	0.001024	8.499462043800767e-05	3.769051418384384e-08	0.0012623863943557467	2.1810127114085822e-07	False

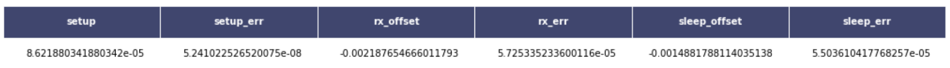
12 Listen/Sniff Mode Timings

The measurements depict the absolute values of sleep and Rx period in sniff mode. The are reconstructed by measuring the time intervals between falling- and rising-edge of the BUSY signal.

```
In [40]: %autoreload
         %matplotlib inline

         from experiments.measure_time_sniff_mode import MeasureTimeSniffMode

         sniff_mode = MeasureTimeSniffMode()
         df = pd.read_csv("../../data/{}.csv".format(sniff_mode.name), index_col="sample")
         timings = sniff_mode.analyze(df)
```



```
In [41]: timings.style.set_precision(9)

Out[41]: <pandas.io.formats.style.Styler at 0x2194aed0860>

In [42]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x21950496978>
```



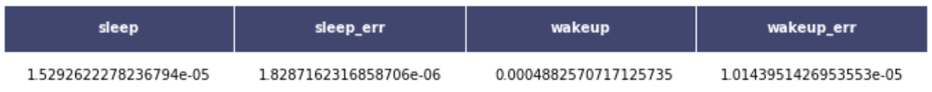
setup	setup_err	rx_offset	rx_err	sleep_offset	sleep_err
8.621880341880342e-05	5.241022526520075e-08	-0.002187654666011793	5.725335233600116e-05	-0.0014881788114035138	5.503610417768257e-05

13 Sleep Mode Timings

The measurements depict the delays t_{ex} to $t_{busy_{high}}$ of going to sleep and wakeup from warm state on a single node. NSS and BUSY on a single node are required.

```
In [43]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_sleep_mode import MeasureTimeSleepMode

         sleep_mode = MeasureTimeSleepMode()
         df = pd.read_csv("../../data/{}.csv".format(sleep_mode.name), index_col="sample")
         timings = sleep_mode.analyze(df)

In [44]: timings.style.set_precision(9)

Out[44]: <pandas.io.formats.style.Styler at 0x21950590780>
```

```
In [45]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x219503cf7f0>
```

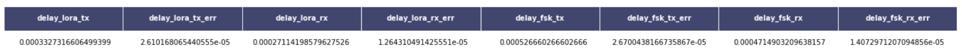


sleep	sleep_err	wakeup	wakeup_err
1.5292622278236794e-05	1.8287162316858706e-06	0.0004882570717125735	1.0143951426953553e-05

14 Config Timings

The measurements depict the delays t_{init} to t_{ex} of setting various radio configurations (Tx/Rx, LoRa/GFSK). NSS on a single node is required.

```
In [46]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_config import MeasureTimeConfig

         time_config = MeasureTimeConfig()
         df = pd.read_csv("../../data/{}.csv".format(time_config.name), index_col="sample")
         timings = time_config.analyze(df)

In [47]: timings.style.set_precision(9)

Out[47]: <pandas.io.formats.style.Styler at 0x219505ef438>

In [48]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x21950601f28>
```



delay_lora_tx	delay_lora_tx_err	delay_lora_rx	delay_lora_rx_err	delay_fsk_tx	delay_fsk_tx_err	delay_fsk_rx	delay_fsk_rx_err
0.0003327316606499399	2.610168065440555e-05	0.00027114198579627526	1.264310491425551e-05	0.000526660266602666	2.6700438166735867e-05	0.0004714903209638157	1.4072971207094856e-05

15 Set Payload Timings

The measurements depict the delays t_{init} to t_{ex} of setting the payload depending on the modem LoRa/GFSK (due to longer commands involving setting the payload buffer length). NSS on a single node is required.

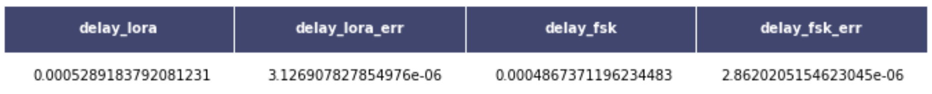
```
In [49]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_set_payload import MeasureTimeSetPayload

         set_payload = MeasureTimeSetPayload()
         df = pd.read_csv("../../data/{}.csv".format(set_payload.name), index_col="sample")
         timings = set_payload.analyze(df)
```

```
In [50]: timings.style.set_precision(9)
```

```
Out[50]: <pandas.io.formats.style.Styler at 0x2195064fa58>
```

```
In [51]: render_mpl_table(timings, header_columns=0, col_width=3.0)
```

```
Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x2195066e978>
```



delay_lora	delay_lora_err	delay_fsk	delay_fsk_err
0.0005289183792081231	3.126907827854976e-06	0.0004867371196234483	2.8620205154623045e-06

16 Get Payload Timings

The measurements depict the delays t_{init} to t_{ex} of reading the payload depending on the modem LoRa/GFSK (due to longer commands involving getting the payload buffer length). `NSS` on a single node is required.

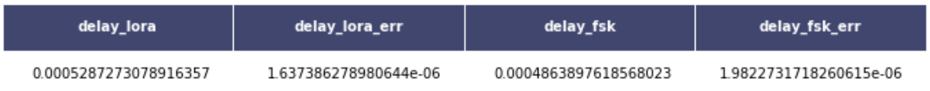
```
In [52]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_get_payload import MeasureTimeGetPayload

         get_payload = MeasureTimeGetPayload()
         df = pd.read_csv("../../data/{}.csv".format(get_payload.name), index_col="sample")
         timings = get_payload.analyze(df)
```

```
In [53]: timings.style.set_precision(9)
```

```
Out[53]: <pandas.io.formats.style.Styler at 0x219506a65f8>
```

```
In [54]: render_mpl_table(timings, header_columns=0, col_width=3.0)
```

```
Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x219506700b8>
```

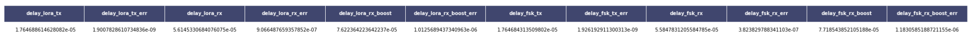


delay_lora	delay_lora_err	delay_fsk	delay_fsk_err
0.0005287273078916357	1.637386278980644e-06	0.0004863897618568023	1.9822731718260615e-06

17 SetFS Timings

The measurements depict the delays t_{init} to t_{ex} of setting the radio into the given operation mode Tx or Rx (i.e. how long the `SetTx` or `SetRx` itself take. `NSS` on a single node is required.

```
In [55]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_set_fs import MeasureTimeSetFS

         set_fs = MeasureTimeSetFS()
         df = pd.read_csv("../../data/{}.csv".format(set_fs.name), index_col="sample")
         timings = set_fs.analyze(df)

In [56]: timings.style.set_precision(9)

Out[56]: <pandas.io.formats.style.Styler at 0x219506a6d30>

In [57]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x21950a97e10>
```



delay_lora_tx	delay_lora_tx_err	delay_lora_rx	delay_lora_rx_err	delay_lora_rx_boost	delay_lora_rx_boost_err
1.764688614628082e-05	1.9007828610734836e-09	5.6145330684076075e-05	9.066487659357852e-07	7.622364223642237e-05	1.0125689437340963e-06

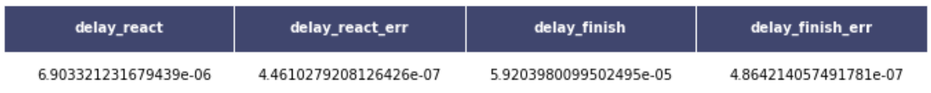
delay_fsk_tx	delay_fsk_tx_err	delay_fsk_rx	delay_fsk_rx_err	delay_fsk_rx_boost	delay_fsk_rx_boost_err
1.764684313509802e-05	1.926192911300313e-09	5.5847831205584785e-05	3.823829788341103e-07	7.718543852105188e-05	1.1830585188721155e-06

18 Process IRQ Timings

The measurements depict the delays t_{irq} to t_{init} and t_{irq} to t_{ex} of processing an IRQ by `DIO1`. `NSS` and `DIO1`on a single node are required.

```
In [58]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_irq_process import MeasureTimeIRQProcess

         irq_process = MeasureTimeIRQProcess()
         df = pd.read_csv("../../data/{}.csv".format(irq_process.name), index_col="sample")
         timings = irq_process.analyze(df)
```

```
In [59]: timings.style.set_precision(9)

Out[59]: <pandas.io.formats.style.Styler at 0x21952ea5978>

In [60]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x21952ec5a58>
```

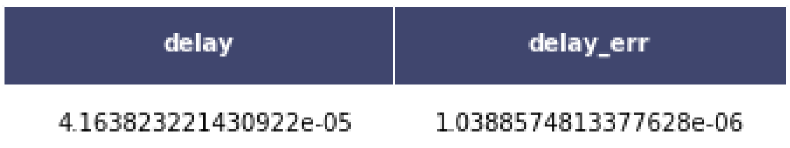


delay_react	delay_react_err	delay_finish	delay_finish_err
6.903321231679439e-06	4.4610279208126426e-07	5.9203980099502495e-05	4.864214057491781e-07

19 Get Packet Status Timings

The measurements depict the delays t_{init} to t_{ex} of reading the packet status. NSS on a single node is required.

```
In [61]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_get_pkt_status import MeasureTimeGetPktStatu

         get_pkt_status = MeasureTimeGetPktStatus()
         df = pd.read_csv("../../data/{}.csv".format(get_pkt_status.name), index_col="sample")
         timings = get_pkt_status.analyze(df)

In [62]: timings.style.set_precision(9)

Out[62]: <pandas.io.formats.style.Styler at 0x21952f0ecf8>

In [63]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x21952ed0400>
```



delay	delay_err
4.163823221430922e-05	1.0388574813377628e-06

20 SetCAD Timings

The measurements depict the delays t_{init} to t_{ex} of sending the CAD command. `NSS` on a single node is required.

