TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Methods of Improving Wireless Communication in Home Automation and Security

Dissertation

Study programme: P2612 – Electrical Engineering and Informatics
Study branch: 2612V045 – Technical Cybernetics

Author: **Ing. Tomáš Jakubík, M.Eng.**
Supervisor: Ing. Jiří Jeníček, Ph.D.

Liberec 2022

Declaration

I hereby certify, I, myself, have written my dissertation as an original and primary work using the literature listed below and consulting it with my thesis supervisor and my thesis counsellor.

I acknowledge that my dissertation is fully governed by Act No. 121/2000 Coll., the Copyright Act, in particular Article 60 – School Work.

I acknowledge that the Technical University of Liberec does not infringe my copyrights by using my dissertation for internal purposes of the Technical University of Liberec.

I am aware of my obligation to inform the Technical University of Liberec on having used or granted license to use the results of my dissertation; in such a case the Technical University of Liberec may require reimbursement of the costs incurred for creating the result up to their actual amount.

At the same time, I honestly declare that the text of the printed version of my dissertation is identical with the text of the electronic version uploaded into the IS STAG.

I acknowledge that the Technical University of Liberec will make my dissertation public in accordance with paragraph 47b of Act No. 111/1998 Coll., on Higher Education Institutions and on Amendment to Other Acts (the Higher Education Act), as amended.

I am aware of the consequences which may under the Higher Education Act result from a breach of this declaration.

16. 9. 2022 Ing. Tomáš Jakubík, M.Eng.

Acknowledgements

I would like to thank my alma mater Technical University of Liberec for material support for this work as well as for the knowledge gained during my studies.

Similarly, I would like to thank the engineers of Jablotron Alarms a.s. for support with the development of hardware and consultation of the ideas.

Acknowledgment has to go to the developers behind Airspy, Liquid SDR and Lime Suite projects. Without inspiration and use of their open source code, some development would take much longer.

And lastly, most sincere thanks and respect to all people who will endure this work and read to the very end. Without readers, this wouldn't be worth the paper it is printed on, nor the bits used to store in.

Metody vylepšení bezdrátové komunikace v domácí automatizaci a zabezpečení

Abstrakt

Tato práce představuje možnosti vylepšení bezdrátové komunikace pro systémy domácí automatizace a zabezpečení. Většina dnešních systémů používá jednofrekvenční komunikaci. Přidání frekvenčního skákání zvyšuje odolnost proti rušení, ale přináší problémy s výdrží baterie nebo s rychlostí odezvy, které nejsou v této třídě elektroniky jednoduše řešitelné.

První metoda představená v této práci je vícekanálový přijímač pro centrální jednotku. To umožňuje senzorům spát a po probuzení neřešit synchronizaci se sítí.

Druhá metoda je kombinace vícekanálového přijímače s komunikací bezdrátových kamer. Komunikace senzorů se skryje do přenosu obrazu bez přidání dalšího rádia.

Klíčová slova: frequency hopping, frequency agility, OFDM, home automation, security system, sensor network

Methods of Improving Wireless Communication in Home Automation and Security

Abstract

This thesis presents methods of improving wireless communication in home automation and security. Most current systems use single-frequency communication. Frequency hopping improves resistivity to interference but brings problems with battery lifespan or communication delay, which cannot be simply solved in this class of electronics.

The first method proposed in this work is an all-channel receiver for the central unit. It allows the sensors to sleep and avoid lengthy network synchronization after wakeup.

The second method is a combination of the all-channel receiver with a communication of wireless cameras. The sensor communication is hidden in video transfer without additional hardware.

Keywords: frequency hopping, frequency agility, OFDM, home automation, security system, sensor network

Contents

Acronyms

3GPP 3rd Generation Partnership Project. 29, 30

AC Alternating Current. 26
ADC Analog-to-Digital Converter. 45–48, 51, 54–56, 65, 66, 68, 96, 105
AES Advanced Encryption Standard. 15
AFE Analog Front-End. 54–56, 65, 66, 68
AGC Automatic Gain Control. 44
ALU Arithmetic Logic Unit. 48, 49
ASK Amplitude Shift Keying. 26
AWGN Additive White Gaussian Noise. 37, 38, 40, 41, 70–72, 82, 84

Balun Balanced to Unbalanced (adapter). 54
BER Bit Error Rate. 37, 71–73, 77, 94
BLE Bluetooth Low Energy. 21–23, 25

CCA Clear Channel Assessment. 19, 28, 41, 44, 60, 64, 67, 85, 86, 94
CDC Communications Device Class (USB). 58
CE Counter Electrode (3-lead electrochemical cell). 56
CMSIS Cortex Microcontroller Software Interface Standard. 105
CMSIS-DSP CMSIS - DSP (library). 48, 50, *see* DSP
CORDIC Coordinate Rotation Digital Computer. 50
CRC Cyclic Redundancy Check. 36, 37, 50, 51, 61, 62, 78, 87
CSI Camera Serial Interface. 88, 89, 106
CSS Chirp Spread Spectrum. 27
CU Central Unit. 15–22, 25, 31–33, 44, 47, 52, 54, 55, 63–65, 67, 68, 70, 84, 85, 87, 88, 90, 91, 93, 96–98, 106

DAC Digital-to-Analog Converter. 46, 54, 55, 65
DBPSK Differential Binary Phase Shift Keying. 28, 81, *see* PSK
DC Direct Current (not alternating, constant value). 46, 78, 91
DMA Direct Memory Access. 47
DSI Display Serial Interface. 88
DSP Digital Signal Processing. 61, 78
DSSS Direct Sequence Spread Spectrum. 22, 23
DVB-T Digital Video Broadcasting - Terrestrial. 25, 33, 43, 44, 48, 86, 96

EC-GSM-IoT Extended Coverage GSM IoT. 30, *see* IoT
eDRX Extended Discontinuous Reception. 30

EGPRS Enhanced General Packet Radio Service. 30

FCC Federal Communications Commission. 23
FEC Forward Error Correction. 22, 61, 78, 82, 84, 87
FET Field-Effect Transistor. 54
FFT Fast Fourier Transform. 34–39, 47, 49–51, 58, 61, 70, 75, 77, 79, 83, 88, 97
FHSS Frequency Hopping Spread Spectrum. 22, 31–33, 36, 70
FIFO First In, First Out. 37, 50, 61, 88
FIR Finite Impulse Response (filter). 61
FMC FPGA Mezzanine Card. 58
FPGA Field Programmable Gate Array. 58–60, 97
FSK Frequency Shift Keying. 23, 25, 26, 34, 35, 62, 69, 98
FUOTA Firmware Update Over-the-Air. 28

GFSK Gaussian Frequency Shift Keying. 18, 20, 24, 26–28, 31–33, 35, 69, 77, *see* FSK
GMSK Gaussian Minimal Shift Keying. 22, 24, 29, 35, 36, 39–42, 49–52, 58, 61, 62, 65, 69–72, 75, 77, 78, 81, 82, 84–88, 90, 91, 93, 94, 96–98, 105, *see* MSK
GPS Global Positioning System. 28

HD High Definition (picture or video). 22, 24
HRP High Rate Pulse repetition frequency (IEEE 802.15.4). 24

I2C Inter-Integrated Circuit. 46, 54
IDE Integrated Development Environment. 47
IEEE Institute of Electrical and Electronics Engineers. 22
IF Intermediate Frequency. 48, 50, 51
iFFT Inverse Fast Fourier Transform. 62, 63, 78, 83
IIR Infinite Impulse Response (filter). 61, 79
IoT Internet of Things. 15, 27, 31, 98
IQ In-phase/Quadrature. 24, 35, 43, 77, 98

KET Department of Materials and Technology (Katedra materiálů a technologií). 52

LBT Listen Before Talk. 19
LCD Liquid Crystal Display. 88
LDC/HR Low Duty Cycle / High Reliability. 36
LDO Low-Dropout (regulator). 54, 56, 57, 68
LoRaWAN LoRa Wide Area Network. 18, 26–28, 30, 32, 33
LPWAN Low-Power Wide-Area Network. 26–30, 98
LR-FHSS Long Range FHSS. 27, 32, *see* FHSS
LRP Low Rate Pulse repetition frequency (IEEE 802.15.4). 24
LTE Cat-M1 LTE for Machine-Type Communications Category M1. 30

m-sequence Maximum Length Sequence. 61, 78, 79, 90

MCU Micro Controller Unit. 18–20, 31, 33, 43, 45–49, 51, 52, 54–56, 64–66, 96, 97
MIMO Multiple Input Multiple Output. 22
MSK Minimal Shift Keying (special case of FSK). 35, *see* FSK

NB-IoT Narrowband IoT. 30, *see* IoT
NFC Near Field Communication. 24

OFDM Orthogonal Frequency Division Multiplex. 17, 22–24, 30, 62, 69–73, 75, 77–79, 82, 86–91, 93, 94, 97, 98, 105, 106
OFDM-MFSK OFDM based M-ary FSK. 62, 69, 98, *see* FSK
OFDMA Orthogonal Frequency Division Multiple Access. 69, 98
OoB Out of Band (emissions or limit of emissions). 62
OOK On Off Keying. 23
OSI Open Systems Interconnection (model). 33, 64, 90, 98

PAPR Peak-to-Average Power Ratio. 62, 90
PC Personal Computer. 43, 51, 59, 60, 83, 87, 88, 96
PCB Printed Circuit Board. 32, 43, 46, 52, 54, 56
PER Packet Error Rate. 37–42, 44, 51, 83, 94, 97
PHY Physical Layer (OSI L1). 22, 24, *see* OSI
PIR Passive Infrared (sensor). 15, 16
PLCP Physical Layer Convergence Procedure. 78
PLL Phase Locked Loop. 48
PoC Proof of Concept. 52, 54, 55, 58, 65, 68
POE Power Over Ethernet. 17
PSK Phase Shift Keying. 69, 70, 79, 82, 87, 91, 98
PWM Pulse-Width Modulation. 56

QAM Quadrature Amplitude Modulation. 98

RAM Random Access Memory. 47
RBW Resolution Bandwidth. 62, 91
RE Reference Electrode (3-lead electrochemical cell). 56
RF Radio Frequency. 24, 31, 32, 43–46, 51, 52, 54, 67, 96, 97
RFID Radio Frequency Identification. 24
RMS Root Mean Square. 60, 63, 89
Rx Receive. 26, 29, 39, 51, 58–60, 63–65, 69, 79–81, 85, 86, 88, 90, 96

SAW Surface Acoustic Wave. 54
SDR Software Defined Radio. 5, 24, 30, 33, 34, 43, 44, 48, 52, 58, 59, 61–63, 65, 69, 78, 87, 88, 90, 94, 97, 98, 106
SNR Signal to Noise Ratio. 38–40, 42, 43, 71, 72
SPI Serial Peripheral Interface. 52, 63
SRD Short Range Device. 20, 22, 85
SSH Secure Shell (protocol). 88
SUN Smart utility network (IEEE 802.15.4). 24

TCP Transmission Control Protocol. 57
TS-UNB Telegram-Splitting Ultra Narrowband. 29, *see* UNB
TSMA Telegram Splitting Multiple Access. 29
TUL Technical University of Liberec (Technická univerzita v Liberci). 5
Tx Transmit. 18, 22, 24–26, 29, 39, 44, 52, 59, 60, 62–65, 67–69, 80, 83, 85, 87, 88, 90, 91, 93

UART Universal Asynchronous Receiver Transmitter. 50, 51
UNB Ultra Narrowband. 28
USB Universal Serial Bus. 43, 46, 58, 59, 87–89
UWB Ultra Wideband. 23, 24

WE Working Electrode (3-lead electrochemical cell). 56
WUR Wake-Up Radio. 23

ZCU University of West Bohemia (Západočeská univerzita v Plzni). 52

1 Introduction

Home automation and security is a specific area of consumer electronics. A security system usually consists of a Central Unit (CU) and many independent devices which need to be small and cheap. There are Passive Infrared (PIR) sensors, magnetic door contacts, smoke detectors, key fobs, smart light switches and many more. All can be connected to the CU via a wired bus or wireless network. This thesis is focused only on wireless communication.

Wireless communication in the area of home automation and security is advancing much slower than in other areas of consumer electronics. There are several difficulties [1] that don't allow quickly reusing foreign ideas. Requirements of very low energy consumption, short communication delay and relatively long range stand against each other. Current wireless networks are not usable in home automation and security for various reasons:

- Modern industrial technologies are several orders of magnitude faster than what is needed, but cannot be powered by batteries [2].
- Modern consumer technologies have high bandwidth and a nearly acceptable delay, but still consume too much current [3].
- Modern IoT technologies can live a long time on a small battery [4], but the delay before the information gets processed is neither usable for automation nor for security systems.

The hardware used in this area has improved over the past decade, but it barely matched the increasing requirements of security. CR2032 battery remains to be a very limited reservoir of energy. Security, on the other hand, has seen constant development in attacks and countermeasures. Older garage door remotes used static codes and can be easily opened by the de Bruijn sequence in 8 seconds [36]. Some manufacturers sell these even today, but it should be avoided if possible. The only viable solution today is AES, possibly improved by an asymmetric key exchange. The cheap and small devices need enough power for computing and more complicated exchange of packets. There is very little space for improvements in modulation and communication techniques.

A common solution nowadays is still a single-frequency network that is susceptible to interference and doesn't efficiently use the available spectrum. The purpose of this work is to design and verify new methods which would allow new communication techniques, with emphasis on frequency agility, while satisfying the requirements for this area of electronics.

2 Goals

The main goal of the thesis is to present ways to improve wireless communication in home automation and security. The improvements need to satisfy both technical needs such as latency, power, datarate, range, or size of the devices as well as financial limits. There might be a more elegant solution, but if it would increase the price of a sensor by an order of magnitude, it is not viable. If Moore's law should hold, we can discuss at least those solutions which will probably drop into the available budget in a foreseeable future.

2.1 Example Situation

The example situation is a small house with one larger CU and many small low-power sensors. Sensors can be magnetic contacts guarding closed doors, PIR detectors for person movement, acoustic glass-break detectors, flooding detectors, smoke detectors, light switches and many more. The CU is powered by mains at all times and has a large backup battery in case there is a power outage or in case the power connection is intentionally cut. The size of the CU's battery is designed to keep the system running only for a few days, sometimes even only hours. On the other side of communication are sensors that need to survive many years on a small battery. The delay between triggering any sensor and information being available in the CU needs to be at most a fraction of a second.

At least one device in the network is usually the keypad. This device does not communicate directly with the sensors but allows the user a normal day-to-day operation of the system. The user interface can be composed of several kB of texts. That puts more constraints on the available network datarate. The latency of the user interface should be a fraction of a second, similar to the sensors. A keypad can usually hide a somewhat larger battery in exchange for output functionality.

Another output device is a siren. An outdoor siren needs a considerably larger battery to be able to drive the 100 dB piezo element even when the outside temperature is $-20\,^{\circ}\mathrm{C}$. The output devices are usually time synchronized with the CU to periodically open receiving windows. When the device is synchronized, it requires only a time-frequency chart to add pseudorandom frequency switching. It makes most of this thesis not applicable to this class of devices, but even synchronous devices can use the ability to randomly switch

frequencies at will. Either way, it adds more constraints on compatibility with the rest of the wireless devices.

The network needs to reach over a small family house. Having routing between sensors is not practical for various reasons. On one side, the network is set up in advance and most of the devices do not move. That would allow storing paths and time synchronization of an optimal tree network permanently in all devices. On the other side, all devices would have to open their receiving windows at a precise time. Using only devices with a larger battery for routing would not bring many benefits as most devices have small batteries. The need to synchronize all devices will increase power consumption and latency. On top of that, there would be retransmissions discharging unevenly the devices near CU.

Routing or a mesh network would be a viable option in home automation, where there are a lot of output devices connected to the mains supply and the user can quickly replace the batteries of the rest. The preferred way for a security system is to cover the entire house with one or at most a few radio hubs. These radio hubs can be connected to the CU by a high speed wired bus or they can be directly embedded inside of the CU. In the scope of this text, the CU is synonymous with the radio hub and the connection between them is neglected.

Modern systems can also optionally provide visual verification. When an intrusion is detected, the system makes one or more pictures of the situation. A homeowner or the security agency gets a picture and can decide whether the situation is a real threat (eg. burglar) or a false alarm (eg. misbehaving pet). Even though there are such products available, transmitting pictures over the sensor network is not a viable solution. It takes a minute to carry a 640×480 pixel low resolution picture [37]. In a model situation, a person entering the building will be gradually triggering low-power sensors while the cameras start transmitting video. The low-power sensor network has to work together with the high-bandwidth link of the cameras and not interfere. The needs of the sensor network are almost the opposite of the needs of the camera.

Security cameras require a lot more power, so it is common to use Power Over Ethernet (POE) or a wireless connection with a power adapter and a small battery for backup. It is a part of the security system, so it needs tampering detection and connection to the secure network, even if there already is Wi-Fi for video. Having two radios is more complicated, expensive and adds interference. In particular, interference between Orthogonal Frequency Division Multiplex (OFDM) and frequency hopping is known for Bluetooth and Wi-Fi [5].

2.2 Current Consumption

Common wireless sensors can consume between $0.1\,\mathrm{mW}$ and $0.2\,\mathrm{mW}$ which is being matched by the design of power sources [6]. But depending on the

context, contemporary home security sensors can go almost an order of magnitude lower. Some simpler sensors can live for over two years on a single CR2032 coin cell battery [38]. Context, in this case, is to comply with grade 2 of EN 50131 [39]. To arm the system, the latest message from the sensor needs to be 20 minutes old or newer. There needs to be at least one transmitted message from the sensor and its acknowledgment received by the sensor over the 20 minute period. An automation device doesn't need to periodically communicate at all and its consumption can be even lower.