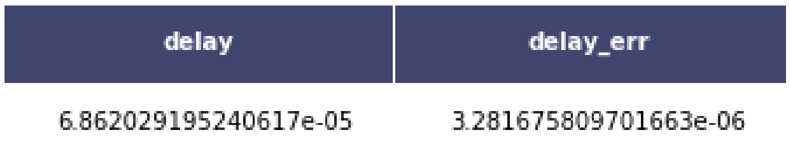
```
In [64]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_set_cad import MeasureTimeSetCAD

         set_cad = MeasureTimeSetCAD()
         df = pd.read_csv("../../data/{}.csv".format(set_cad.name), index_col="sample")
         timings = set_cad.analyze(df)
```

```
In [65]: timings.style.set_precision(9)
```

```
Out[65]: <pandas.io.formats.style.Styler at 0x219535bde48>
```

```
In [66]: render_mpl_table(timings, header_columns=0, col_width=3.0)
```

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x21952f36f60>
```



delay	delay_err
6.862029195240617e-05	3.281675809701663e-06

20.1 CAD Scan Time

We can now use the information to calculate the theoretical time needed run channel activity detection (also with GFSK) over all ISM/SRD bands available for every Gloria modulation index.

```
In [67]: total_time = 0

         for i in range(0,8):
             single_time = 6.8620292e-05
             single_time += delays_nodetection.loc[i].cad2done
             single_time += 5.92039801e-05
             total_time += single_time

         for i in range(8,10):
             config = RadioConfiguration(i, preamble=2, )
             math = RadioMath(config)
```

```
            single_time = 7.71854385e-05
            single_time += math.get_symbol_time() * 2
            single_time += 0.000357
            total_time += single_time

        total_time *= 48

        total_time
```

```
Out[67]: 15.139843509816378
```

For only one ISM/SRD band:

```
In [68]: total_time / 48
```

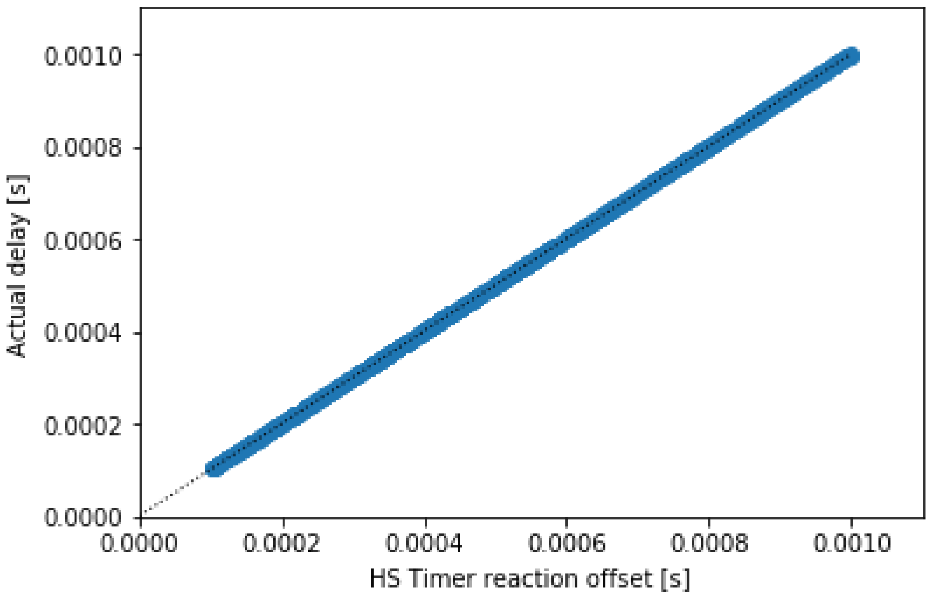
```
Out[68]: 0.3154134064545079
```

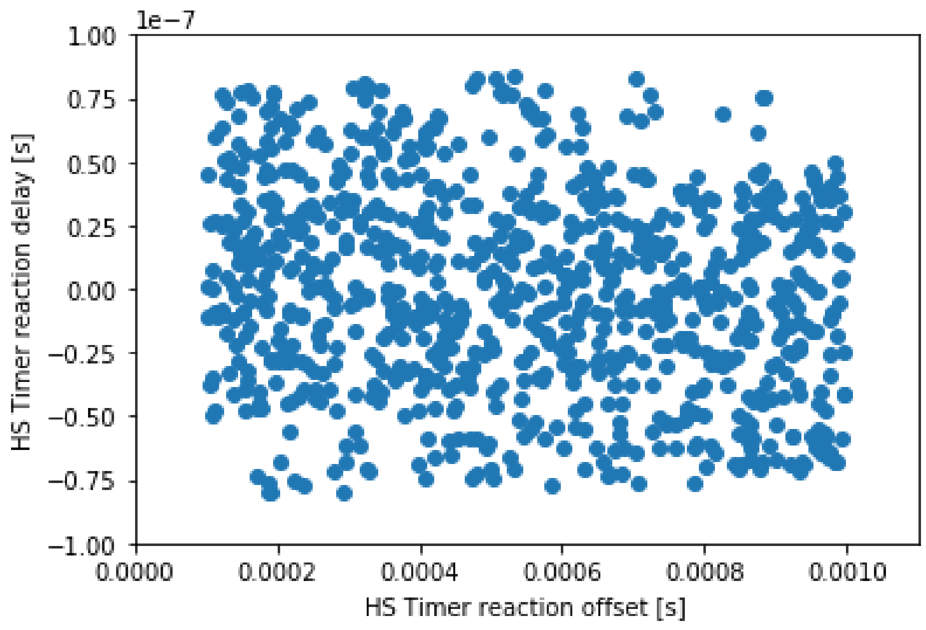
21 HS Timer Sync Timings

The measurements depict the delays t_{irq} to t_{ex} depending on specified delay times for calibration of TIM2. `NSS` and `DIO1` on a single node are required.

Note the two square shaped overlapping zones in the `reaction delay` chart, which is depicting the error.

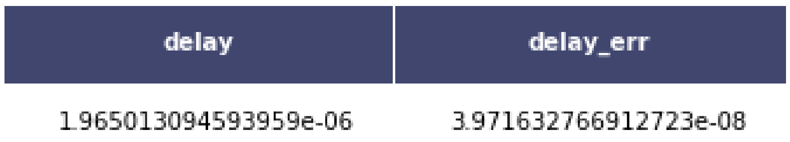
```
In [69]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_sync import MeasureTimeSync

         sync = MeasureTimeSync()
         df = pd.read_csv("../../data/{}.csv".format(sync.name), index_col="sample")
         timings = sync.analyze(df)
```





```
In [70]: timings.style.set_precision(9)
```

```
Out[70]: <pandas.io.formats.style.Styler at 0x21952f364a8>

In [71]: render_mpl_table(timings, header_columns=0, col_width=3.0)

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x21953583588>
```

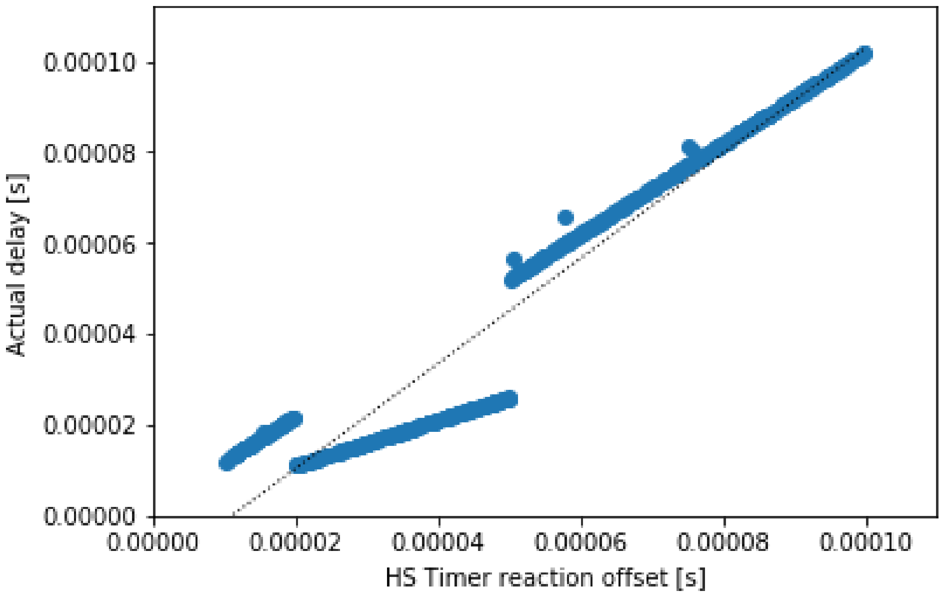


delay	delay_err
1.9650130945939e-06	3.971632766912723e-08

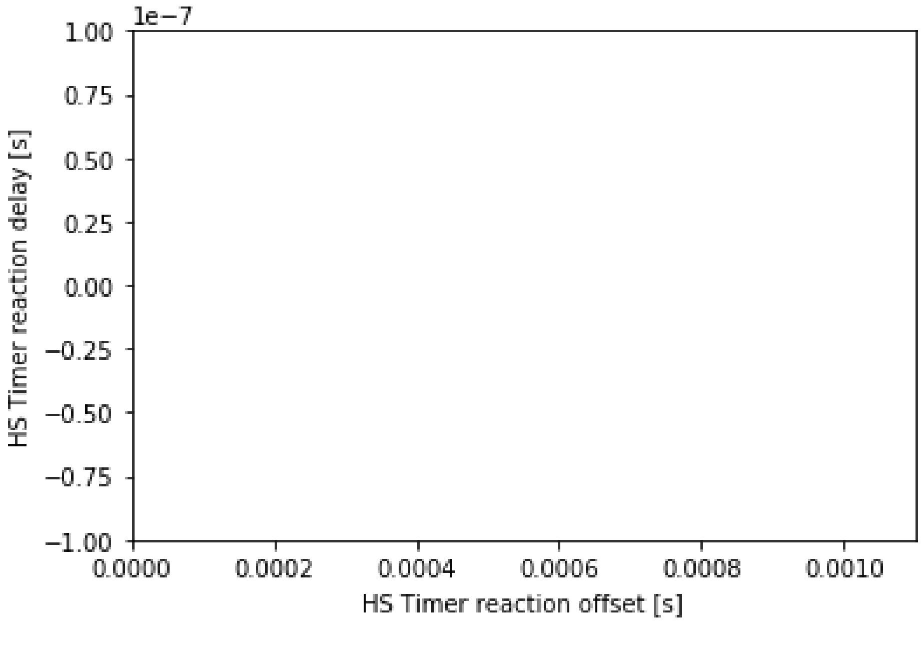
21.1 Short schedule intervals < 100 us

Scheduling of TIM2 below 100 us shows some chaotic sideffects. Schedule operations should therefore be always higher than 100 us.

```
In [72]: %autoreload
         %matplotlib inline

         from flora_tools.experiments.measure_time_sync import MeasureTimeSync

         sync = MeasureTimeSync()
         df = pd.read_csv("../../data/{}Lower100us.csv".format(sync.name), index_col="sample")
         timings = sync.analyze(df)
```



Actual delay [s]
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0.00000
0.00002
0.00004
0.00006
0.00008
0.00010
HS Timer reaction offset [s]



1e−7
HS Timer reaction delay [s]
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
HS Timer reaction offset [s]

21.2 Tx Rx SYNC/HEADER_VALID Sweep

The measurements depict the simultaneous reception and transmission on two DevKits. The Rx window is dimensioned to be the same length as the preamble's ToA. A relative offset of zero means that Tx preamble and Rx window are perfectly aligned. -1 means that the Tx preamble has been sent before the Rx window began, +1 means that the Rx window is prepending the Tx preamble (therefore not being able to receive the packet in both cases).

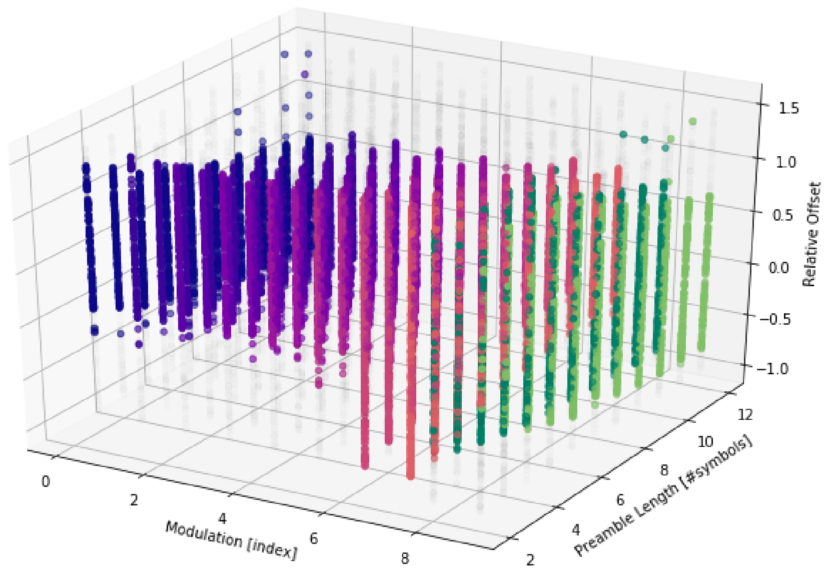
The receveing node is using the `StopTimerOnPreamble` configuration (see SX1262 datasheet) meaning that with recognition of a preamble symbol the radio stays in Rx mode indefinitely. It has to be put therefore into Standby mode manually by the MCU.

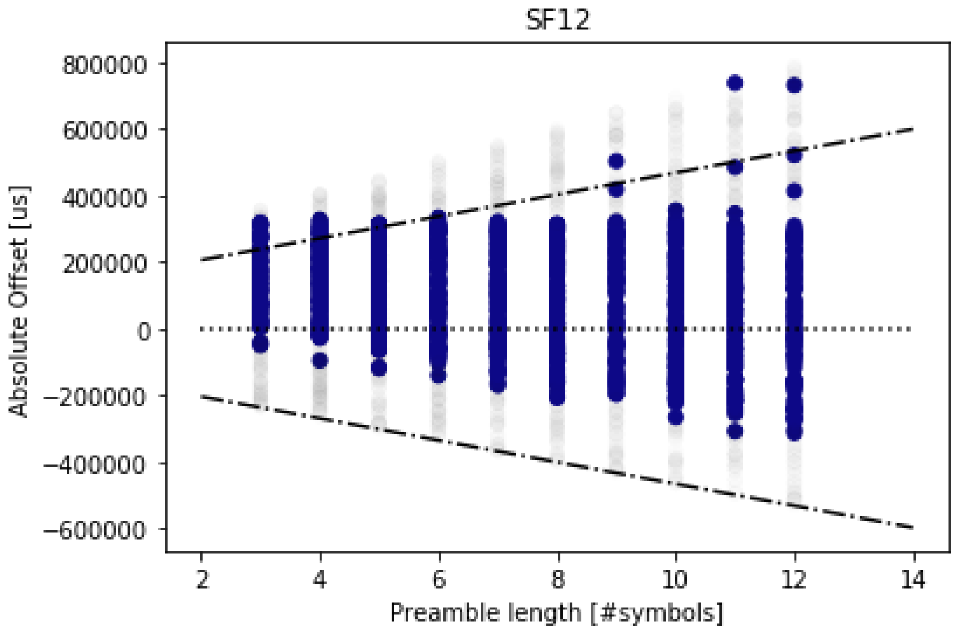
Colored points indicate a correct reception of the header, gray-transparent points did not receive the header correctly or did not even detect the preamble.

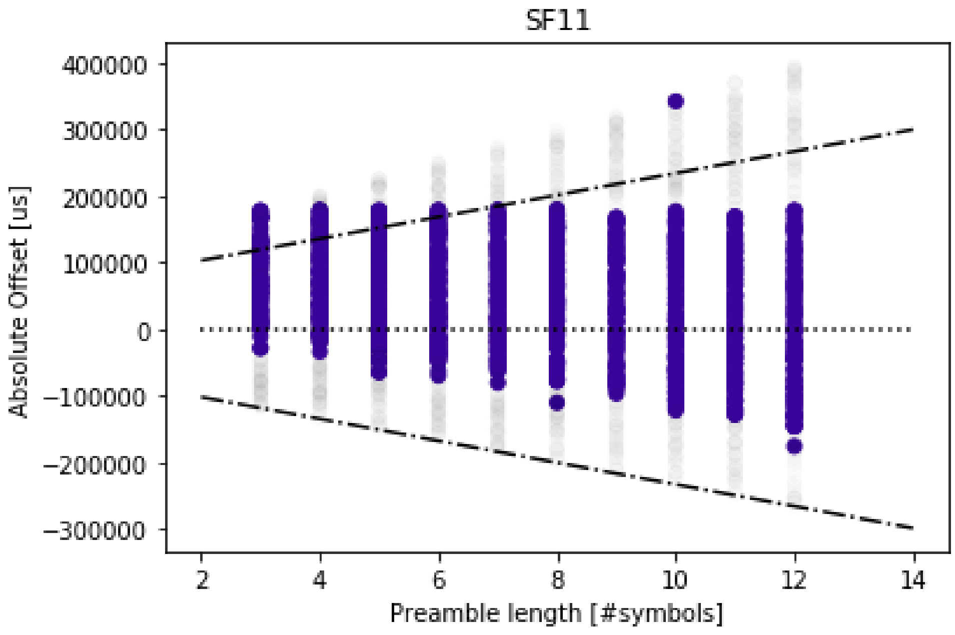
For this experiment, TIM2 synchronization is required: `DIO1` on the synchronization sink has to be connected to `DBG_GPIO1` (`PC9`, `CN5` Pin 1). For more than one sink the signal has to be buffered with a high-speed schmitt-trigger or ampflifier. The author used a low-capacitance MOSFET half-bridge for his measurements.

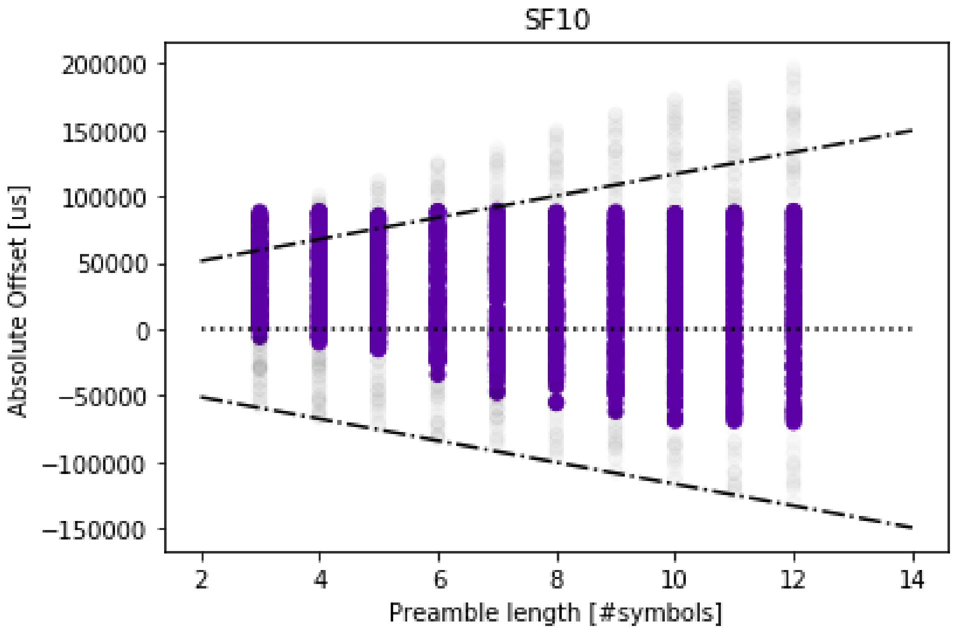
Signals to measure include `NSS` and `DIO1` on the receiving node.

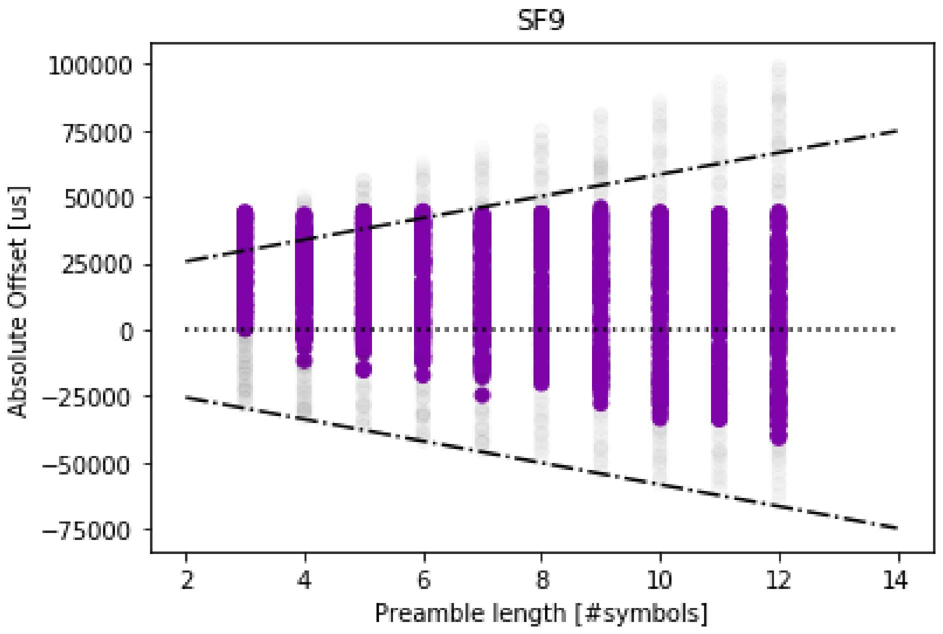
```
In [73]: %autoreload