The CR2032 battery has a capacity around 220 mAh which gives a continuous current of

$$I = \frac{0.22}{24 \times 365 \times 2} \approx 13\,\mu\text{A} \tag{2.1}$$

for two years of service. During this time, the battery has a voltage between 3 V and 2 V which gives an average power

$$P = 13 \times 2.5 \approx 31\,\mu\text{W} \tag{2.2}$$

With some safety margin, it is less than 10 µA of cumulative current consumption for radio, MCU and the sensor. Some devices, such as magnetic contacts hidden in the window frame, can also be deployed in a harsh environment of high humidity or sub-zero temperatures that can reduce the battery capacity significantly.

Part of this current is consumed all the time by the sleeping MCU and radio, part of the current is needed to keep the actual sensor running and very little current is remaining for wireless communication. Common Gaussian Frequency Shift Keying (GFSK) transceivers consume more than 10 mA when active [40]–[42], which translates to only a small amount of short packets in either direction.

2.3 Communication Delay

Another important constraint is the delay between triggering the sensor and a reaction in the CU. A light switch needs to turn the lights on in a fraction of a second. A smoke detector cannot have any unnecessary delay when a fire is detected.

Some wireless technologies take time to get from a sleeping state to a ready state in which information can be passed over. This is especially true for frequency hopping networks. The hopping device first needs to find the correct channel, learn the hopping sequence and synchronize with the other side before sending any information.

Other wireless technologies are intentionally designed for applications that are not time critical and in exchange are optimized for battery consumption and range. For example, LoRa Wide Area Network (LoRaWAN) device can Transmit (Tx) a packet at any time, but the packet is acknowledged after one

second at the earliest [43]. Any lost packet means a delay in the range of seconds.

In our case, the system should be stateless and quick to respond to the device. A light switch, which is completely powered off, should be able to quickly wake up and start transmitting useful information. The wireless hub has to be able to immediately verify and acknowledge the message. If any of those messages get lost, the device needs to quickly repeat the message. There cannot be any exchange of packets negotiating parameters of the communication. There is also no time for establishing encryption and message authentication, but such algorithms already exist and are beyond scope of this work.

2.4 Range and Cohabitation

Communication from sensors to the CU needs to work over an area of a one-family home. Datasheet values can be a few hundred meters in an open area [38] and in reality even more. The range is much shorter when considering multipath propagation and other indoor effects. That is the main reason for trying to bring principles of frequency agility and hopping into this area [7].

Another reason is the cohabitation of multiple systems from the same manufacturer. In especially bad conditions, two systems will be far enough that they will not detect each other by their Listen Before Talk (LBT) and Clear Channel Assessment (CCA) tools, but they will add too much interference to each other's messages. The problem can be worse if both systems use the same timing principles. Using many frequency channels for hopping will significantly reduce the risk of collision in these situations.

The range is also one of the significant reasons for using the sub-GHz frequencies in these systems.

2.5 Size and Economic Aspects

The security system is composed of one larger CU and many small sensors. The price of these sensors can add up quickly to an amount comparable to other household reconstruction works. The price of the sensors is an important aspect, so the system is affordable to many potential customers. There is very little space to improve the hardware of the radio or MCU in sensors.

The cost of the CU is important as well, but the increase is not multiplied by the number of sensors, so a larger increase can be tolerated. The CU has also much faster MCU, sometimes even running a full operating system, which could handle some computation of the radio.

Some of the devices need to be hidden, which puts constraints on their size. Other devices are directly in sight and customers require small size together with fashionable design. Antennas have to be hidden inside and very small which puts even more restrictions on wireless communication.

3 State of the Art

3.1 Common Solution

One simple solution for a wireless network for home automation or security system is to use a single frequency channel with GFSK modulation. All devices at all times use one narrow predefined channel. These systems are common in the sub-GHz Short Range Device (SRD) frequency band, known in Europe as 868 MHz band.

The usual solution of an asynchronous sensor takes these steps:

1. A sensor is sleeping and consumes a few µA from its battery. A mains-powered CU is always receiving. Consumption of its radio, although less important, is in tens of mA, usually well below other components of CU.
2. The sensor needs to communicate with CU. Either it detects a change or it needs to report its presence. The change can be an open window, a change in temperature, flipping the light switch or any other event.
3. Sensor's MCU and GFSK radio both wake up. Sensor's consumption rises to tens of mA.
4. The sensor sends a frame which is received by the CU.
5. CU processes the information. Appropriate action can be taken immediately.
6. CU sends a response frame which is received by the sensor to verify successful communication. If the sensor doesn't get a response, it repeats from step 4.
7. Sensor goes back to sleep with consumption of a few µA.

The whole communication takes only a few tens of ms. In case of an unplanned event, the information gets to CU in less than 100 ms even with possible packet repetition. Security devices need to periodically report their presence, so depending on the system configuration this can take place several times an hour [39]. The average cumulative consumption from the battery can be close to the few µA sleeping value. The automation device does not need to

periodically report. It needs to communicate only when needed, perhaps a few times a day.

Communication from the CU to the sensor can happen only after a message from the sensor. The system can be optionally improved with output devices. Either by a wake-up radio [8], if its range is sufficient, or by synchronization. But synchronized devices can have an order of magnitude larger consumption.

This solution cannot be easily improved by frequency hopping between frames. The sleeping slave has no way to know which channel is currently in use. In step 4 it cannot just send a packet to CU, because the CU is most likely listening on a different channel. Finding the correct channel takes time and causes an unacceptable delay in communication. Synchronization of sensors the same way as output devices is possible, but it requires frequent beacon messages and a precise oscillator. Both increase consumption beyond one CR2032 battery.

It is less complicated to add basic frequency agility. The sensor can repeat transmissions on a different channel or CU can do channel scanning. The number of used channels needs to be quite low, it would otherwise affect communication delay or require an unreasonably long preamble.

3.2 Existing Short Range Technologies

Several existing wireless technologies are intended to work over an area similar to a small house. Most of them are proven by decades which is a large bonus for a reliable device such as a smoke detector. Some are getting close to the requirements of home automation and security, but various parameters are still making it difficult to use.

3.2.1 BLE

Bluetooth, especially the newer Bluetooth Low Energy (BLE), is popular for short-range communication in consumer electronics. It provides various settings compromising current consumption and response speed. Compromises that make the technology universal, make it less than ideal for a sensor network.

BLE was developed independently from "classic" Bluetooth and was integrated into it in version 4.0. Version 4.0 of the standard uses three fixed channels for advertising to synchronize two devices together. Synchronized devices can then hop over all 40 frequency channels. These channels are spread over the 2.4 GHz band, separated by 2 MHz. The fixed nature of the advertising removes a bit of spread spectrum advantages. Frequency-specific noise or other independent Bluetooth devices on those three channels can complicate communication.

Version 5.0 added Extended Advertising which can use all 40 BLE channels. It can send only a small header on the three fixed channels and the rest

of the data on any other channel. This helps to ease the traffic on the three fixed channels. This version also added LE Coded Physical Layer (PHY) with Forward Error Correction (FEC) to increase the range.

BLE uses Gaussian Minimal Shift Keying (GMSK) modulation at 1 or 2 Msample/s. BLE transmitter can use at most +10 dBm, depending on its class, which is less than +14 dBm available in the European sub-GHz SRD band. Together with the higher frequency, it may be very difficult to deploy a BLE network over a whole house.

It can reach transfer rates over 1 Mbit/s. That is almost ready for High Definition (HD) video streaming, but not reliable. For a wireless camera, it would need to be supplemented by Wi-Fi, which brings the already mentioned compatibility problems [5].

The BLE design doesn't allow both quick response from the sensor to CU together with µA consumption. Optimization for communication delay [9] increases consumption. The consumption requirements are met only if the sensor is completely shut down most of the time and needs to connect in order to Tx data. Even without the use of hopping, the connection latency is too high [10]. Yet, there are ideas to improve BLE connection mechanisms [11] and BLE might get into the required delay in future versions.

3.2.2 IEEE 802.11

IEEE creates various sets of standards for wireless communication. The most known set IEEE 802.11, known under the brand name Wi-Fi, was developed for the wireless connection of user devices to the Internet. Orientation for high datarate means that these protocols are great for the transport of large amounts of data but very bad for low-power sensors. Consumer devices have batteries designed to last for days or sometimes hours and not years. Sensor network built on top of IEEE 802.11 would be several orders of magnitude over the power consumption limit [3].

Datarates of IEEE 802.11 range from a few Mbit/s to several Gbit/s. The lower values can support at least one wireless camera and the newer modes would easily cover many.

The first versions of the standard used Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS), but were quickly outdated. Most of IEEE 802.11 is built on the OFDM technology. Recent standards add Multiple Input Multiple Output (MIMO) principles that use multipath propagation to push the datarates even higher. Multipath propagation usually complicates communication inside buildings, but with the use of multiple antennas, it can be turned into an advantage.

A new addition is also a set of light-based protocols IEEE 802.11bb marketed as Li-Fi. It is useful for short-range connections not affected by electromagnetic interference. In the case of a security system, it would be hard to eavesdrop or jam the signal from the outside. As a connection of sensors and automation in a single room, it might be useful.

There is an IEEE 802.11af that uses unused parts of the TV spectrum. The future of this standard might be short-lived as there are already hints about canceling the TV broadcast in favor of mobile networks. Even if it wasn't entirely true, the frequencies allocated for TV are slowly diminishing.

Another outlier is a Wake-Up Radio (WUR) using On Off Keying (OOK) modulation. This low-power radio can be receiving and able to wake up the sleeping main radio to achieve a short communication delay with low power consumption. The consumption of two radios in one is still too much for a security system [12]. A variation of low-power radio for a sensor which doesn't interfere with regular OFDM for a camera would be beneficial. But in this case, the WUR must be complemented with full OFDM radio and able to send a legacy OFDM preamble, so it is detected by other legacy devices.

IEEE 802.11ah

This part of the standard branded as Wi-Fi HaLow moves the OFDM technology into the sub-GHz frequency band. First routers and device modules are beginning to appear on the market.

It is too early to definitively comment on sensor consumption, but it seems that a device in its deep sleep state alone has a consumption of 20 μA (unofficial specification for NRC7292). The cumulative consumption of the entire sensor operation will be even higher and surely over the budget of a CR2032 battery.

High datarate of this technology will be great for a more complicated sensor or a security camera. On the other side, wide adoption of this standard may bring a lot of traffic to a relatively empty sub-GHz spectrum. Low-power sensors may have a much harder time competing for available clear channel.

3.2.3 IEEE 802.15.4

Especially for home automation and security, there is an IEEE 802.15.4 family with several proprietary and perhaps a few open technologies. It is mostly known for the ZigBee standard, but also WirelessHART, 6LoWPAN, Thread and more build on top of IEEE 802.15.4. This family can use spread spectrum, mostly DSSS, but also OFDM and Ultra Wideband (UWB) [44].

Parameters of these technologies can fit inside the requirements. Comparison with BLE [13] shows that ZigBee has larger current consumption in a cyclic sleep scenario, but the communication process is much shorter. The delay from sleep to a functional state is acceptable.

The FCC regulations in the USA require the use of spread spectrum in the sub-GHz frequency band [45]. That is perhaps one of the reasons for DSSS modulation. Current consumption for this kind of receiver can be equal to simple Frequency Shift Keying (FSK) modulation and the resistance to narrowband interference is tempting, but this kind of spread spectrum doesn't improve on interference between multiple instances of the same system. Plus

the wider bandwidth means there are a lot fewer available channels and more opportunities for collision.

SUN OFDM PHY

An OFDM communication PHY was added to this group of standards with the name Smart utility network (SUN) OFDM. At least one sub-GHz OFDM RF transceiver is already available and more are planned. When this technology reaches the consumption and price of a common GFSK transceiver, it will be a great option for the sensors. So far both parameters [46] are two or three times larger than a common GFSK transceiver [42].

This standard allows various datarate settings. Only the highest values would allow an HD camera connection.

Additionally, In-phase/Quadrature (IQ) interface of these chips (where it is user accessible) opens possibilities of Software Defined Radio (SDR). It could be a reasonably priced solution for the connection of a camera and GMSK sensors through one radio.

LRP and HRP UWB PHY

Two promising technologies are in development, Low Rate Pulse repetition frequency (LRP) and High Rate Pulse repetition frequency (HRP) UWB PHY. Both technologies use very short pulses with a bandwidth of around 500 MHz.

Apart from low-power data communication, it enables ranging with 10 cm accuracy and, in the case of HRP, a detection of an angle of arrival. The exchanged packet sequence can be short, only a few ms, and the hardware consumes less than a µA [47], [48] when powered down, so cumulative power consumption should be comparable to a single-frequency GMSK sensor.

LRP should require 125 times less power [14] with similar range as HRP. On the other hand, LRP allows only a datarate of up to 1 Mbit/s, while HRP's 6.8 Mbit/s might be enough to transfer video.

Coverage area of this technology might be smaller than sub-GHz GMSK, but estimations are over 200 m [14] which would suffice. In some cases inside buildings, the spread spectrum UWB might even be better.

3.2.4 Backscatter

In the area of low-power communication, there is ongoing research on backscatter technology [15]. Well-known and already widely used is Radio Frequency Identification (RFID) and Near Field Communication (NFC), where one communicating device has no own power at all.

There are multiple ways to modify the electromagnetic signal coming from another source to Tx data. Ambient backscatter uses Wi-Fi, TV broadcasting and other signals which are available all around us. It would be a candidate for home automation and security sensors when range, reliability and delay reach the necessary requirements.

This system could have a problem with reliability. If a neighbor's Wi-Fi or a far away DVB-T tower is used as a source of signal for communication, it can fail just when the security system needs it. Providing an own ambient signal can be technically more difficult. It would probably require two separate devices instead of one CU, one transmitting ambient carrier wave and the other receiving. This problem would have to be solved to use backscatter in home automation and security.

3.2.5 Mesh

A mesh network is a way to cover a larger area with short-range links. The sensor needs to transmit the message only to its neighbor and not all the way to the CU. It could fix the limited range of 2.4 GHz BLE communication.

The problem for the security system could be the volatility of the routing algorithm. When a smoke is detected, the information must get to the CU and to sirens. Adding hops that will be selected on the fly might not be reliable enough. A properly installed security system should have a map with the location of all devices. This information could be used to plan and fix the routing paths during installation.

Consumption of the network will be increased by the repetition of the messages. This effect must be balanced by lower transmitting power in the nodes. Otherwise, it would be more beneficial to increase Tx power and disable meshing.

The meshing network needs to be synchronized. All devices need to wake up at the right time to forward messages. Periods of sleep need to be relatively short, so a message can get from the sensor to CU quickly. Synchronized, periodically waking devices will have larger consumption than sleeping sensors.

Randomly placed devices will deplete the batteries of unlucky nodes. A node that is close to CU will bear the most load from its neighbors [16]. Batteries in a security system are commonly not user replaceable, so uneven discharging would complicate maintenance. The routing algorithm needs to tackle this issue.

Power problems can be avoided if routing is done only by mains powered devices such as automation relay or smart light bulb. Yet, for a security network, routing devices need to have their own backup in case mains power is interrupted. There might not be enough space for batteries in a small device such as a light bulb.

3.2.6 Other Less Known Systems

There are many more technologies for home automation or security. Some are single-frequency systems, some are trying to adopt frequency hopping or agility with the usual disadvantages and some have other inventions.

Z-Wave is a proprietary mesh network for home automation. It uses up to three channels in the unlicensed sub-GHz frequencies with FSK modulation.

Insteon combined AC powerline communication with proprietary sub-GHz FSK. Unfortunately, the company went out of business in 2022 and was forced to shut down the cloud functionality of its devices.

IQRF uses GFSK on a single sub-GHz channel, selected in configuration. It combines a synchronous mesh network with asynchronous sleeping sensors.

PowerG adds some channel hopping to its sub-GHz proprietary GFSK technology. It uses a synchronized network and hops 64 times a second. It uses 4 channels in 868 MHz band and 50 channels in the 915 MHz band. Different from other technologies, PowerG also offers security systems and not just home automation.

EnOcean is a set of standards for sub-GHz Amplitude Shift Keying (ASK) or FSK communication optimized for energy harvesting. Sensors can use energy from a flip of a switch, a change in temperature or a small solar panel to send several short packets. It is a great solution for home automation. Security device would have to harvest enough energy to periodically report its presence even at night.

DASH7 is a sub-GHz GFSK technology with ad-hoc synchronization. When a gateway wants to communicate with a sleeping device, it will continuously send synchronization packets for several seconds. The battery-powered device is duty-cycling its receiver to detect that and synchronize. There is also an option for a gateway to scan multiple frequency channels. The device can then use any of the scanned channels to transmit.

KNX RF is a set of sub-GHz GFSK protocols. There is BiBat with synchronous devices that Receive (Rx) and Tx in defined time slots. There is KNX RF Multi with 2 slow and 3 fast channels which are periodically scanned for basic frequency agility. The frames have a long preamble, 500 ms for slow channels, which can easily be received by scanning or a duty-cycled receiver.

Wireless M-Bus is a sub-GHz GFSK standard for reading out utility meters. There are various modes of operation with several baudrates. Some of them are unidirectional from a sensor to a master. Bidirectional sensors always initiate communication to keep low power consumption.