In [74]: %matplotlib inline

         from flora_tools.experiments.measure_time_sweep_tx_rx import MeasureTimeSweepTxRx

         sweep_tx_rx = MeasureTimeSweepTxRx()
         df = pd.read_csv("../../data/{}.csv".format(sweep_tx_rx.name), index_col="sample")
         timings = sweep_tx_rx.analyze(df)
```

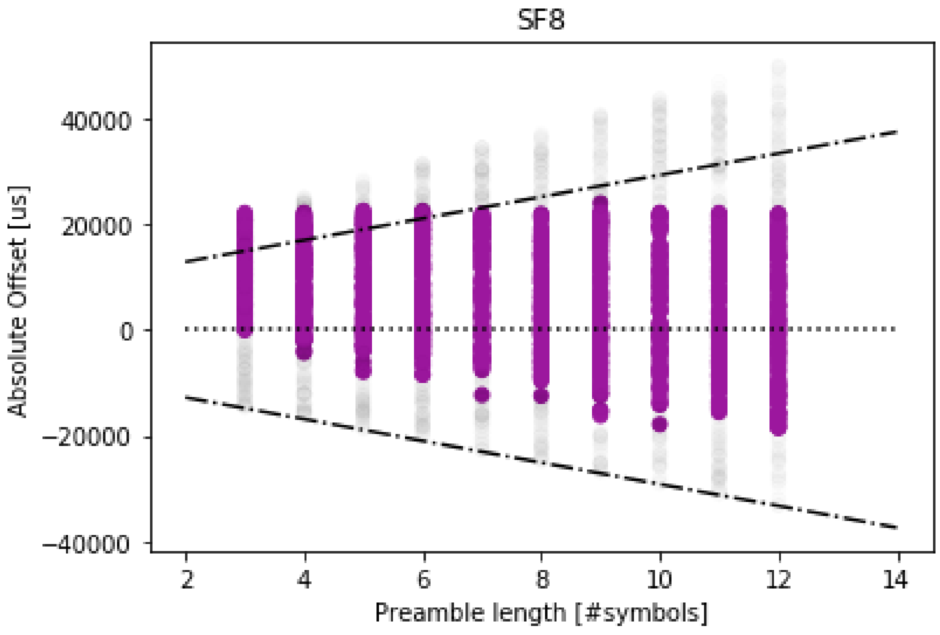


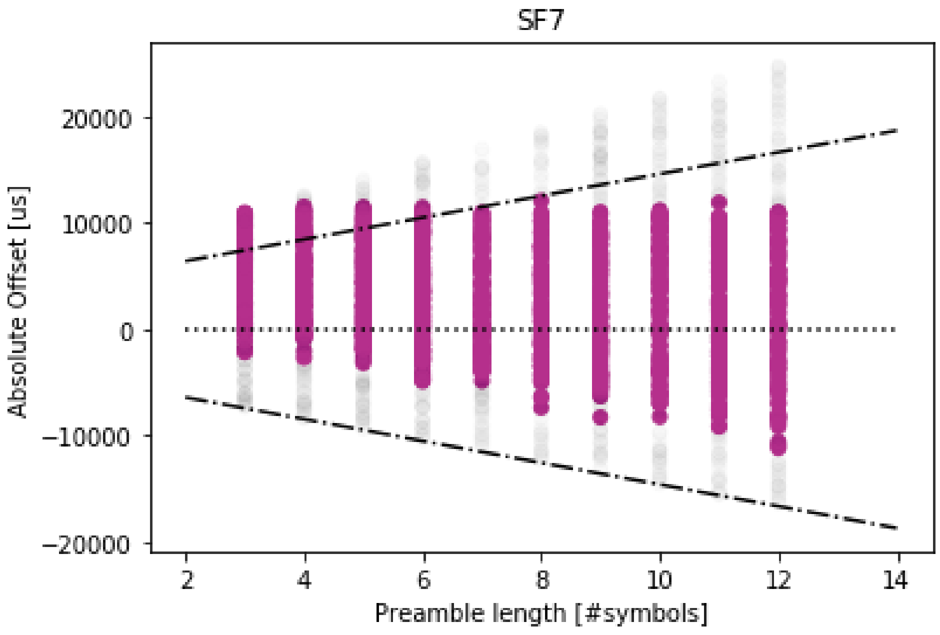

SF12
Absolute Offset [us]
800000
600000
400000
200000
0
−200000
−400000
−600000
2
4
6
8
10
12
14
Preamble length [#symbols]

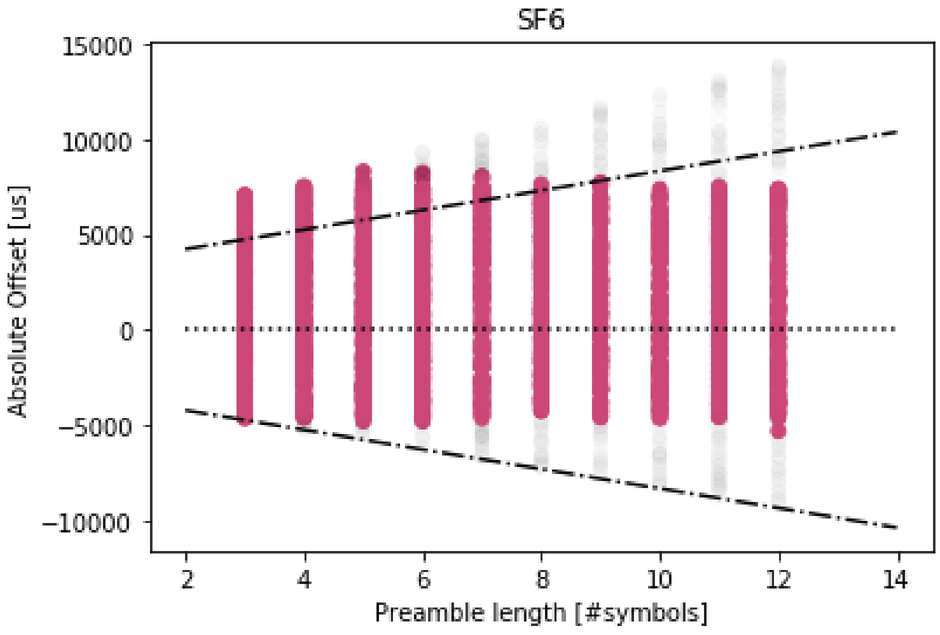

SF11
Absolute Offset [us]
400000
300000
200000
100000
0
−100000
−200000
−300000
2
4
6
8
10
12
14
Preamble length [#symbols]

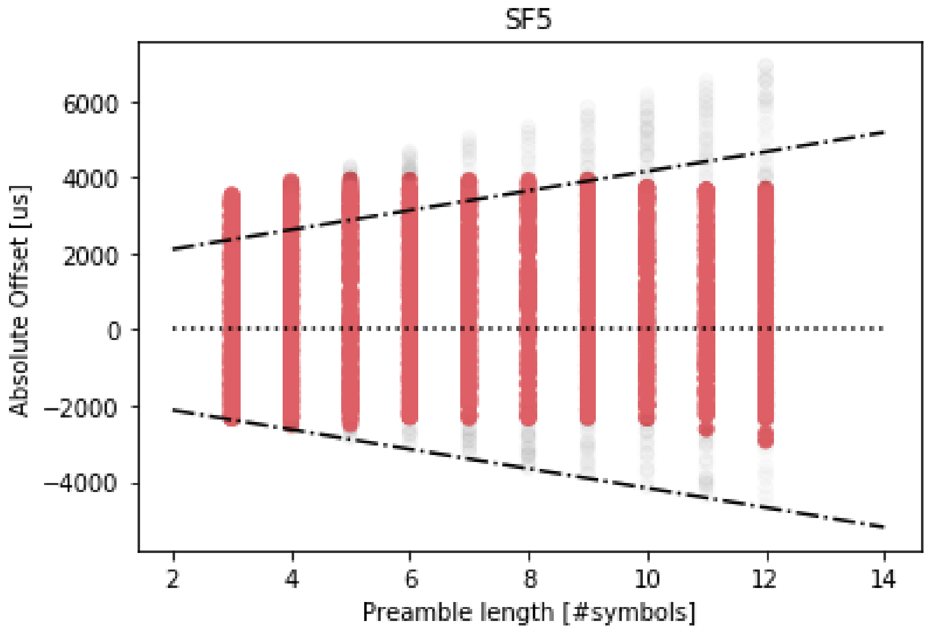

SF10
200000
150000
100000
50000
0
−50000
−100000
−150000
Absolute Offset [us]
2
4
6
8
10
12
14
Preamble length [#symbols]

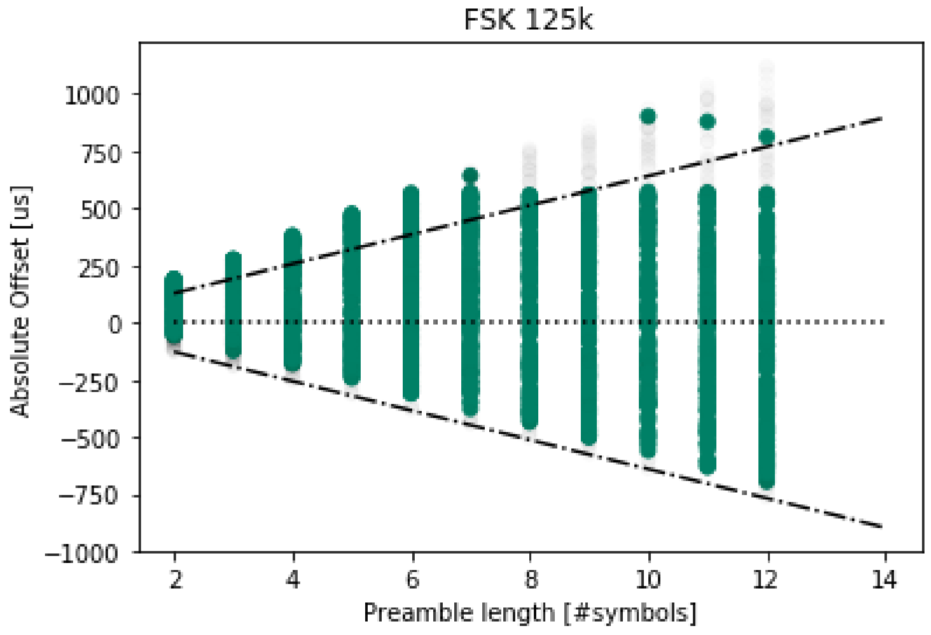

SF9
100000
75000
50000
25000
0
−25000
−50000
−75000
Absolute Offset [us]
2
4
6
8
10
12
14
Preamble length [#symbols]

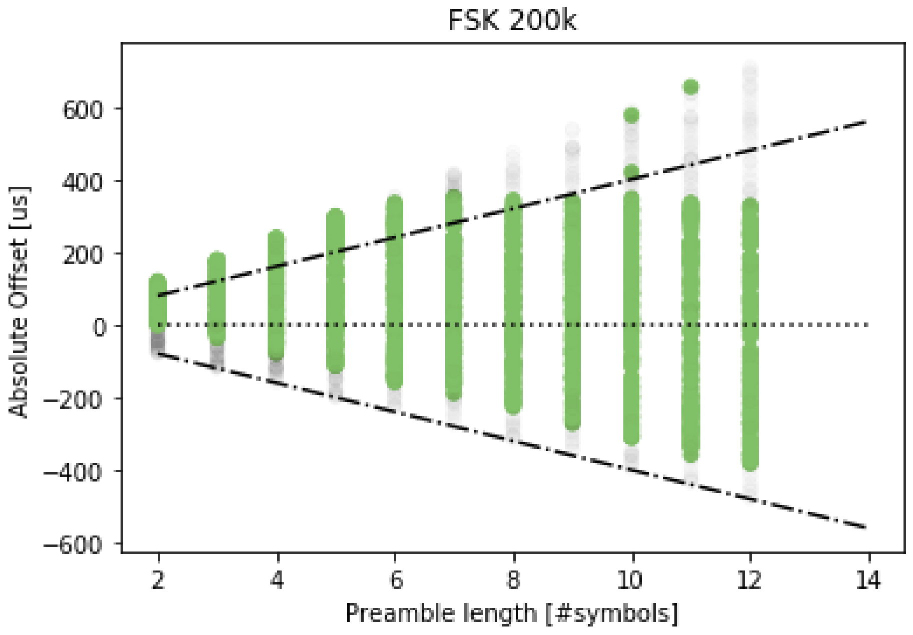

SF8
Absolute Offset [us]
40000
20000
0
−20000
−40000
2
4
6
8
10
12
14
Preamble length [#symbols]


SF7
Absolute Offset [us]
20000
10000
0
−10000
−20000
2
4
6
8
10
12
14
Preamble length [#symbols]


SF6
15000
10000
5000
0
−5000
−10000
Absolute Offset [us]
2
4
6
8
10
12
14
Preamble length [#symbols]


SF5
6000
4000
2000
0
−2000
−4000
Absolute Offset [us]
2
4
6
8
10
12
14
Preamble length [#symbols]


FSK 125k
Absolute Offset [us]
1000
750
500
250
0
−250
−500
−750
−1000
2
4
6
8
10
12
14
Preamble length [#symbols]


FSK 200k
Absolute Offset [us]
600
400
200
0
−200
−400
−600
2
4
6
8
10
12
14
Preamble length [#symbols]

22 Tx Tx Rx SYNC/HEADER_VALID Sweep

The measurements depict the simultaneous reception and transmission on three DevKits. The Rx window is dimensioned to be the same length as the preamble's ToA. A relative offset of zero means for a given node means that it's Tx preamble and Rx window are perfectly aligned. -1 means that the Tx preamble has been sent before the Rx window began, +1 means that the Rx window is prepending the Tx preamble (therefore not being able to receive the packet in both cases).

The measured zone is triangular in shape, as every combination can be achieved.

The receveing node is using the `StopTimerOnPreamble` configuration (see SX1262 datasheet) meaning that with recognition of a preamble symbol the radio stays in Rx mode indefinitely. It has to be put therefore into Standby mode manually by the MCU.

Colored points indicate a correct reception of the header, gray-transparent points did not receive the header correctly or did not even detect the preamble.

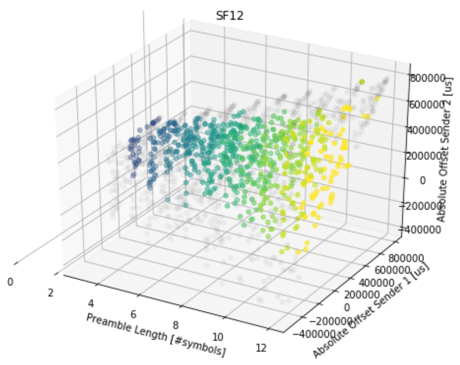
For this experiment, TIM2 synchronization is required: `DIO1` on the synchronization sink has to be connected to `DBG_GPIO1` (`PC9`, `CN5` Pin 1). For more than one sink the signal has to be buffered with a high-speed schmitt-trigger or ampflifier. The author used a low-capacitance MOSFET half-bridge for his measurements.

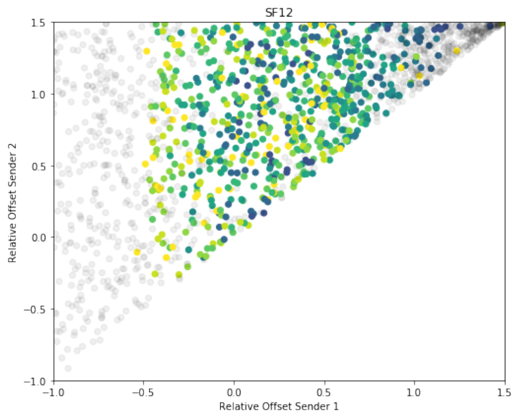
Signals to measure include `NSS` and `DIO1` on the receiving node.

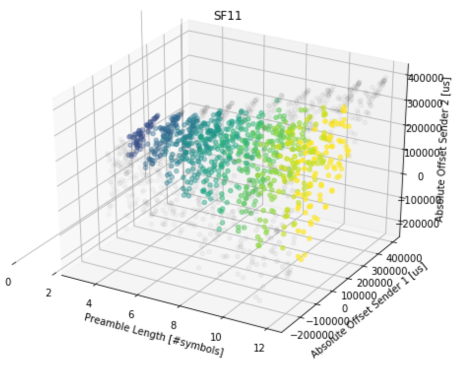
```
In [75]: %autoreload
```

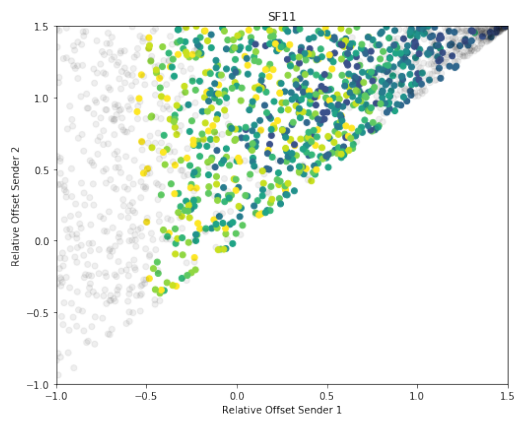
```
In [76]: %matplotlib inline
```

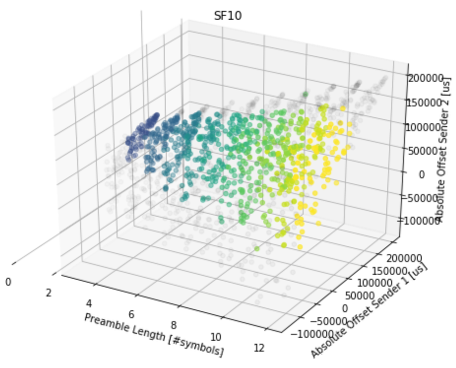
```
In [77]: from flora_tools.experiments.measure_time_sweep_tx_tx import MeasureTimeSweepTxTx
         import pandas as pd