ANT is a 2.4 GHz GFSK personal area sensor network primarily intended for fitness trackers. One master node and one or more slave nodes open synchronized channels and exchange information only once in the channel period, defaulted to 4 Hz. Each opened channel can use a different frequency. If needed, it can use frequency agility on selected 3 out of all 125 frequency channels. Another mode is Continuous Scanning where one node is continuously receiving a single frequency for messages from other nodes.

3.3 Existing LPWAN Technologies

LoRaWAN, Sigfox and more popular technologies are in the category of Low-Power Wide-Area Network (LPWAN). These networks are optimized for the

operator and gathering of IoT data. The low-power aspect and connection of many devices into one gateway were balanced by higher latency. The tradeoff is visible in most LPWAN technologies [17].

Mostly, the delay is not acceptable for a home security device. But even if the network would be able to reliably deliver the response in a fraction of a second, any component on the way to the cloud and back could fail. We have seen outages even of the biggest internet companies. Home automation and security systems will mostly prefer a network that is centered inside the protected house and will continue to work even when the internet connection fails.

3.3.1 LoRaWAN

At least two large-scale networks were recently deployed over Europe and more or less over the whole world. In the case of LoRaWAN, which is open, more than a single provider may be available at any given place.

These networks are designed for IoT nodes with very low power consumption. The node consumption can get to tens of μA [4], almost as low as simple GFSK system while communicating over much longer distances, but probably over the CR2032 limit. The involvement of the cloud means that it is good for a long-term data gathering, but not the right technology for home automation nor for security.

The LoRa modulation, used as a base for the LoRaWAN network, can be licensed for use in a proprietary system. It uses a Chirp Spread Spectrum (CSS) with multiple partially orthogonal spreading factors and bandwidths that allow balancing data rate and available link budget. LoRaWAN uses bandwidth of 125, 250 and 500 kHz in some regions, but the hardware is capable of various other values. LoRaWAN uses sub-GHz unlicensed spectrum.

There is also a Long Range FHSS (LR-FHSS) modulation usable for uplink. It was added to improve the congestion of many devices at one gateway. Compared with LoRa modulation, it can support one or two orders of magnitude more devices per gateway [18]. It solves the bottleneck, especially with the prospect of using satellites instead of ground base stations.

The LR-FHSS splits one channel into many subchannels and uses a receiver capable to listen to all subchannels of the channel. It means that the device doesn't need to be synchronized and the gateway can receive many hopping sequences at once. The device sends 1 to 4 copies of a header on different subchannels with a 9 bit selection of a hopping sequence. The gateway needs to receive at least one of the headers to synchronize and follow the device's hopping. The rest of the packet uses a coding rate of $1/3$ or $2/3$ to be able to decode data even with a loss of many packet fragments.

LoRaWAN defines three classes of devices [43]. **Class A** is the simplest class, just an asynchronous sensor. It needs to be implemented on all devices. When the device has something to send, it transmits the uplink message and then opens two receive windows for acknowledge and a downlink message. The

device checks the availability of the channel by CCA algorithm only in regions where it is necessary. In Europe, LoRaWAN uses a duty cycle limit instead. The two receiving windows start 1 s and 2 s after the end of the device's transmission. Downlink is optional and discouraged as the gateway also needs to limit its transmitting duty cycle.

Delay before the two receiving windows is caused by connection with the cloud. The gateway needs 1 s to communicate with the network server and with the application server to know what to respond. Technically, the information from the sensor to an application server might get almost instantly, but any lost frame will delay the information by several seconds before the device can repeat its message.

LoRaWAN device can randomly select from multiple channels for the uplink message. Gateway needs to have hardware capable of receiving multiple channels at once. The price and power requirements of this hardware are not that important when the gateway can handle a large area. It can be accomplished by one or two demodulators with a multichannel baseband receiver [49] capable of receiving multiple frequencies and multiple spreading factors at the same time.

Class B device opens periodic receiving windows for quicker downlink messages. LoRaWAN gateways send periodic beacons once every 128 s. Class B devices synchronize with them and open receiving windows at precise times known by the gateway.

The gateway can be tightly synchronized with GPS with sub-µs precision, in which case it transmits all beacons. Beacons from multiple gateways merge in the device's antenna and the LoRa modulation is successfully received. If the gateway cannot guarantee the necessary precision, it is only loosely synchronized and will randomly send or drop beacons. It gives the device in range of multiple gateways a chance to receive only one beacon. The probability of transmitting a beacon is configured by a network server based on the distance to neighboring gateways.

Class C device is always receiving. This class has a large power consumption and therefore is used only by mains powered devices.

Class B and Class C devices also support multicast used to deliver the same data to multiple devices. It can be the only way to get a Firmware Update Over-the-Air (FUOTA) to all devices while complying with the duty cycle limit of a gateway.

3.3.2 Sigfox

Sigfox is another LPWAN technology using unlicensed sub-GHz frequency bands. With this technology, every device is registered and connected to the cloud by the same operator.

Sigfox uses Ultra Narrowband (UNB) signal with bandwidth of 100 Hz or 600 Hz depending on region. Uplink uses Differential Binary Phase Shift Keying (DBPSK) while downlink uses GFSK.

The device is not synchronized with the gateway. It starts by transmitting 3 copies of a message on randomly selected channels. The gateway needs a radio capable of receiving all subchannels at once. After 20 s from the first message, there may be an optional receive window. If there is a downlink message, the device responds and the conversation ends.

The extremely low datarate has several disadvantages. Packets can carry only 12 B uplink and 8 B downlink and still, the transceiver needs to be active for several seconds. That in turn increases sensor power consumption. In the situation of home automation and security, the average current consumption would be over 100 µA [19], which is an order of magnitude more than a CR2032 battery.

To comply with duty cycle limits, in some regions the device can use only 140 uplink messages a day, plus 4 messages used by the protocol itself. The actual limit of messages per day might be lower, depending on the service subscription. That would be a limiting factor even for a simple security sensor.

Downlink messages are limited to only 4 messages a day. Together with the delay between Tx and Rx it disqualifies Sigfox from any operation with a verified uplink.

3.3.3 ETSI TS 103 357, TS-UNB

Several companies have grouped into Mioty Alliance with intention to use Telegram-Splitting Ultra Narrowband (TS-UNB) part of ETSI TS 103 357 [50]. This technology uses GMSK modulation in the unlicensed sub-GHz frequency band together with Telegram Splitting Multiple Access (TSMA) for LPWAN star-shaped network. End-points can be either uplink-only Class Z or bidirectional Class A. Downlink messages come only after uplink with approximately 7 s delay.

The main principle is to split one packet into many bursts, each on a different channel. Coding of the message can take care of a missing burst. That can overcome interference that is limited in time or in frequency.

There can optionally be Sync-burst Data Unit before the packet. It contains information about the frequency-time burst pattern to be used by low-complexity receivers. This burst is sent on a dedicated channel C=24. The available documentation doesn't specify how much more complex the radio needs to be to be able to receive any pattern without the Sync-burst.

The documentation mentions cases where low latency is required, but the network timing for downlink response, to acknowledge uplink message, prohibits using this in anything like a smoke detector.

3.3.4 3GPP

3rd Generation Partnership Project (3GPP) is an organization responsible for most modern mobile telecommunication standards. In the area of LPWAN, they created standards for three independent technologies. As with other

3GPP standards, they use licensed frequencies and are deployed by large established mobile network operators.

Mobile phone chips with modems for various 3GPP standards could theoretically be used as a base for a wireless camera. The chips have a fast processor, graphics accelerator and OFDM-capable SDR modem which can be tuned to sub-GHz frequencies. All of them are necessary for a wireless camera in a home security system.

NB-IoT

Narrowband IoT (NB-IoT) is the lowest power oriented. It provides only up to 159 kbit/s and has a higher latency 1.5 s to 6 s [20]. Different from the other LPWAN, it verifies transmitted messages. Depending on a situation, the delay could be acceptable for security devices if the technology can guarantee delivery in that time. Light switches and other automation devices wouldn't work with this delay.

Current development even points to satellite networks [21] similarly to LoRaWAN.

EC-GSM-IoT

Extended Coverage GSM IoT (EC-GSM-IoT) is based on older Enhanced General Packet Radio Service (EGPRS) and can be deployed over existing 2G networks. It inherited its capabilities from 2G, datarate 474 kbit/s and latency of 700 ms to 2000 ms.

LTE-M

LTE for Machine-Type Communications Category M1 (LTE Cat-M1) is the most powerful out of the three. It can provide 1 Mbit/s with latency of 10 ms to 15 ms. Consumption of the devices is also closing the consumption limit of a coin cell battery. Nordic Semiconductor nRF9160 consumes 18 µA [51] while in Extended Discontinuous Reception (eDRX) mode with period 81.92 s. While in eDRX mode, the device automatically sleeps the modem to save power and wakes without needing to reconnect to the network.

4 Related Work

4.1 Custom FHSS Network

As a master thesis, I have tried to design an FHSS[1] network with the previously explained constraints in mind [22].

The main principle of this network is that CU sends a beacon frame every 100 ms on one of 4 channels. These 4 channels can be changed after a longer period. A sensor that wakes up is able to quickly scan beacon channels and from the first beacon frame quickly obtain information necessary to join the network. Space between beacons can be used for communication between CU and sensors.

This solution has lots of disadvantages similar to different frequency hopping networks. The main difference is that CU of this network creates a lot of radio traffic to compensate for the frequency hopping nature and the sleeping sensors. The European RF norms have since then slightly changed and now it most likely would not pass the certification.

4.2 Cortex-M Simulator

During my internship in STMicroelectronics, I was working with a brand new STM32WL MCU with GFSK and LoRa transceiver on-chip. I was allowed to publish an article about a simulator for Cortex-M microprocessors [23].

This tool does not directly fit into the dissertation topic, but might be very helpful for developers in this area. Testing microprocessor firmware can be challenging. Devices in home automation, security and more broadly any IoT tend to sleep for long periods of time. A full system integration test can in some cases take days and require a lot of resources.

The simulator developed can be used to do integration test of production firmware without any modifications needed for testing. The simulation can be run in a fraction of the time, save on the testing costs and thus increase the amount of testing and discovered problems.

[1] Note that the category FHSS is used loosely here, because the spectrum spreading is discutable. Frequency is changed from frame to frame and not inside one frame nor inside one symbol. Especially for a sensor with only one or two frames sent, the spectrum would not be spread at all.

5 Affordable All-channel Receiver

The first idea of improving the communication in a security system is similar to LoRaWAN LR-FHSS setup. A specialized receiver could be added to the system CU, where there is enough room for a slight price increase. This solution doesn't consider the high bandwidth cameras which are requested for some security systems. Parts of this chapter were published in [24], [25].

The main problem of the frequency agile or hopping network are sleeping sensors which need to quickly connect to the hopping communication. The sensor is sleeping long enough to lose synchronization. It doesn't have information about time and channel mapping and cannot easily transmit to CU. Basic frequency agility could be solved by scanning multiple channels or by having more physical receivers.

Scanning only works for a very limited number of channels, requires a longer preamble and highly depends on the transceiver used. For example, CC1200 transceiver needs 630 µs to switch frequency and check if there is a GFSK signal at 38.4 kBd/s [22]. Regular 4 B preamble is 833 µs long, so scanning more than just one channel presents a probability that the packet will not be detected. Scanning for a few channels can be done with a longer preamble, but it increases usage of the RF channel and batteries.

Having multiple radios in CU is another option with similar results. Several 3 $ transceiver chips can be hidden in a price of the CU. Regulations for sub-GHz FHSS require to have 47 or 58 channels respectively [52] or 50 channels [45]. That is unrealistic both as a price increase and as a space on the CU's PCB.

A receiver listening on all channels would eliminate the problem with synchronization to the hopping network. The sensor device could sleep as in the single-frequency solution. When communication is needed, the sensor could choose randomly one of its preferred channels and start transmitting immediately. CU would receive the message, know the important information without any delay, and respond on the same frequency with its regular transceiver. The response might be a simple acknowledge or synchronization data for the hopping network. The idea is depicted in Figure 5.1.

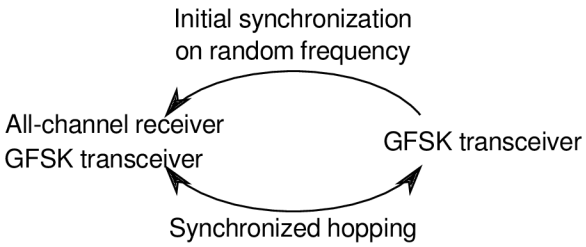

Figure 5.1: Sensor and CU with an Additional All-channel Receiver

The task at hand is to create a receiver for CU that works on many channels[1] concurrently. It might not be profitable to design a new radio integrated circuit for the amounts produced in this area of electronics. The remaining solution is to use existing mass-produced electronics for SDR and put necessary development into the software.

If the transceiver chip in the sensor is sufficiently advanced, this solution could be improved to a full FHSS the same way as in LoRaWAN. The sensor could send a small header with a hopping pattern, the CU's receiver would synchronize and the rest of the frame could be spread over many channels. In the scope of this work, only the task of receiving a single frame is investigated. Whether the frame contains sensor data or FHSS synchronization pattern is not important.

In the same way, this work doesn't handle higher Open Systems Interconnection (OSI) layers, for example L4 mechanisms for correct packet retransmission. These mechanisms are well known and don't need to be discussed here.

For the initial development, a simulation and RTL-SDR receiver with Matlab were used. RTL-SDR is an affordable and easy-to-use [26] SDR receiver based on DVB-T tuner. The radio concept was then ported to a Cortex-M MCU.

5.1 Design

The lowest level modulation, binary GFSK with $BT = 0.5$, is given by the available hardware for sensors. It is not feasible to put more expensive radio into devices that cost only a few $ to manufacture. Frequency deviation, symbol rate and channel spacing could be set to the needs of the receiver.

[1] Separate carrier frequencies might be called *subchannels* and independent sets of many subchannels can be called *channels*. Whether the communication can be used on multiple independent sets of carrier frequencies is an implementation detail. In this text, *channel* is used for carrier frequency and not sets of them.

5.1.1 Receiving Symbols

Simplified binary FSK signal might look as

$$s = A \cdot sin\left(2\pi t \cdot (f_0 \pm f_{dev})\right) \tag{5.1}$$

where f_0 is carrier frequency, and f_{dev} is frequency deviation. The $\pm$ sign can switch each T_s symbol duration.

Signal s is received, demodulated to a lower frequency, sampled and digitized by the SDR. There are multiple types of SDR constructions. It can be done even without the demodulation step [53] which shows that development of this area still continues.

The digital signal first enters an FFT to receive all channels at once. The FFT has bins set to frequency f_0 of each channel. Signal s won't fit precisely into any bin. The best it can do is

$$s = A \cdot sin(2\pi t f_0 \pm 2\pi t f_{dev}) \tag{5.2}$$

where $2\pi f_0$ is the bin frequency and A with

$$\pm 2\pi t f_{dev} = \varphi(t) \tag{5.3}$$

are amplitude and phase, products of the FFT bin.

The first derivative of the phase is

$$\frac{d\varphi(t)}{dt} = \pm 2\pi f_{dev} \tag{5.4}$$

and that is exactly the bit of information we are interested in. We look at sign of $\frac{d\varphi(t)}{dt}$ and decide on the received symbol.

In practice, it has to be replaced by discrete approximation, in the simplest by a difference of two consecutive samples. Each sample $\varphi(t)$ must be in range of $\langle 0, 2\pi)$. If it goes above or below, it wraps around. The same has to be done with the difference. That limits phase difference which can be detected to

$$\varphi\left[\frac{t}{T_c}\right] - \varphi\left[\frac{t - T_c}{T_c}\right] \in \langle -\pi, \pi) \tag{5.5}$$

where T_c is the duration of the samples at the output of FFT. The maximum detectable frequency deviation is

$$2\pi T_c \cdot max(f_{dev}) = \pi \tag{5.6}$$

$$max(f_{dev}) = \frac{1}{2T_c} \tag{5.7}$$

Normally, the sampling time could be synchronized with the received signal to minimize the intersymbol interference [27] [28]. Here, several individual devices transmit their symbols randomly shifted in time. It wouldn't help to

synchronize sampling to only one of them, and interpolating the signal for each channel separately would be too complicated. For a 30 ppm quartz crystal, $\frac{T_c}{2}$ shift happens after 8333 symbols or 1042 bytes. That is several times more than the longest packet of most similar systems, so the symbol synchronization was simplified to the bare minimum. I have selected $T_s = 2T_c$ to sum two neighboring samples.

The condition for maximal received frequency deviation can now be expressed as

$$max(f_{dev}) = \frac{1}{2T_c} = \frac{1}{T_s} \tag{5.8}$$

which can be compared to a Minimal Shift Keying (MSK) modulation ($f_{dev} = \frac{1}{4T_s}$) and gives three-quarter margin for noise and oscillator imperfections.

MSK is a special case of FSK where $f_{dev} = \frac{1}{4T_s}$. It simplifies the phase shift caused by one symbol to $\varphi(T_s) = \pm\frac{\pi}{2}$. The difference between both symbols is exactly half of the sine wave. It is the variant with minimal bandwidth while keeping both symbols orthogonal to each other.

Another special case for both FSK and MSK are GFSK and GMSK. A Gaussian filter is used for the f_{dev} change which smooths the signal and reduces bandwidth. The disadvantage is intersymbol interference. Both methods of reducing signal bandwidth are helpful when putting many channels next to each other.

The channel size or size of the FFT frequency bin is equal to sampling frequency at the output, $f_{ch} = \frac{1}{T_c}$. The size must be greater than the bandwidth used by the communication to fit the signal inside the provided channel. It is hard to estimate the bandwidth of GFSK, but Carson's rule can give a rough estimate of

$$f_{bw} = 2 \cdot \left(\frac{BT}{T_s} + f_{dev} \right) \tag{5.9}$$

where BT is the Gaussian filter bandwidth bit period product. For GMSK with $BT = 0.5$ it would simplify to $f_{bw} = \frac{1.5}{T_s}$. That gives another condition on the sampling rate

$$f_{ch} = \frac{1}{T_c} \geq f_{bw} = \frac{1.5}{T_s} \tag{5.10}$$

$$T_s \geq 1.5T_c \tag{5.11}$$

which is satisfied by previously selected $T_s = 2T_c$ and a small reserve remains for the bandwidth estimate.

Using higher f_{dev} than GMSK would increase the signal bandwidth, and force us to increase T_s to T_c ratio and decrease the amount of useful data. Since GMSK has been successfully used for decades, there is no need to increase f_{dev}.

An RTL-SDR dongle has limited IQ sample rate of 2.4 Msample/s which limits the received bandwidth. The size of the FFT has to be $n = 64$ as it is the lowest possible power of 2 which satisfies regulations. Some of these channels can be permanently silent to avoid neighboring communication or

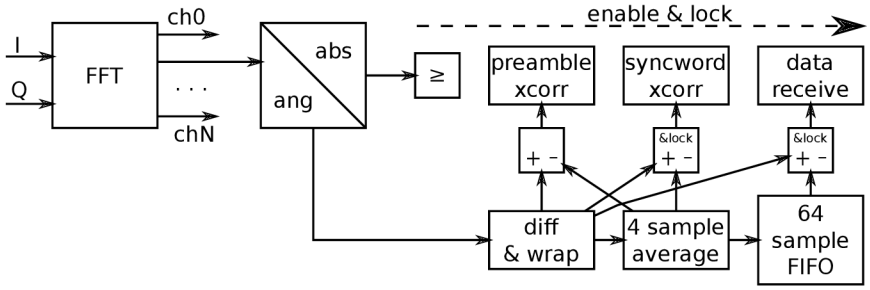

Figure 5.2: Basic Scheme

reserved channels. Channels are $37.5\,\text{kHz}$ apart, the symbol rate has to be $1/T_s = 18.75\,\text{kBd/s}$ and the frequency deviation $f_{dev} = 4.6875\,\text{kHz}$. Channels were put from $865\,\text{MHz}$ up, so no Low Duty Cycle / High Reliability (LDC/HR) channel is overlapped. The LDC/HR is a reserved group of channels that cannot be used by an FHSS.

5.1.2 Processing Packets

Each packet starts with a preamble and a syncword. The preamble is an alternating sequence of zeros and ones used for synchronization. The syncword is a pseudo-random value, selected beforehand, to know when preamble ends, useful data start and to differentiate between systems with the same modulation. Running cross-correlation was chosen to detect preamble and syncword. The match on the syncword also sets the symbol synchronization. After the syncword, it is common to include a length of the useful data in the packet. At the packet's end, there is usually a CRC.

GMSK transceivers usually have some frequency offset. It can be calibrated in a factory, but time and temperature will cause some additional uncalibrated offset. It needs to be compensated by the receiver. An averaging filter was used to get transmitter carrier frequency from the alternating preamble. The filter has a length of 4 samples. Length in multiples of $4T_c = 2T_s$ will average the same amount of negative and positive f_{dev} and get the offset. The offset is stored in a circular buffer of 64 values. At the time of syncword match, the last value is used for symbol decision. The buffer delays the valid offset value, because at the time of syncword match, the filter has already processed the syncword. The syncword is not balanced in zeros and ones and would bring a frequency offset of its own.

Now the simplified scheme of the receiver can be seen at Figure 5.2. Individual blocks can be enabled only at the time they are needed to save on computational power. Only the FFT and power detector need to be always on. When a detector is triggered on a particular channel, more blocks are enabled, but only blocks for the given channel. The first step is a power rising over a given threshold which enables calculation of phase angle, its difference and preamble correlator. The next stage freezes frequency offset and activates sync-

word cross-correlation. Last stage updates frequency offset from the 64-sample FIFO and receives data. Everything must be protected by a hysteresis or fail-safe timeout. Absolute signal power is expected to keep high during the whole packet, so there can be a hysteresis and a condition resetting everything on packet failure. Cross correlations on the other hand are expected to fall shortly after detection, well before the packet ends, so there needs to be a protective timeout resetting the receiver only if something fails.

5.2 Simulation

The receiver implemented in Matlab Simulink was tested by a simulation. The RTL-SDR module returns preprocessed baseband complex signal in a single floating point type. It is equivalent to synthetic $e^{2\pi ift}$ or to the output of the `comm.GMSKModulator()` function. That simplifies the test and doesn't require almost any changes in the receiver model.

All simulations were done with the signal passed through an Additive White Gaussian Noise (AWGN) channel which is the simplest method. Individual channels of this receiver are relatively thin and the receiver doesn't presume any multipath effects nor frequency distortions. Therefore, this kind of test should be enough for this receiver. Frequency-specific effects would be interesting in comparison to a single-frequency system, but that is not specific for this receiver and it has already been studied [7].

The input to all simulations were packets with 20 useful data bytes. That is 31 B in total, including all overhead (4 B preamble, 4 B syncword, 1 B length and 2 B CRC). Perhaps it is more common to simulate Bit Error Rate (BER), but here the success depends on syncword match and frequency offset elimination. Packet Error Rate (PER) was used as a simulation output. No error correcting code was used and a single erroneous bit means the packet is not received successfully. The received data were compared with the data sent because the CRC used (CRC-16-IBM) offers only a basic protection.

The packets were generated by Matlab's `comm.GMSKModulator()` already at the base sample rate with 64×2 samples per symbol. Packets were multiplied by a step window passed through a Gaussian filter to smooth the power rise and fall. A similar process, power ramping, is done by real transmitters. Additionally, a random time in a range between 1 sample and 2 symbols was prepended before the packet and a complement of that time was appended after the packet to have a constant length. This was to test the syncword match and the symbol detection. In the end, the packet was shifted to the frequency of one of the channels by multiplying with $e^{2\pi ift}$, where f was channel offset from the receiver center frequency in a range from -32 to $+31$ divided by 64, size of the FFT. Variables f and t also contained 2.4 Msample/s sample rate to fit the same receiver to both hardware and simulation.

The simulation was set up in a complicated manner to shrink the simulation time down to a practical level. As it turns out, Matlab and Simulink have quite

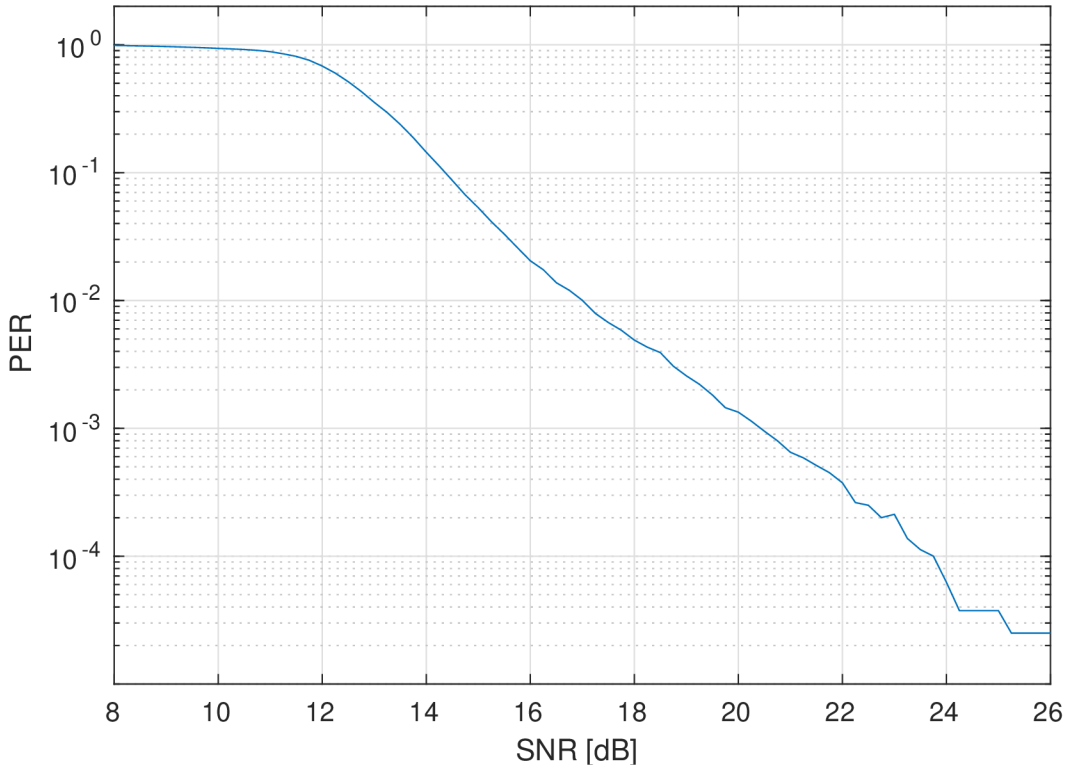

Figure 5.3: Packet Error Rate

bad memory management and parallel computing capabilities. It is not a problem to create several thousand packets and start `parsim()` simulation, yet starting and stopping each simulation takes orders of magnitude more time than the actual simulation. The problem even increases after a few hundred simulations. I found the optimal setup to be 50 packets in one signal, separated by a few bytes of zeros (approximately 2 million samples). Approximately 100 of these signals were simulated together in one `parsim()`. With these numbers, the signals were not too long for the initial model build, the initial build was done 6 out of 100 times and the latter simulations were not yet slowing down. This was repeated 20 or more times to get enough data.

5.2.1 Receiving Packets Through AWGN Channel

Figure 5.3 depicts simulation of different Signal to Noise Ratio (SNR) and the resulting PER. The SNR was corrected for the fact that the FFT filters out 63/64 of the noise. The SNR level of the AWGN channel was decreased by approximately 18 dB.

$$SNR_{offset} = 10 \cdot log_{10}\left(\frac{1}{64}\right) \approx -18\,\text{dB} \tag{5.12}$$

The Figure 5.3 shows that the receiver starts receiving after 12 dB of SNR with 17 dB for 10^{-2} PER.

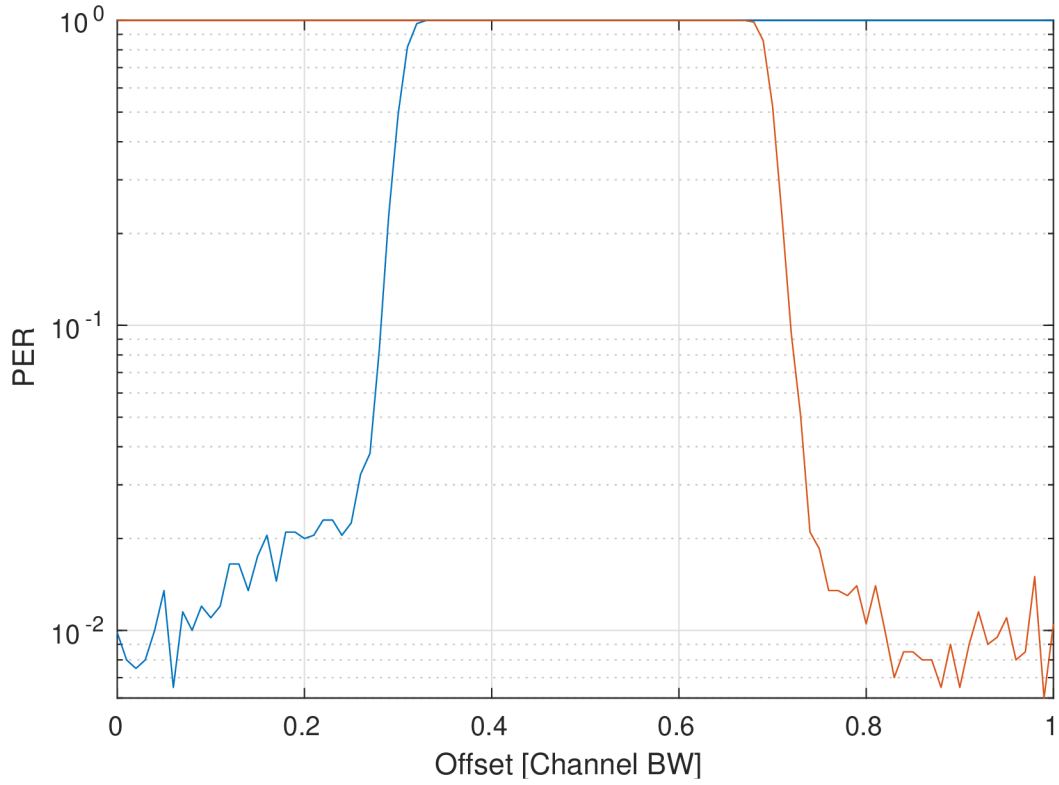

Figure 5.4: Packet Error Rate on Frequency Offset

5.2.2 Receiving with Frequency Offset

Figure 5.4 shows how the Rx is influenced by the frequency offset of the Tx. This simulation was done with 17 dB SNR, so the expected PER at exact channel frequency should be around 10^{-2}.

The left curve in Figure 5.4 rises slowly from the correct channel frequency up to $0.25 \cdot 37.5\,\text{kHz} = 9.375\,\text{kHz}$, where the error rate more than doubles. It shows that even a small frequency offset will influence the communication and production devices should be calibrated. Rx stops receiving completely when Tx shifts for more than approximately $0.3 \cdot 37.5\,\text{kHz} = 11.25\,\text{kHz}$. This is perhaps less than a normal GMSK receiver would [40]. It is also less than the limit of the detectable deviation on FFT output

$$f_{dev\_max} - f_{dev\_gmsk} = \frac{1}{T_s} - \frac{1}{4T_s} = 14.0625\,\text{kHz} \tag{5.13}$$

but that would be only in ideal conditions without any noise.

The right curve is the same curve flipped, when the receiver starts receiving the packet on a wrong channel.

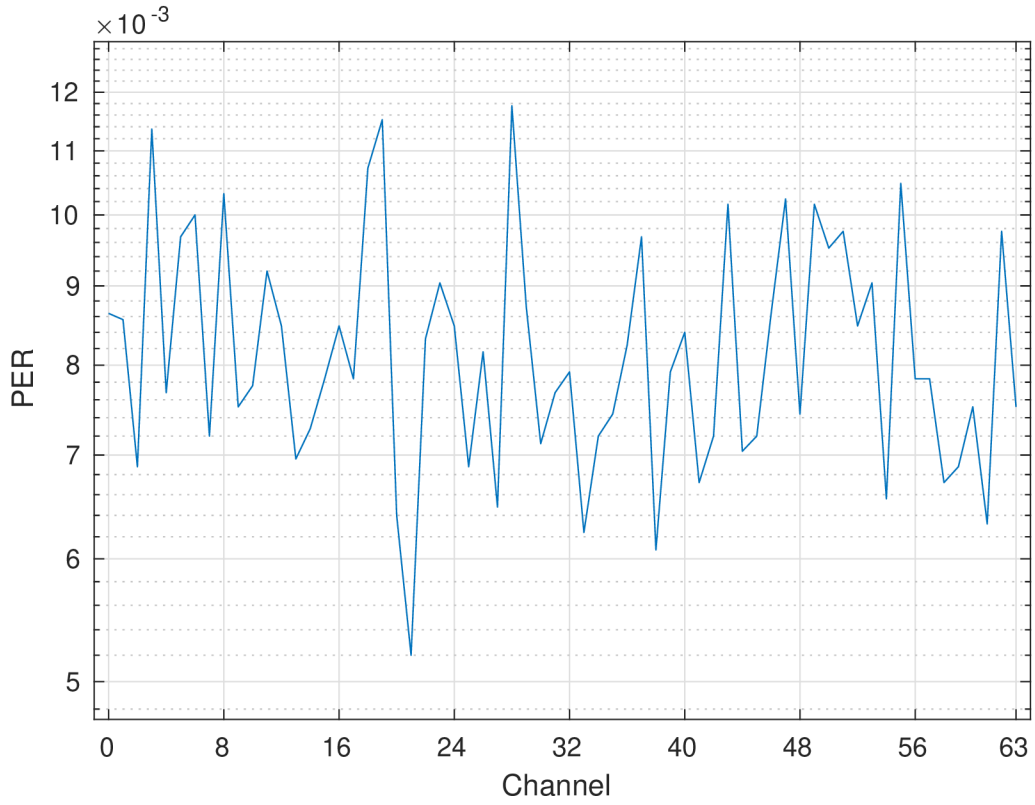

Figure 5.5: Packet Error Rate on Different Channels

5.2.3 Comparison of Different Channels

Figure 5.5 shows dependency of the receiver on channel used. This simulation was again done with 17 dB SNR and expected PER should be around 10^{-2}. Only one random sequence was generated for all channels. This is to ensure no influence of randomness on a specific channel. Channels were selected randomly, but incremented by the simulation index. There were 64 simulations which means that all channels were always simulated, but in the scope of one signal, the channel appeared random.

The result is only a noise below 10^{-2} which verifies that the receiver shouldn't be dependent on any particular channel used.