         sweep_tx_tx = MeasureTimeSweepTxTx()
         df = pd.read_csv("../../data/{}.csv".format(sweep_tx_tx.name), index_col="sample")
         timings = sweep_tx_tx.analyze(df)
```

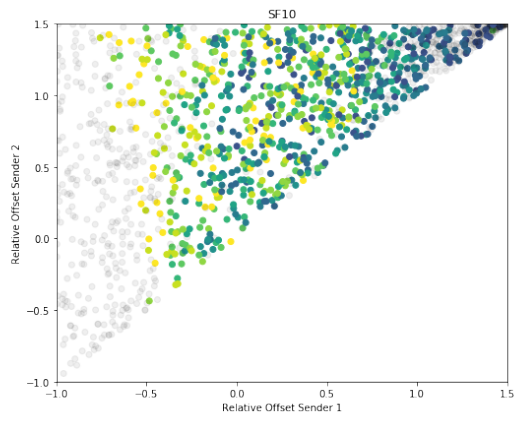


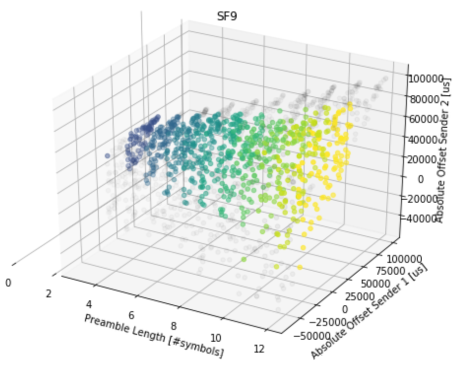


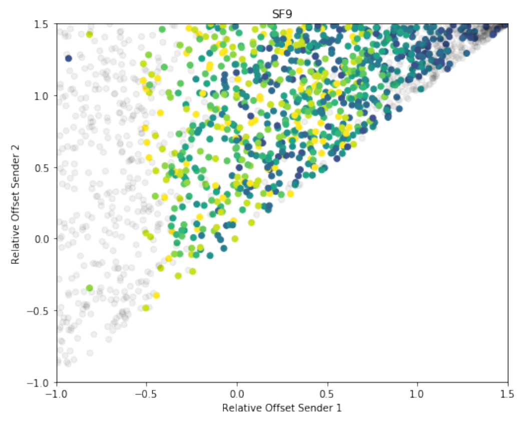

SF11
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]

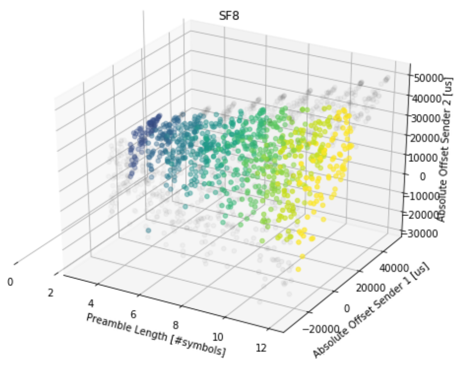

SF11
Relative Offset Sender 1
Relative Offset Sender 2

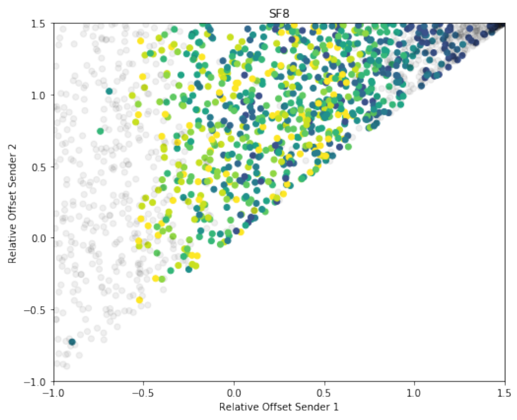

SF10
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]

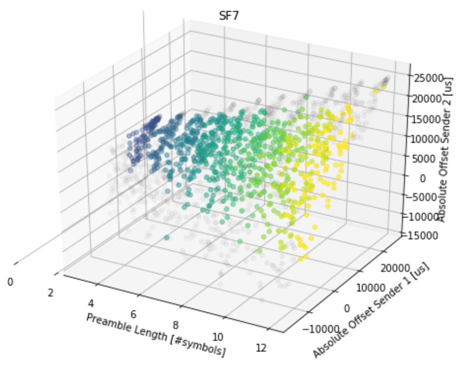

SF10
Relative Offset Sender 1
Relative Offset Sender 2

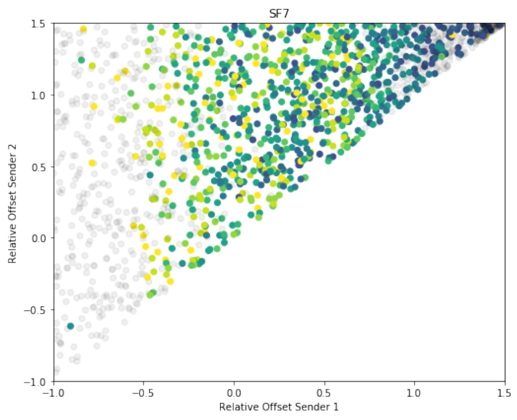

SF9
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]

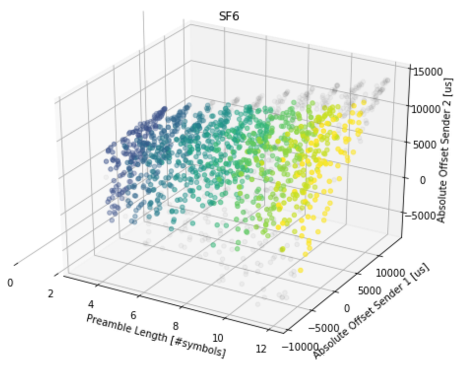

SF9
Relative Offset Sender 1
Relative Offset Sender 2

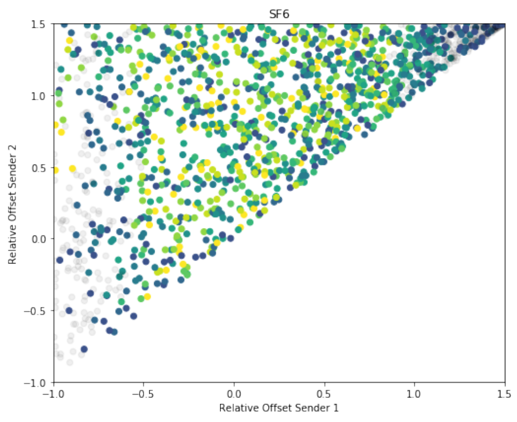

SF8
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]
0
2
4
6
8
10
12
−20000
0
20000
40000
50000
40000
30000
20000
10000
0
−10000
−20000
−30000

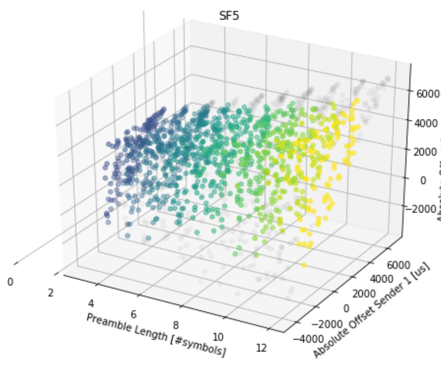

SF8
Relative Offset Sender 1
Relative Offset Sender 2
−1.0
−0.5
0.0
0.5
1.0
1.5

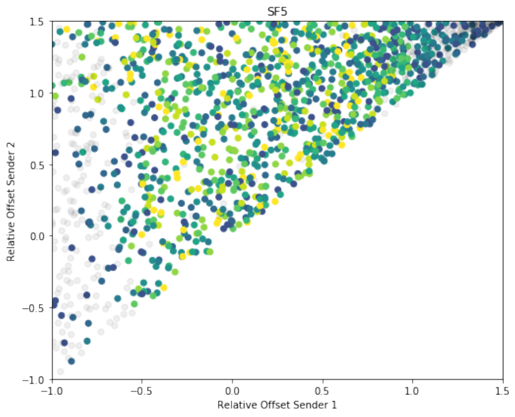

SF7
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]
0
2
4
6
8
10
12
−10000
0
10000
20000
25000
20000
15000
10000
5000
0
−5000
−10000
−15000

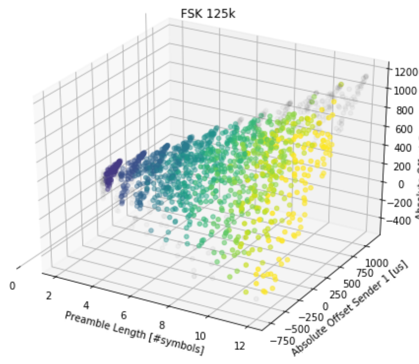

SF7
Relative Offset Sender 1
Relative Offset Sender 2
−1.0
−0.5
0.0
0.5
1.0
1.5

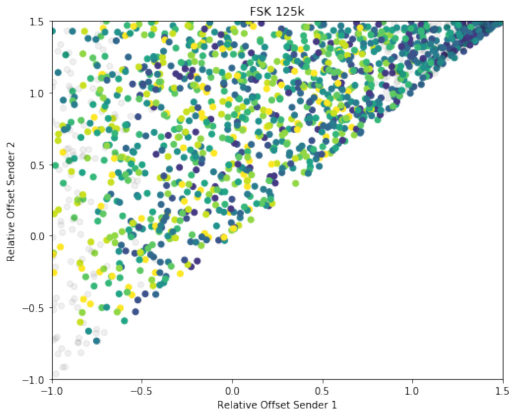

SF6
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]
0
2
4
6
8
10
12
−10000
−5000
0
5000
10000
15000
10000
5000
0
−5000

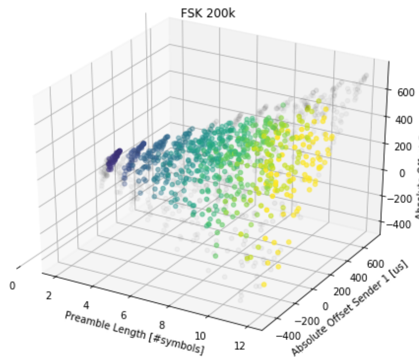

SF6
Relative Offset Sender 1
Relative Offset Sender 2
−1.0
−0.5
0.0
0.5
1.0
1.5

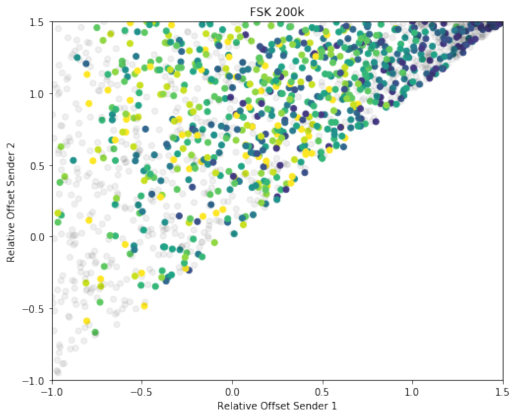

SF5
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]


SF5
Relative Offset Sender 1
Relative Offset Sender 2


FSK 125k
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]


FSK 125k
Relative Offset Sender 1
Relative Offset Sender 2


FSK 200k
Preamble Length [#symbols]
Absolute Offset Sender 1 [us]
Absolute Offset Sender 2 [us]


FSK 200k
Relative Offset Sender 1
Relative Offset Sender 2

Appendix G

Task Assignment

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering and Networks Lab

ETZ G 75
CH-8092 Zurich

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

Master Thesis Project
For
Markus Wegmann

Supervisor: Jan Beutel, Roman Trueb, Reto Da Forno

Start Date: March 2, 2018
Initial Presentation Date: March 7, 2018
End Date: August 31, 2018

Reliable 3rd Generation Data Collection

The PermaSense project develops, deploys and operates wireless sensing systems customized for long-term autonomous operation in high-mountain environments. Around this central element, we develop concepts, methods and tools to investigate and to quantify the connection between climate, cryosphere (permafrost, glaciers, snow) and geomorphodynamics.

This thesis project aims at platform integration and protocol development for the 3rd generation Dual Processor Platform (DPP) based on a recently released long-range, low-power, sub-GHz RF transceiver by Semtech (SX1262). These transceivers incorporate different modulation modes (LoRa, (G)FSK, OOK) and in this project we want to investigate if and how these modulation modes can be used jointly in a communication scheme in order to facilitate reliable data collection with variable bandwidth requirements (bytes/sec to 10s of kbytes/sec) as well as low-latency notifications for urgent events. The work bases on existing protocols (Dozer, Glossy/LWB) and other currently ongoing investigations. The application scenario is drawn from our ongoing work in environmental sensing. The design of the hardware is currently ongoing therefore the first part of the project will rely on developer boards and then transfer to the final system design and to our testbed environment on FlockLab. A successful prototype implementation shall be tested in a lab setting as well as optionally on a PermaSense field site.

Thesis Project Assignment

- Formulate a time schedule and milestones for the project (max. duration 6, months). Discuss and approve this time schedule with your supervisor.
- Familiarize yourself with relevant past works within the project and the significant literature in the field. Search for new approaches and examine which concepts and new components could be of use here. Position your work within this context.

- Explore how FSK modulated communication and LoRA-based communication can be used in conjunction to enable both high data throughput for bulk data transfers as well as low-latency notification in the same network.
- Develop a specification of a protocol system.
- Implement and evaluate a prototype system.
- Characterize the performance of this prototype in a suitable test setup.
- Perform extensive performance testing using our in-house FlockLab testbed.