5.2.4 Influence of GMSK on Other Channels

Figure 5.6 shows the influence of a GMSK signal on a neighboring channel. The format of tested packets was kept the same as in the previous simulations. The AWGN channel was kept on the same level of 17 dB SNR between useful packets and white noise. An additional GMSK signal with the same parameters and randomly selected symbols was added on a neighboring channel. This noise signal was continuous without a header and pauses between packets. The power of the noise GMSK was varied. SNR in Figure 5.6 is a difference

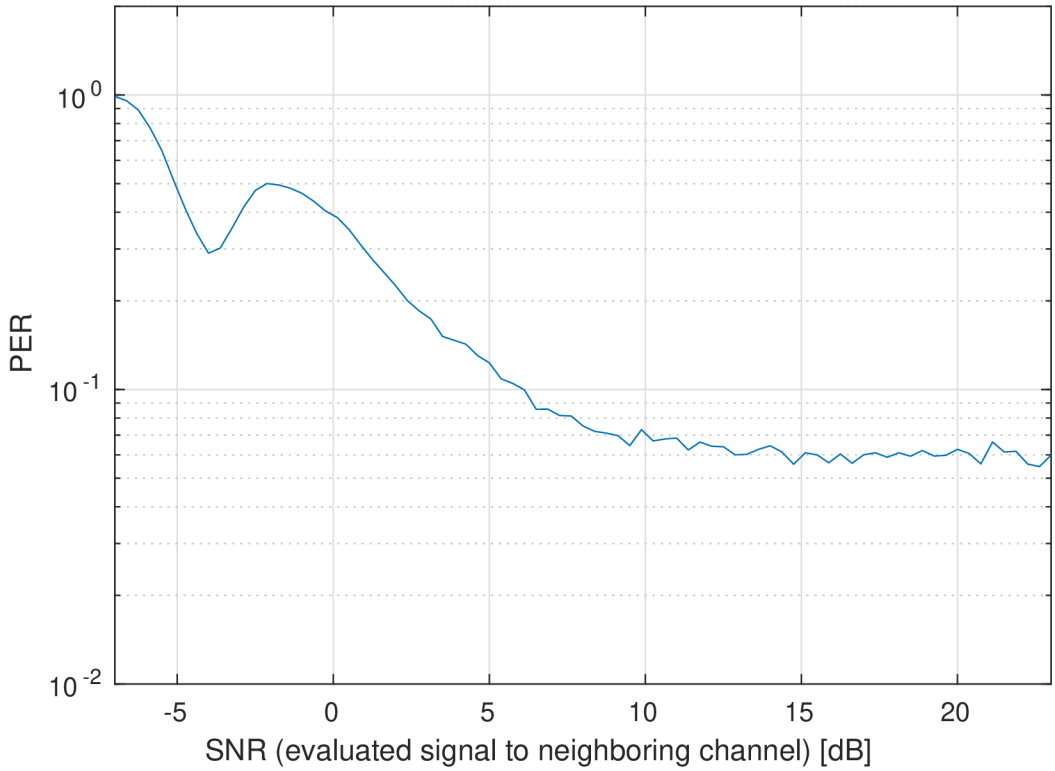

Figure 5.6: Packet Error Rate with GMSK on Neighboring Channel

between power of the useful packets to the noise GMSK on the neighboring channel.

The expected error rate without signal on a neighboring channel is 10^{-2}. Even with a much weaker neighboring signal, the PER doesn't fall to the expected value but stays approximately 6 times higher. With both signals having the same power, the receiver is more than an order of magnitude worse.

Figure 5.6 also shows a bump between -4 and -2 dB. A stronger noise influences the receiver less than a weaker signal. A similar bump shows even when varying simulation parameters, so it doesn't seem as an error of the simulation. The bump could be explained by two influencing factors together, but the exact cause of this is not known.

Influence on the neighboring channel is significant, but less important if the communication will randomly select a different channel. The probability of a collision of two devices selecting the same channel is $1/64$ which should be prevented by CCA. The probability of selecting a neighboring channel is $2/64 = 0.03125$ which almost compensates for the worse PER.

Figure 5.7 shows the same two signals with larger channel distance between them. Both signals have the same power and AWGN was still set to 17 dB from the useful packets, not counting the power of the GMSK noise. The influence of the other GMSK signal falls quickly with the first few channels of distance. With another GMSK further apart, the receiver PER stays approximately 6 times higher than without another GMSK. It seems that both a weak

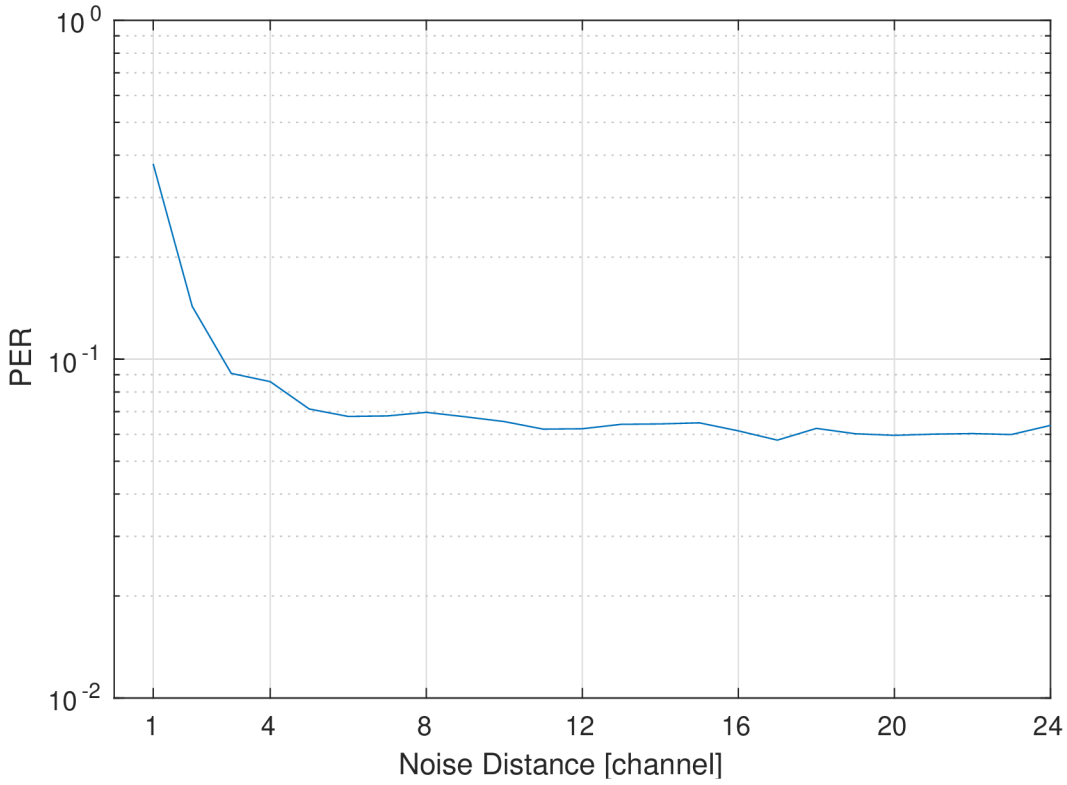

Figure 5.7: Packet Error Rate with GMSK on Other Channel

signal on the neighboring channel and a strong signal far apart have the same influence.

5.2.5 Multiple Packets Received at Once

Figure 5.8 shows how the receiver works when multiple packets are running simultaneously. Instead of one packet in previous simulations, N individual packets were generated separately and then summed together. All packets were created with the same power. SNR value is the distance between the noise and the power of one GMSK signal.

In this simulation, the random shift of the packets was increased to between 1 sample and 16 B. This should be closer to reality where the chance of packets starting at the exact same time is negligible.

Concurrent packets were randomly spread between available channels by using permutation and picking the first N elements. This was to prevent collisions, so there were no two packets at the same frequency at the same time. The random selection of a channel adds randomness in how close the channels are but is similar to a real device trying to transmit.

Data at Figure 5.8 shows expected increase in PER with increasing number of concurrent packets. For 64 packets at once the PER doesn't fall as quickly with higher SNR, similar to the limit caused by a signal on a neighboring chan-

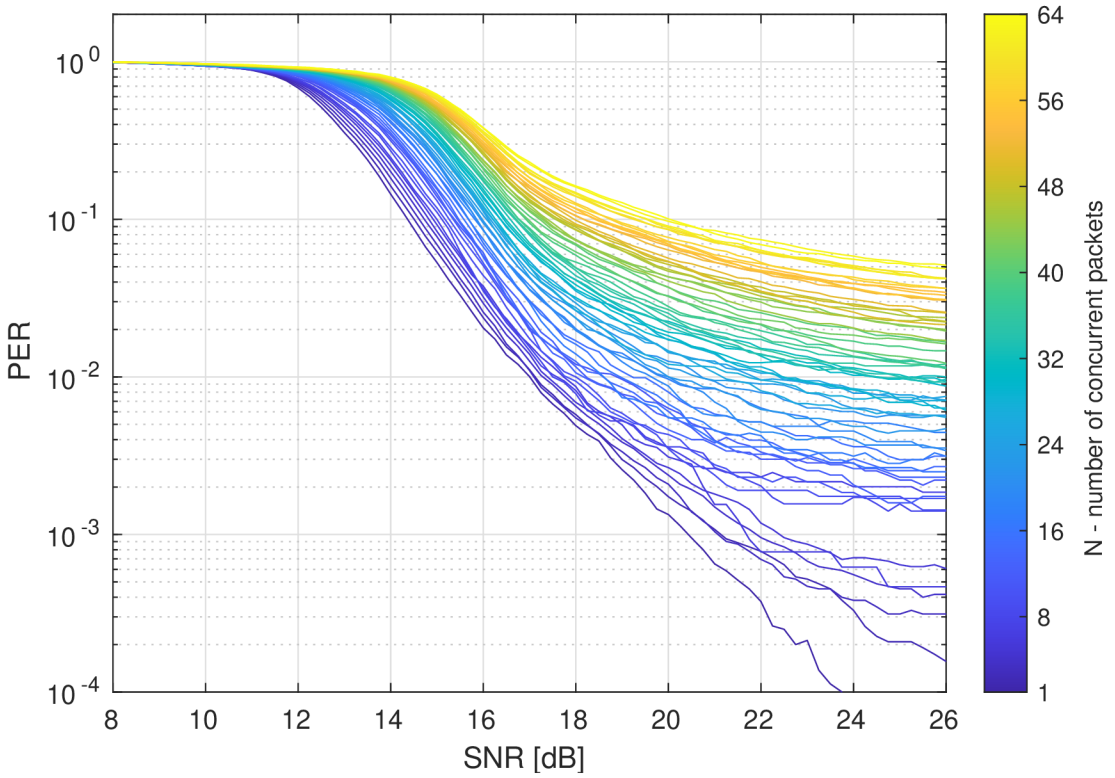

Figure 5.8: Packet Error Rate on Concurrent Packets

nel. Even in this case of entirely full radio spectrum, the receiver would have 90 % chance of success at 20 dB SNR. A bigger problem would be planning the L4 protocol layer to be able to respond to all 64 devices in a reasonable time. The other problem would be computational power. Matlab simulation doesn't need to run in real time if it doesn't have any real-time inputs and 64 packets at once might be too much for a common computer.

For the real scenario, where only a few packets are expected at the same time, the change in performance is relatively small.

5.3 Test with RTL-SDR

This time the Matlab Simulink receiver was connected to RTL-SDR hardware with a long stick antenna. This device is constructed on a base of DVB-T USB dongle. The regular DVB-T receiver is switched to raw mode and sends IQ samples through USB to host PC. It is a cheap alternative to professional tools, essential for students to explore SDR.

The receiver was running in Matlab Simulink in an infinite real-time simulation. Transmitter was a repurposed electronics board with STM32L0 MCU, CC1200 RF transceiver and an onboard PCB antenna. It sent packets with 20 B of useful data. The first four bytes were the hardware address of the transmitter. Following was one byte with a channel on which the packet was sent. The

next two bytes were 16 bit counter incremented with each packet. The counter was used to evaluate missing packets. For each received packet, the number of received packets was incremented by one and the number of missed packets was incremented by counter difference minus one. The rest of the packet were fixed values to validate the received data.

The transmitter started by 2 ms receiving, followed by a transmission of the packet. The receiving part was implemented as CCA to prevent collisions from influencing the results. This was repeated on different channels. The transmitter was incrementing frequency channels by one on each packet. This simplified processing of the received data, as all missed packets were easily pinned to a given channel. Random channel selection would require knowing the selection at the receiver to know what channel was used for the missing packets.

Several tests were made inside an old university building. In each test, over $20 \cdot 10^3$ packets were sent. For the transmitter on the same table approximately 1 m apart, PER was $1 \cdot 10^{-3}$. Interestingly, a few packets were received as copies on a neighboring channel. Gain of the RTL-SDR was set to maximum and Automatic Gain Control (AGC) was turned off, so a weaker packet would not be influenced by a stronger packet on another channel. Maximal gain creates a significant distortion as the signal is clipped. It shows how robust can frequency modulation be.

When the transmitter was one floor up and approximately 30 m far, the PER was even better $2 \cdot 10^{-4}$.

The test at Figure 5.9 was done with transmitter two floors up, about 50 m far and behind complicated wall structure. Basically as far as the school building allows. PER in this situation was $5 \cdot 10^{-2}$, which is still usable 95 % of packets successfully received. At this distance, the transmission on a single channel wasn't received by the same hardware used for Tx, though the comparison is a bit misleading as the hardware receiver had a different antenna.

The blue line in Figure 5.9 shows that the receiver is bad on several low channels. It may be caused by filters in the demodulator [54] or filters in the DVB-T receiver which are not well documented. The resulting PER without first 3 channels would be $3 \cdot 10^{-2}$.

The overall performance of the receiver would require much more elaborate measurements and a laboratory environment. Still, it is seen that the receiver works at a reasonable distance, considering the thick walls and floors of an old building. When compared with a single-channel chip receiver, this SDR receiver is at least comparable if not better.

5.4 Hardware Platform

The overall receiver concept is tested and the next step is to prove that the receiver can run in affordable electronics. Running a full computer with Matlab Simulink is not practical for an embedded electronics CU. The hardware used is based on LPC-Link 2 with a custom RF demodulator expansion board.

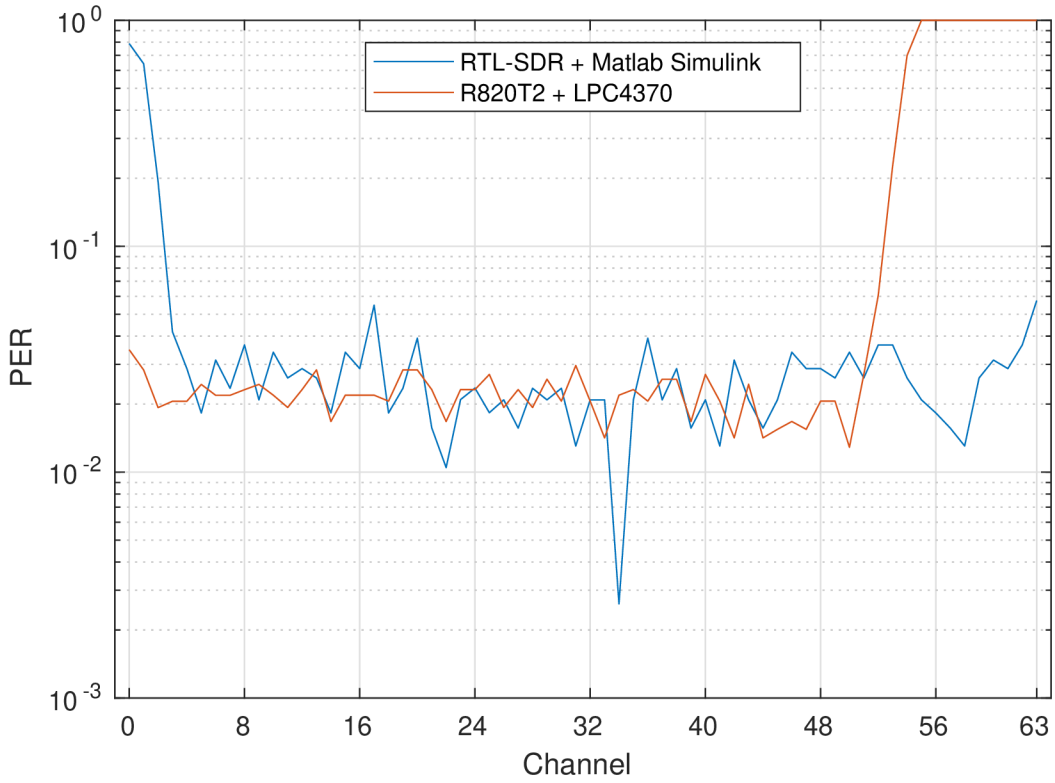

Figure 5.9: Channel Statistics

5.4.1 LPC-Link 2

The LPC-Link 2 is a fast debugger for Cortex-M LPC MCUs made by NXP. It itself is also based on a powerful Cortex-M MCU and can be used as a development kit debugged by another LPC-Link 2.

The kit provides the LPC4370 MCU, external flash program memory, expansion connectors and a handful of other necessary components. This multicore MCU has one Cortex-M4 core, two Cortex-M0 cores, all running at 204 MHz, and for an MCU extremely fast 80 MHz ADC [55]. Nowadays, there are more powerful MCUs with the same size and power requirements, but none has comparable ADC.

5.4.2 Demodulator Expansion

An expansion board with a demodulator was designed to receive 868 MHz RF signal and demodulate it for the ADC. The demodulator chip is Rafael Micro R820T2, the same as in RTL-SDR. This chip was selected because part of its documentation leaked to the public and is available.[2]

[2]Unfortunately, the original R820T2 is at end of its life [56] and its successors are kept secret with no documentation available. According to authors of RTL-SDR, its replacement R860 is identical [57] and so the documentation for R820T2 is still usable.

The demodulator has an onboard PCB antenna connected to its RF input. This antenna is common in sub-GHz home electronics. Better antennae exist, but this design is sufficient for the purposes of this experimental implementation.

The output of the demodulator is a differential pair with frequencies of several MHz. In the path of the signal are capacitors to remove any DC component from both signals. The resulting signal is put through the LPC-Link 2 expansion connector to the differential input of the ADC. There is also a layout for a voltage divider and a pair of anti-parallel diodes clamping the differential signal to protect the MCU's ADC input which can survive only 400 mV. Later measurements showed that they were not needed and the output stays under the limit.

A mistake was made in the board design due to insufficient documentation. The LPC4370's differential ADC has internal bias resistors. They are used on DC separated signal to provide safe DC value. In that case, an additional 100 kΩ pull-down resistors need to be connected to the ADC inputs. This was not mentioned in the datasheet [55] nor in the user manual [58] of LPC4370. Without these resistors, the measured signal has a significant offset which complicates further calculations. The resistors were added later to the prototype boards to fix the problem.

The demodulator is set and controlled through an I2C bus. In contrast to standard I2C connection, there are resistor-capacitor filters on the signal wires. These were copied from the reference application schematic found in the leaked datasheet for the demodulator. The purpose is probably to balance I2C signal quality and interference from quick falling edges.

There are extra pads for an I2C DAC which leads to analog gain control of the demodulator. Layout for it was etched into the PCB as a precaution but the components were not placed. Gain controls inside the demodulator were enough for the function of the prototype.

The whole PCB was designed with 4 copper layers to provide a ground plane close to the demodulator chip. The expansion board, including the antenna, is 4×4 cm large. Resulting hardware platform is depicted in Figure 5.10, where the blue longer board is LPC-Link 2 and square green board is the expansion demodulator.

5.4.3 Current Consumption

The LPC-Link 2 can be powered by 5 V from mini USB connector or by 5 V or 3.3 V on solder pins. There are linear voltage regulators between 5 V and 3.3 V which makes powering the board by 5 V quite ineffective.

When the receiver is running powered by 3.3 V, the whole board consumes 394 mA. Out of this value, only 203 mA are consumed by the MCU and 191 mA are consumed by the demodulator. The consumption of the demodulator is surprisingly high for an analog chip only 1/4 size of the MCU, but the values correspond to values found in respective datasheets.

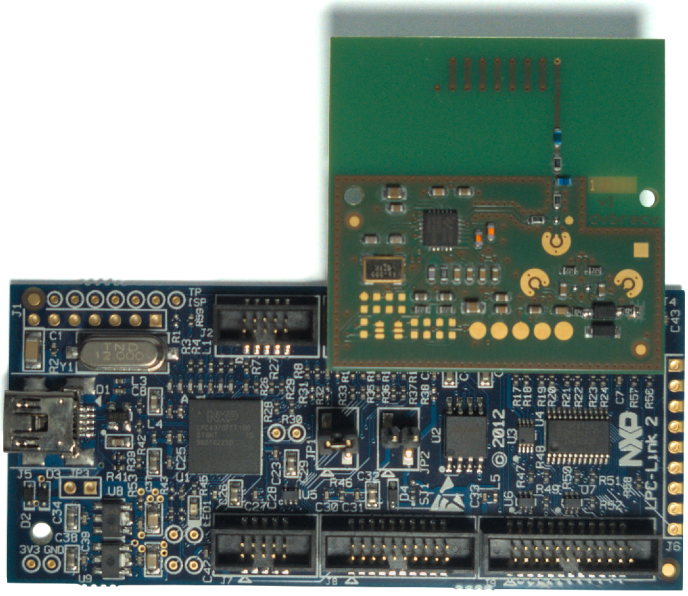

Figure 5.10: Hardware Platform

The total consumption is approximately 1.3 W. It is an order of magnitude more than the consumption of radio for a single-frequency network, but power requirements of the CU are not as big an issue and this value is acceptable.

5.5 Firmware

All firmware is written in C language. The MCU manufacturer provides an Eclipse-based IDE MCUXpresso which comes with project examples. These examples are important for the control of the MCU's cores and the external flash memory. On power up, the MCU's Cortex-M0 cores are stopped and the Cortex-M4 core loads its program code from the external memory. Provided linker scripts add M0 firmware as data for the M4 which loads them into proper RAM sections in its startup code.

5.5.1 Demodulator, ADC and DMA Control

The demodulator, ADC and Direct Memory Access (DMA) unit are controlled by the M0 core. The DMA is set up to transfer ADC samples, full FFT frame at a time, directly to RAM. The ADC produces 12 bit samples already in two's complement format. When one frame is transferred, it is automatically switched to a second buffer and an interrupt is triggered. This way the mea-

surement is fully automatic and the M4 core can immediately start calculations on the frame.

The ADC takes differential signal but only one stream of real data. RT820T2 normally converts the DVB-T signal to Intermediate Frequency (IF) of about 4 MHz with bandwidth between 6 and 8 MHz. That is the main difference in SDR architecture from RTL-SDR. In this setup, the ADC speed must compensate for the missing complex component and the signal must be later processed digitally. The simplest method is to sample twice as fast as in the Simulink receiver, 4.8 MHz, which results in 128-sample blocks per one T_c. The lowpass filter of the demodulator can be set around this low value [54]. The ADC can take samples much faster, but the MCU would not be able to process the signal.

Control of the demodulator was quite a challenging task since the available documentation is very brief. There is an open-source driver for R820T2 available in the Linux repository, but it is flawed. It isn't able to set the demodulator's Phase Locked Loop (PLL) frequency to the 868 MHz band with the quartz crystal that is suggested by the documentation. This is probably just another bad effect of no public documentation. In most cases, it is not a problem, because different crystal values may be used in different receivers and DVB-T signal should not be at 868 MHz. Luckily, the Airspy open source project solved the PLL setting differently and it was possible to take inspiration there.

5.5.2 MCU Speed and CMSIS-DSP

The speed of the MCU is barely enough for the task to receive on all channels at once. Several changes had to be made for the necessary calculations to fit inside the given time interval.

Most of the calculations are done with Cortex Microcontroller Software Interface Standard - Digital Signal Processing (CMSIS-DSP) which is a mathematical library for Cortex-M processors. This library is highly optimized although there are examples where user code can be better. Table 5.1 shows time needed for various operations on the M4 core at full speed 204 MHz. Calculations are done for type `q15_t` provided by CMSIS-DSP.

In some cases, there are simple tricks to speed up firmware operations. The ADC output is 12 bit and needs to be multiplied by 16 to get the full resolution of a 16 bit `q15_t`. The first tip, used by almost all programmers is to use bitshift instead of multiplication. Another tip is to do two shifts in one MCU clock by using 32 bit Arithmetic Logic Unit (ALU). The 12 bit values are stored in 2 B next to each other in memory and cannot overflow if they are both shifted at the same time. This can in some cases be used also for addition and other operations if the result cannot overflow the given space. The last tip is called loop unrolling. Most `for` loops can be rearranged to do more calculations in each cycle to be a bit faster.

Table 5.1: Duration of Various Operations

Operation	CMSIS-DSP	MCU Cycles	Time [µs]
128×copy	•	370	1.8
128×copy		201	1.0
128×offset	•	537	2.6
64-sample complex FFT	•	2745	13.5
128-sample real FFT	•	4761	23.3
—||— without bitreversal	•	4377	21.5
64×magnitude^2	•	539	2.6
128×bitshift & overflow		273	1.3
128×bitshift		569	2.8
128×copy & sin		699	3.4
2×decimate 1×tap	•	2937	14.4
2×decimate 3×tap	•	3586	17.6

5.5.3 FFT and Magnitude Squared

The goal is to keep the communication speed of the receiver designed in previous sections. This means that one 64-sample FFT and one sample of the resulting signal need to be calculated in

$$T_c = \frac{T_s}{2} = \frac{1}{2 \times 18.75\,\mathrm{kBd/s}} \approx 26.6\,\mu\mathrm{s} \tag{5.14}$$

where T_c is the time of a sample of the resulting signal and T_s is the time of one GMSK symbol.

The calculations were split between the M4 core and one M0 core. The M4 core has additional vector instructions and a full ALU, so it is used to calculate the FFT and magnitudes for all channels. Processing of the individual GMSK signals, remainders of the receivers, evaluation of the packet data, control of MCU peripherals and demodulator is handled by the M0. The GMSK processing is done only for a limited number of active channels, so it may seem like more tasks, but it is less work for the smaller core.

Given the time measured in Table 5.1 and the time budget of (5.14), the M4 core has less than 26.6 µs to:

- 1.3 µs, bitshift the signal from 12 bit two's complement to 16 bit two's complement `q15_t`.
- 3.4 µs, copy and interleave from real to complex data type leaving the imaginary element zero and multiply with $f_{sample}/4$ sine and cosine.
- 2×17.6 µs, filter the signal to get rid of the unwanted image.
- 13.5 µs, do a 64-sample complex FFT.

The total would be approximately 53 µs. It is clear that the M4 core cannot further downconvert the signal to zero IF.

The solution is to sacrifice some channels near the zero frequency and use real FFT of twice the size. The optimizations used by CMSIS-DSP allow to calculate only 64-sample complex FFT and then split the result to obtain 128-sample real FFT. Together with the calculation of magnitude squared it would almost fit within the time limit. To get into the time limit and provide a reasonable reserve, the magnitude calculation is done only for a half of the channels alternating with each sample. The magnitude is used only to start following parts of the GMSK receiver and it doesn't need to be known for each sample.

5.5.4 Arctangent and GMSK Decoder

Decided on the value of the magnitude squared, a limited number of channels is selected and marked active. Currently, only 6 packets can be received at once. Only active channels have arctangent calculated. It would not be feasible for the M0 core to calculate arctangent for all 64 channels. It could be possible to use the other M0 core of LPC4370 to calculate an additional 6 channels, but it was not done in this work.

There are many numerical approximations to calculate arctangent [29]. But the best seems to be the Coordinate Rotation Digital Computer (CORDIC) algorithm [27]. Implementation of CORDIC used in this work was made by ST [59]. Speed of approximation such as $a \cdot x - b \cdot x^2$ and CORDIC is almost the same, but the CORDIC algorithm seems to produce much better results for the receiver.

The active channels are passed through four stages. Compared with the receiver implemented in Simulink, there are many smaller changes to speed up the calculations. Instead of preamble and syncword correlations, there are simple comparators with decoded symbols. Instead of long FIFO for calibration of the frequency offset, this implementation uses deviation at the time of preamble match.

In the first stage, the receiver looks for a preamble. Four sample average is subtracted to remove offset and compared with 0 to get GMSK symbols. Even or odd samples are compared with alternating sequence to match the preamble. When the preamble is detected, the stage advances to syncword compare, where all 4 B must fit the signal. The syncword fit also sets the symbol position on the correct one of two samples. The next stage is a receival of individual bits and bytes of data. For simplification, only odd or even samples are used for the GMSK symbol decision.

In the last stage, the packet waits in a buffer until CRC is checked. CRC check is done from main loop with a lower priority, together with printing information to UART console.

5.6 Results

The receiver was tested with a single board CC1200 transceiver. Ordinary GMSK packets were sent on all frequency channels, received by the software receiver and printed through UART to PC. The packet format was the same as in the previous test. There were preamble, syncword, length, 20 B of useful data filled with counter, channel and known pattern and 2 B CRC at the end.

The receiver and transmitter were in the same room, about 1 m apart and about 30 cm from any obstacle. An unusually small distance was selected because the receiver can successfully work only in a range of one room.

Dependence of Rx success rate on channel is depicted back at Figure 5.9 on page 45. The drop around channel 51 (red curve in Figure 5.9) is for the receiver in LPC4370. This drop was expected. The downconversion process in the demodulator flips the frequency spectrum and its high pass image filter cuts approximately 500 kHz [54] which approximates to 13 channels. It is the downside of the FFT simplification mentioned earlier. This drop is a nice change because the test with RTL-SDR showed a different drop (blue curve in Figure 5.9), but the closed nature of Matlab drivers didn't allow any explanation.

When only channels 0 to 51 are considered, 39475 out of 40361 packets were successfully received or PER of $2 \cdot 10^{-2}$. Included are 364 packets that were received on a wrong, neighboring channel. Not included are 18807 packets which were received as an additional copy on a neighboring channel.

5.6.1 Improvements

The range of the receiver is very limited. There may be several reasons for this insufficient range. It may be caused by the antenna which was not impedance fitted to the demodulator input. RF optimization would be a long and expensive process but have little to no effect on the goals of this prototype. It may also be caused by improper matching of the differential IF pair between the demodulator and the ADC. It would need proper documentation from the manufacturers of both chips. The last reason could be any of the many simplifications necessary for the low computational power of the MCU.

There was a very large number of packets received additionally on a different than the original channel. From previous experiments, we know that the receiver architecture is often able to receive a deformed signal that leaks to a neighboring FFT frequency. It would point to a possible problem in clipping with a signal that is too strong. Yet, a distance of several meters is enough for the receiver to stop working. Both problems may be linked together.

A large improvement might be to use a more modern MCU like STM32H7. This MCU has slower ADCs, but a much faster Cortex-M7 core. Its ADCs can be joined in an interleaved mode and do up to 7 Msample/s. It would allow us to fully sample the IF and convert the sampled signal properly to zero IF before doing a complex FFT.

6 Proof of Concept Sensor

This chapter describes a Proof of Concept (PoC) network of a CU with wireless sensors. It should verify what sensor consumption is achievable with modern hardware and certain network parameters. Research in this chapter was done in cooperation with Department of Materials and Technology (KET) at ZCU. It is planned to be published later in a journal. In preparations for Chapter 7, this PoC uses more advanced SDR and the results from this chapter are not directly applicable to the receiver designed in the previous chapter.

The model situation would be a factory that handles dangerous gases. As an additional layer of protection, this factory might want to place sensors guarding gas concentration throughout the entire factory. Many sensors would be able to do long-term measurements and report concentration to detect even the smallest leak. In case of a big leak, the sensors should react immediately to stop the source of the leak and alert everyone to leave the area.

The wireless sensor needs to periodically measure and send the concentration of a dangerous gas. Additionally to that, there is also a trigger that is able to immediately report if the concentration increases over a given threshold.

The sensor needs to have low current consumption and survive more than a year without maintenance. Many sensors would be placed all around the factory, so replacing batteries too often would be complicated. The limit would be to match the batteries to the expiration date of the gas sensor and replace the entire sensor at one time.

Communication between sensors and CU needs to be reliable even in presence of interference. It also cannot have a longer delay, so the alert is transferred to CU as soon as possible. The sensor is using a random frequency channel for Tx and needs an all-channel receiver in the CU.

6.1 Sensor Hardware

Custom PCB was made for the sensor. The sensor is built around a combination of STM32L062 MCU and S2LP GMSK RF transceiver connected together with SPI. The complete PCB is 3×3 cm large. With batteries and the electrochemical cell, it is approximately 2 cm tall. Version `v03` of the sensor can be seen on Figures 6.1 and 6.2.

Previous work showed that an onboard antenna can be a potential source of problems. In this PCB, a U.FL connector was used instead. It can be connected

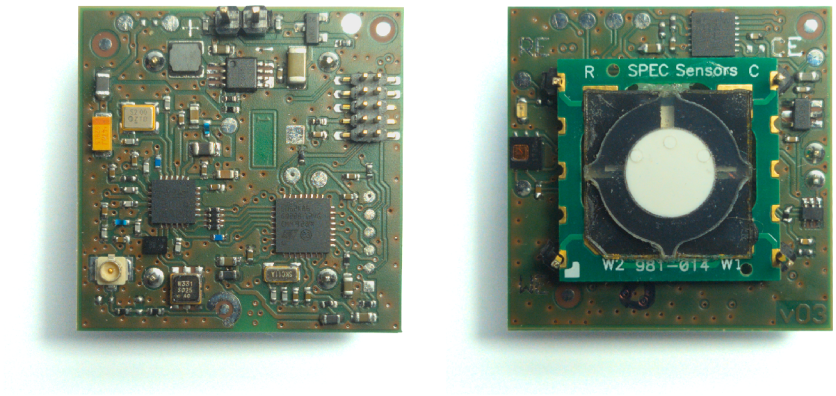

Figure 6.1: Hardware Platform

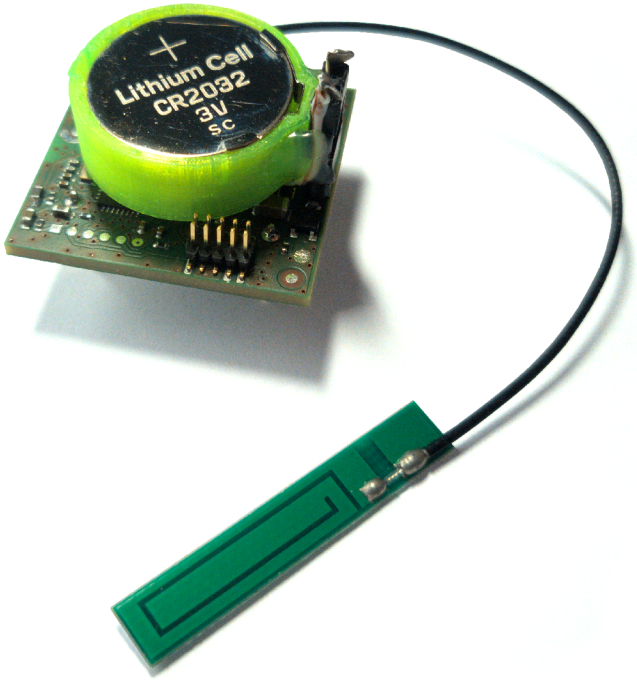

Figure 6.2: Hardware Platform with Batteries and Antenna

either to a sub-GHz antenna or via an attenuator and a 50 Ω cable directly to the transceiver in CU. In between the RF transceiver and the connector, there is an integrated Balun, made specifically for the S2LP by ST, and a Surface Acoustic Wave (SAW) bandpass filter tuned to 869 MHz. The SAW filter passes most of the unlicensed sub-GHz spectrum but attenuates everything else. The production device might work well with discrete components instead of both chips, but it should be a good starting point for this PoC.