- Document your project with a written report, a short initial presentation, a final presentation and if applicable a demonstration of the prototype. As a guideline, your documentation should be as thorough to allow a follow-up project to build upon your work, understand your design decisions taken as well as recreate the experimental results.

Project Organization

General Requirements

- The project progress shall be regularly monitored using your time schedule and milestones. Unforeseen problems may require adjustments to the planned schedule and milestones. Discuss such issues openly and timely with your supervisor.
- Use the work environment and IT infrastructure provided with care. The general rules of ETH Zurich (BOT) apply. In case of problems, contact your supervisor.
- Discuss your work progress regularly with your supervisor. In excess to such meetings, a short weekly status email to your supervisors is required containing your current progress, problems encountered and next steps.

Handing In

- Hand in two paper copies as well as a single PDF file of your project report including the signed plagiarism statement.
- Clean up your digital data in a clear and documented structure using the SVN repository provided.

References:

[1] https://www.tik.ee.ethz.ch/w1/images/c/ca/Student_Thesis_Guidelines_EN.pdf

[2] https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/declaration-originality.pdf

[4] Jan Beutel, Stephan Gruber, Andreas Hasler, Roman Lim, Andreas Meier, Christian Plessl, Igor Talzi, Lothar Thiele, Christian Tschudin, Matthias Woehrle and Mustafa Yuecel: PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery under Extreme Conditions. Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN), p. 265-276, April 2009.

[5] Felix Sutton, Marco Zimmerling, Reto Da Forno, Roman Lim, Tonio Gsell, Georgia Giannopoulou, Federico Ferrari, Jan Beutel and Lothar Thiele: Bolt: A Stateful Processor

Interconnect. Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys 2015), Seoul, South Korea, p. 267-280, November 2015.

[6] R. Lim et al: FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. Proc. IPSN/SPOTS 2013.

[7] Federico Ferrari, Marco Zimmerling, Lothar Thiele and Olga Saukh: Efficient Network Flooding and Time Synchronization with Glossy. Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011), Chicago, IL, USA, April 2011.

[8] Federico Ferrari, Marco Zimmerling, Lothar Thiele and Olga Saukh: Efficient Network Flooding and Time Synchronization with Glossy. Proceedings of the 10th International Conference on Information Processing in Sensor Networks (IPSN 2011), Chicago, IL, USA, April 2011.

[9] Nicolas Burri, Pascal von Rickenbach and Roger Wattenhofer: Dozer: Ultra-Low Power Data Gathering in Sensor Networks. International Conference on Information Processing in Sensor Networks (IPSN), Cambridge, Massachusetts, USA, April 2007.

Bibliography

[1] Core Writing Team, IPCC, *Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, R. K. Pachauri and L. A. Meyer, Eds. IPCC, 2014. [Online]. Available: http://www.ipcc.ch/report/ar5/syr/

[2] University of Zurich, "PermaSense," 2017. [Online]. Available: http://www.permasense.ch/en.html

[3] J. Beutel, "The PermaSense Project. Low-power Sensor Networks for Extreme Environments." [Online]. Available: https://disco.ethz.ch/courses/hs09/asn/lecture/13/20091213_wsnlecture_permasense.pdf

[4] A. Hasler, *Documentation PermaSense*, Glaciology, Geomorphodynamics and Geochronology, Department of Geography, University of Zurich, Switzerland, 2009. [Online]. Available: http://www.permasense.ch/dam/jcr:af62c668-de19-401d-9a37-8d33f80eacf3/sensordocu.pdf

[5] A. Pullini, D. Rossi, F. Conti, R. Andri, L. Cavigelli, F. Zaruba, and D. Schiavone, "Poseidon," 2018. [Online]. Available: http://asic.ethz.ch/2018/Poseidon.html

[6] F. Sutton, M. Zimmerling, R. D. Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele, "Bolt: A Stateful Processor Interconnect," *ACM*, 2015. [Online]. Available: https://www.tik.ee.ethz.ch/file/5acae22d79f04e0e9eb1022d089029d0/SZDLGGFBT2015a.pdf

[7] TIK WSN Research Group, ETH Zurich, "The Sensor Network Museum - TinyNode 584." [Online]. Available: http://www.snm.ethz.ch/Projects/TinyNode584

[8] *MSP430-CCRF development board. User's manual.*, Revision c, july 2013 ed., OLIMEX, 2011. [Online]. Available: https://www.olimex.com/Products/MSP430/Starter/MSP430-CCRF/resources/MSP430-CCRF.pdf

[9] *CC430F6137, CC430F6135, CC430F6127, CC430F6126, CC430F6125 CC430F5137, CC430F5135, CC430F5133. MSP430 SoC With RF Core*, Texas Instruments, 2009. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc430f5137.pdf

[10] STMicroelectronics, "STM32L433xx," 2018. [Online]. Available: https://www.st.com/resource/en/datasheet/stm32l433cc.pdf

[11] ECROS Technology, “Dhrystone Benchmark for MCUs.” [Online]. Available: http://www.ecrostech.com/Other/Resources/Dhrystone.htm

[12] *AN4621: STM32L4 and STM32L4+ ultra-low-power features overview*, STMicroelectronics, Mar. 2018. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/application_note/9e/9b/ca/a3/92/5d/44/ff/DM00148033.pdf/files/DM00148033.pdf/jcr:content/translations/en.DM00148033.pdf