The gas sensor is a 3-lead electrochemical cell from Spec Sensor. It needs a proper Analog Front-End (AFE) potentiostat LMP91000 to get a value measurable by the MCU's ADC. Apart from the analog output, the AFE needs to be controlled by an I2C bus. This AFE consumes approximately 10 µA continuously which is a major part of the 13 µA budget of 2 years on CR2032. Unfortunately, it also needs at least 2.7 V to operate. CR2032 battery starts at 3 V, but its voltage decreases as it gets depleted. For effective use of the battery energy, this AFE needs two CR2032 batteries in series and a power supply lowering the voltage to a value safe for the electronics.

The electrochemical sensor requires knowing the ambient temperature and humidity. Both values can be used in calculations of concentration, to estimate the sensor wear and to indicate failure if the sensor exceeds allowed parameters. There is an indicative temperature sensor embedded inside the MCU, but it is imprecise and influenced by the heat generated by the MCU itself. A more precise temperature and humidity sensor HDC2080 was added to the same I2C bus.

There is a cascade of power sources to make proper voltage for all components. Switching power source TPS6284x converts the voltage of 2 CR2032 cells to 0.2 V above the voltage used by the AFE with high efficiency. It is followed by a linear regulator making a stable voltage for the AFE. Radio, MCU and humidity sensor are powered directly from the switching power source. The board was designed for two assembly variants, one with a precise 3 V reference and another with a 2.8 V Low-Dropout (LDO) regulator. Precise low-power voltage reference at 2.8 V was not yet available when the PCB was made. On the battery connection, there is also a FET to protect the sensor from connecting the battery with inverse polarity.

6.1.1 Analog Trigger

The sensor needs a way to detect a sudden rise in gas concentration even if the MCU is sleeping. This PoC uses a combination of an ADC, a DAC and an analog comparator. On pin PA4 there is a 100 Ω resistor and a 10 µF capacitor. The pin is measured by the ADC and if the value differs from the required threshold for the sensor output, the DAC is enabled to force the correct voltage on the capacitor. After a delay of 1 ms, the DAC is turned off and the pin is measured again. Initial settling after reset can be estimated to $3\tau = 3RC = 3$ ms, but it will depend on tolerances of components and internal resistance of the MCU pin. When the measured value is close enough to the required threshold, the

analog comparator is enabled between PA4 and PA3, where the sensor analog signal is present. The MCU can go to a low-power stop mode. A change in sensor signal will flip the comparator and wake the MCU. Otherwise, the MCU will wake up after a long period to do a long-term measurement. Together with measuring the sensor output, it will check the drift of the voltage on the capacitor and, if needed, repeat activation of the DAC.

Another way would be to wake the MCU in short periods, sample the signal and compare the value. In the case of a slowly changing signal, it might be more energy efficient. Active comparator consumes 160 nA while an active MCU with an ADC consumes over 2.6 mA [60]. To get a reasonably quick one-second sampling with the same cumulative current consumption, one sample would have to be done in

$$T_{run} \times I_{run} = T_{period} \times I_{comparator} \tag{6.1}$$

$$T_{run} = \frac{I_{comparator}}{I_{run}} \times T_{period} = \frac{160\,\text{nA}}{2.6\,\text{mA}} \times 1\,\text{s} \approx 62\,\mu\text{s} \tag{6.2}$$

The exact T_{period} where sampling starts to be more efficient than using a comparator depends on the configuration of the ADC. Reasonable time to wake the MCU, configure the ADC and make a sample would be in range of milliseconds.[1] The turning point in favor of sampling will be at least tens of seconds which might be too much for some applications.

If the device doesn't need to periodically sample at all, the trigger threshold can be designed in such a way that the self-discharge of the capacitor will make the trigger value smaller. The MCU doesn't need to wake up to recalibrate at all and when the trigger crosses the sensor output, it will wake up, measure and tweak the threshold back to its correct value. It would require an extra several MΩ resistor for planned discharge as the ceramic capacitor can have a smaller leak than the connection to supply voltage inside MCU's pin and the value can itself drift up instead of down.

6.2 Gas Sensor

The sensor has two modes. In one mode, it makes 8 samples of the AFE output in a minute, averages them and stores one value a minute of current going through the electrochemical cell. In the other mode, it makes 2 samples a second and averages them to one value a second. The averaging was added to increase the precision of the gas concentration measurement.

The sensor can be set to a secondly or a minutely mode by the CU. If the measured value is higher than the trigger, the sensor switches to a secondly mode automatically. The electrochemical cell can have a longer time constant, so after the initial trigger, the value can be still increasing and needs to be measured often.

[1] Values achieved by this PoC are in Section 6.4.1.

The AFE has amplifiers to control a 3-lead electrochemical cell. One amplifier controls the difference between the Working Electrode (WE) and the Reference Electrode (RE) by setting voltage to the Counter Electrode (CE). Current going out of the WE shows the concentration of the measured gas. Depending on a sensor type, it can be in the range of nanoamperes. This current is amplified and converted to a voltage by another amplifier, so it can be measured by the MCU's ADC. The AFE has many options on setting bias between WE and RE, internal zero or gain of the output amplifier.

6.2.1 ADC, AFE and Power Source

Initial prototype `v01` had a pulse power source generating 2.8 V and a precise voltage reference of 2.5 V as a reference for the AFE. The AFE needs to be powered by 2.7 V, but it can use lower precise voltage as a reference for the operational amplifiers driving the electrochemical cell. It caused problems because the AFE is very sensitive to its supply voltage even if it uses an additional precise reference.

The switching power supply naturally has ripples in its output. For this reason, there are two inputs. One to set the power supply to a PWM mode which makes less noise and another to stop it completely. Both modes were tested with the MCU measuring a constant voltage. With the power supply in its normal mode, the standard deviation was $s = 7.79$ mV. Switching the power supply to PWM mode decreased it to $s = 1.19$ mV. Stopping the power supply completely during the measurements decreased it even more to $s = 0.95$ mV.

Unfortunately, the AFE is very sensitive to its power supply and stopping the power supply makes the supply voltage fall. What happens with the AFE output can be seen in Figure 6.3. The data in Figure 6.3 were gathered by an oscilloscope and passed through a 12.5 kHz lowpass filter. A similar ripple in the AFE output could also be seen when the transceiver transmitted a packet. That led to discontinuing the work on hardware `v01` and redesigning the power supply chain.

Fixed hardware `v03` uses a dedicated precise reference or an LDO only for the AFE. It didn't fix the ripples on the AFE output completely but reduced them significantly. Between the PCB revisions, we were struck by the electronics component outage and the power source needed to be replaced by the same chip in another package. It solved the issue of stopping the power supply because the other package doesn't have the stop pin. During the measurements, the power source is only switched to the PWM mode.

Two methods were added to further improve the sensor precision. The ADC can automatically do a 16× oversampling and produce a 16 bit output instead of its natural 12 bit. It slightly increases power consumption, but even with the oversampling the ADC conversion is done in 0.5 ms. The other method is to measure more frequently and average several samples. The ripples in the AFE output are changing slowly and the average will attenuate them. As a compromise between current consumption and precision, 8 measurements per minute

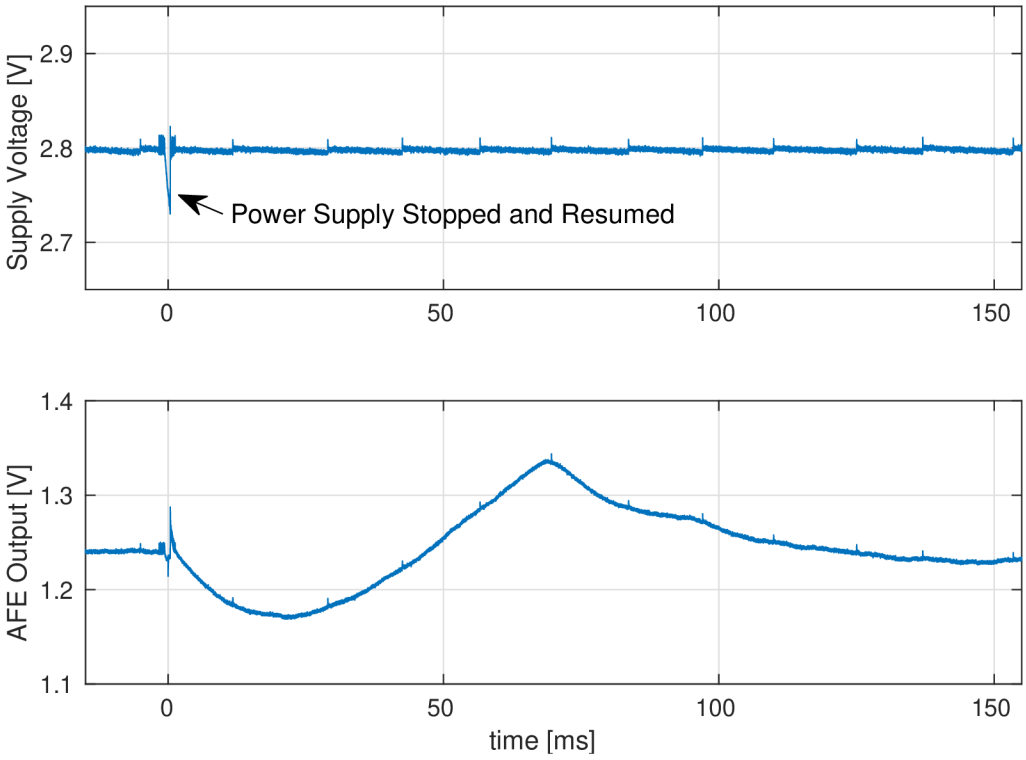

Figure 6.3: Ripple on AFE Output

or 2 measurements per second are averaged to produce minutely or secondly samples.

6.2.2 LDO or Precise Reference

Hardware `v03` was made in two versions. Regular LDO XC6206 is inexpensive, made for various voltages and consumes very little current. The other version was made with a more precise 0.05 % LT6656 voltage reference.

Two-sample F-test for equal variances was done to know if there is a significant difference between them. The same configuration and the same electrochemical cell were used with two hardware to gather data. The test ended with $p = 0.24$ and showed that there is no significant difference in output variance between hardware with LDO and precise reference.

6.3 Radio

A test setup was used to check the feasibility of keeping the Matlab Simulink receiver design. First, a packet was received with RTL-SDR, then it was decoded by the Matlab Simulink receiver model running in real time. Getting real-time data outside of Simulink turned out to be quite problematic. One working way is to use TCP to `localhost`. This was then redirected by a `bash` command

```
netcat -l -p 50000 | cu -l /dev/ttyACM0
```

to Communications Device Class (CDC) USB device. The USB device was a development board with an STM32L443 and an S2LP GMSK transceiver which sent the response on a correct channel back to the sensor. The delay from a packet end and a response start was between 60 ms and 120 ms. That is at least an order of magnitude more than the needed latency for a realistic sensor network design. High latency would require the sensor to stay in Rx with high current consumption and force it to repeat packets after a long delay. We weren't able to make this architecture work.

Another option would be to fix range problems in the Cortex-M-based receiver and connect a custom S2LP board as another peripheral to the LPC-Link 2. But since Chapter 7 required the use of a more powerful SDR together with a transmitter and a blank sensor, the all-channel GMSK receiver was redesigned. The SDR is used not just to receive, but also to send a response to the sensor. The same hardware and parts of the software were used for this PoC and experiments in Chapter 7.

6.3.1 Hardware and Channel Layout

New hardware opens a possibility to change the channel layout from Chapter 5. The license-free European sub-GHz regulations offer 7 MHz bandwidth, but instead of using the entire band, the GMSK baudrate was set. One of the preferred baudrates used in this area is 38.4 kBd/s. With the same condition of two samples per symbol, it is one FFT in $T_c = T_s/2 \approx 52\,\mu s$ and a sample rate of the SDR for 64 channels of 4.9152 Msample/s. The center frequency was set to 866 MHz which leaves approximately 1 MHz guard bands still in the allowed bandwidth.

Attempts with AD9364

There was an attempt to build the prototype on AD-FMCOMMS4-EBZ. It is a development board with AD9364 connected via FMC connector to one of Xilinx's Zynq development boards. Together it is a very powerful set combining SDR frontend, FPGA and fast ARM cores. It is more capable than LimeSDR Minis used in the end but requires a much higher investment in development. Proper reimplementation of the design on this platform would probably increase its performance. Most notable is the response time between packet received and response being transmitted.

When this platform was explored, the available examples were mostly outdated and hard to use. Nowadays, with Analog Device's Kuiper Linux based on Raspbian, the situation might have improved. It would still require deep knowledge of Zynq FPGAs and Linux driver development to be able to accelerate the signal processing.

LimeSDR Mini

LimeSDR Mini[2] is a crowdfunded hardware platform for LMS7002M made by the chip's manufacturer Lime Microsystems. It provides 30.72 Msample/s full duplex SDR through its USB 3 connection with PC.

It is possible to use this SDR with GNU Radio and some unofficial support is also available for Matlab Simulink, but it was dismissed with concerns of latency. This setup was not tested, but the use of Python programming language in GNU Radio would most likely increase the response time to similar levels as with the Simulink model for RTL-SDR. C++ together with the LimeSuite driver library was chosen for the SDR development.

This hardware can run the Rx and Tx streams synchronously from the same oscillator. It is possible to use timestamps and transmit the response with constant offset from the received data. It allows a linear design of the controlling software with receive - process - transmit loop.

The processing software needs to set itself a higher priority by

```
//Set priority
sched_param sch = {sched_get_priority_max(SCHED_FIFO)};
if (pthread_setschedparam(pthread_self(), SCHED_FIFO, &sch))
{
    cerr << "Failed to set priority: " << strerror(errno) << '\n';
    //Continue with regular priority for debugging
}
```

Otherwise, the operating system's scheduler might want to finish something first and the receiving program is delayed. The average time to process one frame is almost the same. Maximal time is usually an order of magnitude higher if the program runs with a normal priority which results in lost frames. The compiled program binary must be allowed to change its priority by

```
setcap 'cap_sys_nice=eip' program_binary
```

or it needs to be run with `sudo`.[3]

A large improvement in software speed was achieved by using 1020 sample sized frames for communication with the library. The LimeSDR Mini uses data blocks that are sized at an optimal power of 2. Part of these data blocks is used for control information, so the remainder for SDR data is 1020 samples large. The library can use 1024-sample frames and it is easier on the controlling software, but there is additional copying done in the driver which increases latency.

The size of frame buffers and offset between Rx and Tx timestamps can be configured. If the latency is set too low, the hardware starts to miss Tx timestamps and transmits 0 instead of the useful frame of data. With the mentioned

[2]The original hardware is no longer available due to the global chip shortage. Fortunately, a second crowdfunding campaign was launched for a LimeSDR Mini 2.0 at crowdsupply.com/lime-micro/limesdr-mini-2. It replaces an unavailable FPGA with a slightly better one. The original FPGA was almost full just handling SDR.

[3]These instructions are valid on Debian or its derivatives. Other distributions might use slightly different commands.