[13] Semtech, “SX1261/2: Long Range, Low Power, sub-GHz RF Transceiver,” 2017. [Online]. Available: https://www.semtech.com/uploads/documents/DS_SX1261-2_V1.1.pdf

[14] R. D. Forno, “FlockLab Homepage,” 2017. [Online]. Available: https://gitlab.ethz.ch/tec/public/flocklab/wikis/home

[15] M. Zimmerling, “LWB Readme: Low-Power Wireless Bus (LWB),” 2016. [Online]. Available: https://github.com/ETHZ-TEC/LWB

[16] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, “Efficient Network Flooding and Time Synchronization with Glossy,” 2011. [Online]. Available: https://www.tec.ee.ethz.ch/publications-and-awards/papers-with-ecitations.html?batch_name=publications&page=0

[17] R. Lim, B. Maag, and L. Thiele, “Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems,” 2016. [Online]. Available: http://people.ee.ethz.ch/~bmaag/files/LMT2016.pdf

[18] N. Burri, “Ultra-Low Power Sensor Networks: Development Tools, Design, and Implementation,” 2010.

[19] M. Keller, “LoRa-based Data Collection on DPP,” 2018. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2018-FS/SA-2018-26.pdf

[20] C. Liao, G. Zhu, D. Kuwabara, M. Suzuki, and H. Morikawa, “Multi-Hop LoRa Networks Enabled by Concurrent Transmission,” *IEEE Access*, vol. 5, pp. 21 430–21 446, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8048465/

[21] S. Ghoslya, “All About LoRa and LoRaWAN.” [Online]. Available: http://www.sghoslya.com/p/lora_6.html

[22] Semtech, “LoRa Calculator: Fast evaluation of link budget, time on air and energy consumption.” [Online]. Available: https://www.semtech.com/uploads/documents/SX1272LoRaCalculatorSetup1_1.zip

[23] M. Knight, “Reversing LoRa. Exploring next-generation wireless,” 2016. [Online]. Available: https://static1.squarespace.com/static/54cecce7e4b054df1848b5f9/t/57489e6e07eaa0105215dc6c/1464376943218/Reversing-Lora-Knight.pdf

[24] Semtech, “AN1200.22: LoRa Modulation Basics,” 2015. [Online]. Available: https://www.semtech.com/uploads/documents/an1200.22.pdf

[25] ITU, “Recommendation ITU-R P.1406: Propagation Effects relating to terrestrial Land Mobile Service in the VHF and UHF Bands,” 1999. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.1406-0-199907-S!!PDF-E.pdf

[26] M. Bor, U. Roedig, T. Voigt, and J. M. Alonso, “Do LoRa Low-Power Wide-Area Networks Scale?” 2016.

[27] B. Ray and Link Labs, “Improving LoRaWAN Scalability,” 2017. [Online]. Available: https://www.link-labs.com/blog/improving-lorawan-scalability

[28] “AN1200.37: SX1261/2 Recommendations for Best Performance,” 2018. [Online]. Available: https://www.semtech.com/uploads/documents/AN1200.37_SX1261-2_Recommendations_for_Best_Performance_V1.1.pdf

[29] Semtech, “AN1200.40: SX1261/2 Reference Design Explanation V1.1,” 2018. [Online]. Available: https://www.semtech.com/uploads/documents/AN1200.40_SX1261-2_Reference_Design_Explanation_V1.1.pdf

[30] COVINGTON, “China Revises Rules on Commercial Encryption Products,” 2017. [Online]. Available: https://www.cov.com/-/media/files/corporate/publications/2017/10/china_revises_rules_on_commercial_encryption_products.pdf

[31] W. Companioni for Kemet, “MLCC Shortage: When You Can’t Find The Cap You Need,” 2018. [Online]. Available: https://ec.kemet.com/mlcc_alternatives

[32] Semtech, “LoRaMac-node GitHub repository,” 2018. [Online]. Available: https://github.com/Lora-net/LoRaMac-node

[33] ——, “Mbed: SX126xDVK1xAS,” 2018. [Online]. Available: https://os.mbed.com/components/SX126xDVK1xAS/

[34] FreeRTOS, “Low Power Support: Tickless Idle Mode.” [Online]. Available: https://www.freertos.org/low-power-tickless-rtos.html

[35] F. K. Gürkaynak, “PULP: An Open Parallel Ultra-Low-Power Processing-Platform,” 2018. [Online]. Available: http://iis-projects.ee.ethz.ch/index.php/PULP

[36] ST, “AN4746: Optimizing power and performance with STM32L4 Series microcontrollers,” 2017. [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/application_note/5c/cb/90/97/4b/84/4e/81/DM00216518.pdf/files/DM00216518.pdf/jcr:content/translations/en.DM00216518.pdf

[37] F. Lambrechts, “STM32Loader,” 2018. [Online]. Available: https://github.com/florisla/stm32loader

[38] Semtech, "AN1200.24: Recommended SX1276 Settings for EU868 LoRaWAN Network Operation," 2015. [Online]. Available: https://www.semtech.com/uploads/documents/an1200.24.pdf

[39] ——, "AN1200.13: Designer's Guide," 2013. [Online]. Available: https://www.semtech.com/uploads/documents/LoraDesignGuide_STD.pdf

[40] ETSI, "ETSI EN 300 220-2 (V3.2.1): Short Range Devices (SRD) operating in the frequency range 25 MHz to 1000 MHz," 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_60/en_30022002v030201p.pdf

[41] BAKOM, "RIR1003-08: Tracking, tracing and data acquisition: NRP/AP," 2018. [Online]. Available: https://www.ofcomnet.ch/api/rir/1003/08

[42] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel, "PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery under Extreme Conditions," *Proceedings of the 8th International Conference on Information Processing in Sensor Networks (IPSN)*, pp. p. 265–276, Apr. 2009.