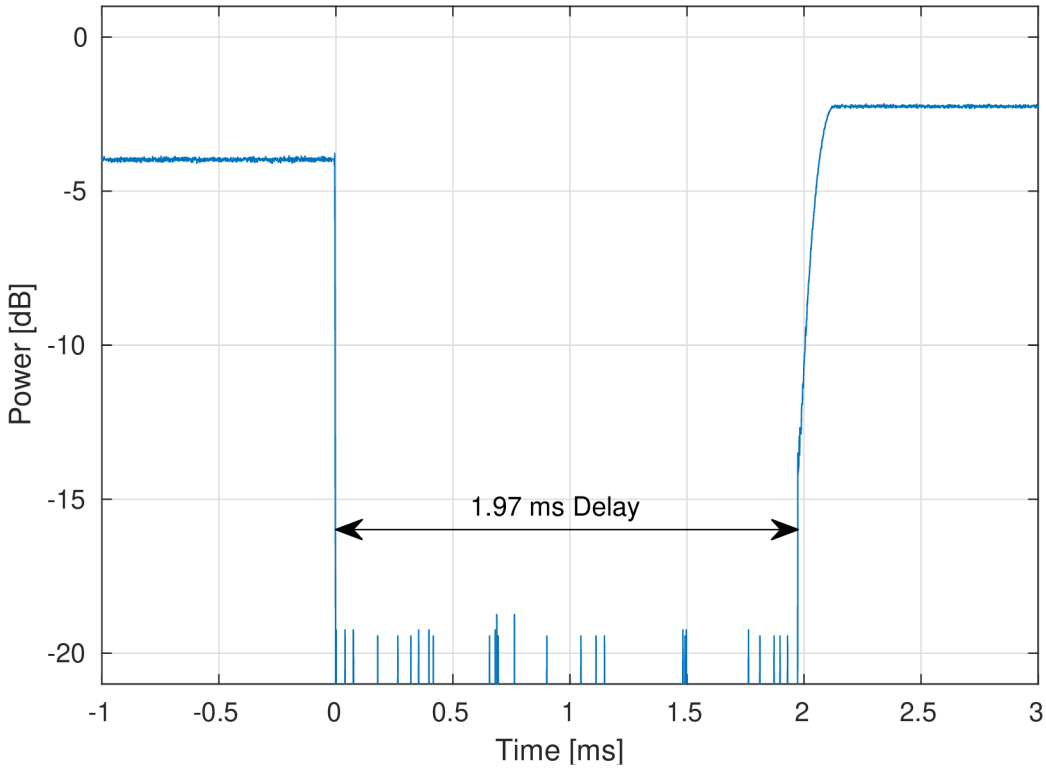

Figure 6.4: Delay Between Sensor's and CU's Tx

4.9152 Msample/s and 1020-sample frames, it is possible to reach latency between Rx and Tx of 4 frames or approximately 0.83 ms. This number doesn't include any processing time between obtaining Rx frame and giving out Tx frame. A real design needs to add at least one more frame or approximately 0.21 ms during which the frame data are processed. The latency value was not dependent on hardware. The same response time is possible with modern PC and with Raspberry Pi 4.

For this prototype, a value of 8 frames was used to increase stability and reduce the possibility of missing frames. Stability is important for data Rx and Tx, but for something such as a CCA algorithm, a lower latency would be possible. The resulting delay after a sensor's packet should be between 9 and 10 frames (1.87 ms to 2.08 ms), depending on when in the frame the sensor packet ends. The result can be seen in Figure 6.4 which was measured by RTL-SDR at 1024 ksample/s and passed through an RMS filter in GNURadio. The power values are tentative and depend on attenuators used to connect all devices.

Some additional improvements, for example even quicker CCA, might be accomplished by offloading some signal processing from the PC to custom FPGA firmware. Similarly to the Zynq option, the FPGA path was not pursued in this thesis.

6.3.2 SDR Receiver

The receiver is based on FFT as in Chapter 5. After the FFT, a liquid[4] DSP library was used to demodulate the GMSK. This library provides objects for signal processing, specifically for the creation of SDRs. The library uses standard C and dynamically allocated structures passed as function parameters instead of C++. Its emulation of C++ templates by macros is at times almost unreadable.

The liquid library provides `gmskmod` and its counterpart `gmskdem` modules to modulate and demodulate GMSK signals. It takes the difference between a phase of consecutive samples and puts it into a matched FIR filter. Whether the use of a matched filter has any advantage in the case of two samples per symbol was not investigated. For a receiver that could be loaded by several GMSK signals at once, it might be an inefficient use of computational power.

Unfortunately, the modules in the liquid library used for frame receival use a specific frame format and cannot be simply used together with common hardware receivers such as S2LP. S2LP can process several packet formats: IEEE 802.15.4, Wireless M-Bus, proprietary ST format and basic packet format, but none of them is close to the liquid frame. Liquid uses no preamble, 63 symbols of m-sequence as a syncword and its own scrambled header with information about CRC and FEC. Except for wM-Bus, all of the S2LP formats use up to 2046-bit preamble and at most 32-bit syncword. This receiver needs to work with a 32-bit preamble, 32-bit syncword, 8-bit length, data and 16-bit CRC.

There are 64 instances of a GMSK receiver class running at all times, one for each channel. Each of them goes through several stages while receiving a packet. Most of the instances should be in the computationally inexpensive *Power* state and waiting for a GMSK signal. In this state, they are looking for a sudden rise in received power by comparing two IIR filters `iirfilt_rrrf`. Input to both filters is a binary logarithm `logbf()` which represents signal power but is very easy to compute. One filter is set to a normalized frequency of 0.05 and the other to 0.02. If the output of the quicker filter is 5 `logbf()` units larger than the slower filter, the receiver advances to another stage. The value was set experimentally.

The following stages are separated after a FIFO, so they can be calculated in parallel by another thread. These stages also have a timeout, returning the receiver to the Power stage. Stage *Preamble* has a correlator `detector_cccf` looking for a pattern of zeros and ones. Stage *Syncword* has a longer correlator looking for a match with the syncword. When it is matched, the frequency offset produced by the correlator is used to set `nco_crcf` oscillator to compensate for the GMSK carrier offset. Stage *Receive* finally uses `gmskdem` to get packet data. Complete packets are checked with CRC and given to the application layer. There, the data are processed and a response is created.

[4] Available on GitHub as jgaeddert/liquid-dsp.

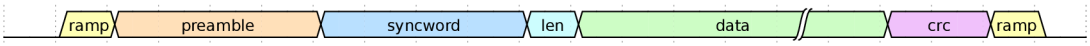

Figure 6.5: GMSK Sensor Packet

6.3.3 SDR Transmitter

The transmitter composes packet data in the same format: 32-bit preamble, 32-bit syncword, 8-bit length, data and 16-bit CRC, but adds one preamble byte with alternating bits at the beginning and one at the end for power ramping. A 6-symbol Hamming window is applied over the signal generated from these bytes for smooth power rise and fall. The result can be seen in Figure 6.5.

The first idea was to use the same structure as for the receiver. Individual modulators use liquid's `gmskmod` to generate 2 samples per symbol. All 64 modulators run in parallel and input to one iFFT which produces the resulting signal. It allows simply adding data to be transmitted disregarding how many channels are currently used.

In reality, it would probably suffer from high Peak-to-Average Power Ratio (PAPR) in a similar way as is common for OFDM. One of the methods of reducing PAPR [30] would have to be applied. There is also a peculiar modulation OFDM based M-ary FSK (OFDM-MFSK) [31] which uses a similar principle of transmitting FSK. Different from the transmitter described here, OFDM-MFSK doesn't use a signal phase for communication, so it can be used to reduce PAPR.

This construction showed very high Out of Band (OoB) interference. Figure 6.6 shows Tx on channel 26, 865.5392 MHz, measured with Resolution Bandwidth (RBW) of 1 kHz. The SDR Tx was connected directly to a spectrum analyzer via a 50-Ω coaxial cable. The useful signal in the middle can be received by a hardware GMSK receiver. In addition to that, there are pairs of sidelobes. Each couple is centered in one neighboring 76.8 kHz channel. The peaks would not break the European sub-GHz regulations [52], but only because the SDR transmits in almost 5 MHz at once and is unable to produce 14 dBm in one narrow channel.

An improvement might be to attempt perfect reconstruction by using overlapping iFFTs and mixing the overlapping output signal blocks with a windowing function. It would require doubling the rate of iFFT calculation, so this method was not pursued.

The solution was to get rid of the iFFT and modulate the GMSK signal directly on the SDR sample rate. The `gmskmod` is set to generate 128 instead of 2 samples per symbol. A harmonic signal made by `nco_crcf` with a required channel frequency is then applied to shift the signal to its correct channel. The transmitted result, shown as the other line in Figure 6.6, doesn't have any problematic emissions outside of the channel and would nicely fit into the OoB limits even with 40 dB amplification. A disadvantage of this solution is that only one signal can be modulated at one time. If two sensors wake up at the same time, only one of them can get a response and the other would need to repeat

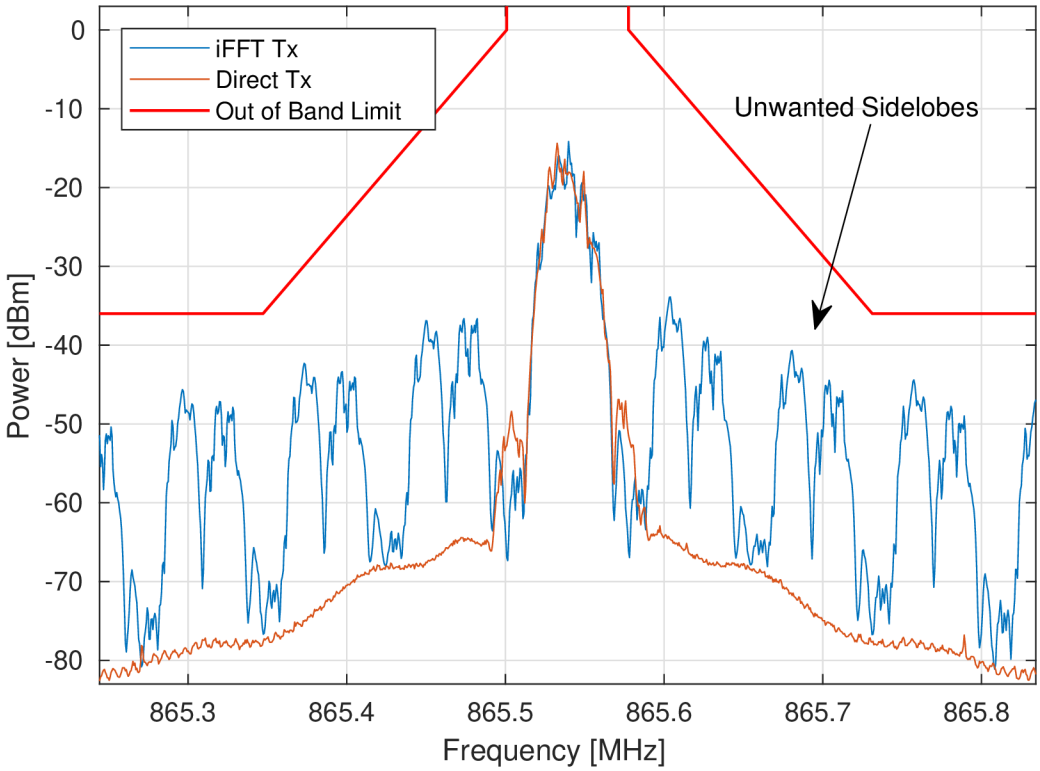

Figure 6.6: The Spectrum of GMSK SDR Transmitter

its message. In our scenario, it will not influence when the information gets to the CU. It will only slightly increase the power consumption of the unlucky sensor. The solution with this particular SDR has also much lower Tx power compared with a solution with SDR Rx and hardware transceiver Tx.

The different constructions of the SDR Tx can also be seen on the power ramp in Figure 6.7 which precedes the packet preamble. The Tx with iFFT has visible steps in the output power. There are also glitches visible between each symbol for the Tx with iFFT. They may be linked with the unwanted spectrum sidelobes. Figure 6.7 was measured by RTL-SDR at 1024 ksample/s and passed through RMS filter in GNURadio. The power values are tentative. Output power measured by a calibrated spectrum analyzer can be seen in Figure 6.6.

6.3.4 S2LP

The S2LP radio is controlled via SPI and one interrupt pin. The firmware uses `S2LP_Library` enclosed in STSW-S2LP-DK support package for S2LP. This library allows controlling most of the S2LP radio but is poorly maintained. On top of known S2LP hardware bugs, it adds a few bugs of its own.

S2LP chip has a channel offset feature. The user sets channel spacing and channel number and the S2LP will add the necessary offset to the carrier frequency. It only works if the channel offset fits perfectly to the register resolution. Otherwise, rounding errors will shift the 64th channel completely away

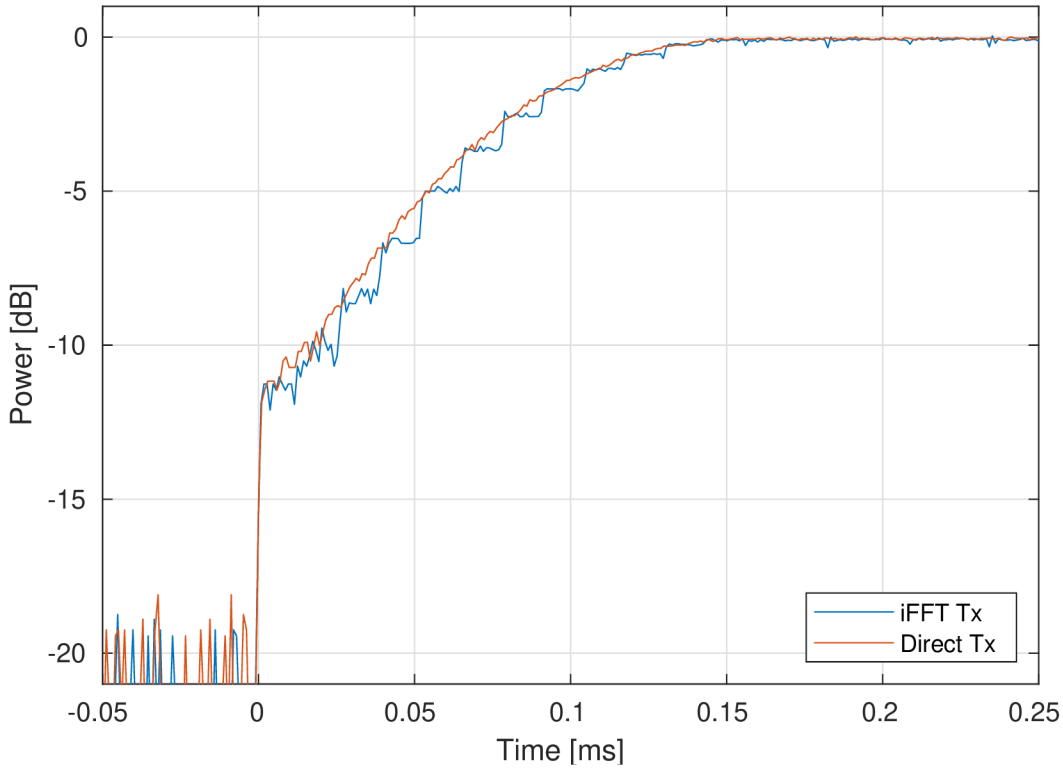

Figure 6.7: Power Rise of GMSK SDR Transmitter

from its required position. It is necessary to calculate channel offset manually and change the carrier frequency before each Tx.

Each radio transaction happens in two stages. First is Tx including CCA done automatically by the S2LP. The radio will listen in Rx for a pre-configured time and if the channel is clear, it automatically starts Tx. The minimum that can be configured is the time of 64 symbols, 1.7 ms in this case. S2LP can even do a random backoff and try again, but it was not used here. This sensor is set to try Tx once and if the selected channel is occupied, it retunes and tries again on a different channel.

The second stage is Rx where the sensor waits for a response from CU. If no response is received within the time limit of 20 ms, the transmission is repeated on another channel. S2LP has an embedded timer that is able to cancel the Rx operation and alert MCU via the interrupt pin.

6.3.5 Packet Format

The packet from the sensor contains temperature, humidity, battery voltage and data of the electrochemical sensor current. The data portion of the packet can include several minutely or secondly spaced values and a timestamp in seconds.

The sensor continuously stores values of current and after 26 minutes transmits all 26 of them. Instead of higher OSI layers, unacknowledged data are

repeated after the next measurement. There are 5 attempts to transmit the data and receive an acknowledgment. If unsuccessful, the sensor sleeps for another minute and tries to send 27 values. At most 30 values can be sent in one packet and after that, the earliest value is lost. It is also a limit when the CU should consider the entire sensor lost.

If the concentration threshold is reached, the sensor creates an extra packet with a different format. It has only one current value, an approximate timestamp and a separate numbering to prevent duplicates. The extra packet is sent up to 15 times before giving up. After that, it is replaced by the nearest periodically measured value. If the measured value is over the threshold the sensor switches to a secondly measurement and tries to transmit every second. The sensor should not stay in this mode for a long time or it will quickly deplete its batteries.

6.4 Current Consumption

One of the main parameters of the PoC sensor is its cumulative current consumption which translates to an expected life on its battery. These measurements show what consumption is possible with this particular gas concentration sensor and also approximate what consumption would be possible independent from the AFE, just MCU, ADC and communication with the CU.

Results were measured with the analog trigger disabled. It lowered the measured value by 160 nA consumed by the comparator [60] and a small current consumed by the DAC. The DAC is enabled only sporadically when the capacitor self-discharge changes the voltage outside of the preset limits. It depends on many variables and would only add more randomness to the measurement.

The sensor in these measurements uses GMSK at 38.4 kBd/s while the receiver in Chapter 5 was built for 18.75 kBd/s. The time that the sensor spends in Tx and Rx would be doubled and the consumption would be somewhat increased. This difference would be smaller than the difference between a sensor that transmits measurement data and a sensor that transmits a binary value. Such a sensor would be for example a magnetic door sensor that needs to report only its presence and state of the door. There is also an option to reduce the number of channels and use the higher baudrate. That would not influence sensor consumption but network robustness.

During the measurements, the sensor was randomly selecting any one of the 64 channels. The SDR of the CU was responding on the same channel. The S2LP transceiver in the sensor was connected with Rx and Tx of the SDR via 50 Ω coaxial cables and appropriate attenuators.

6.4.1 Detailed Consumption Profile

Figure 6.8 shows current consumption during one sensor activity. The sensor measures the last of 26 samples and reports all to CU. This figure was measured

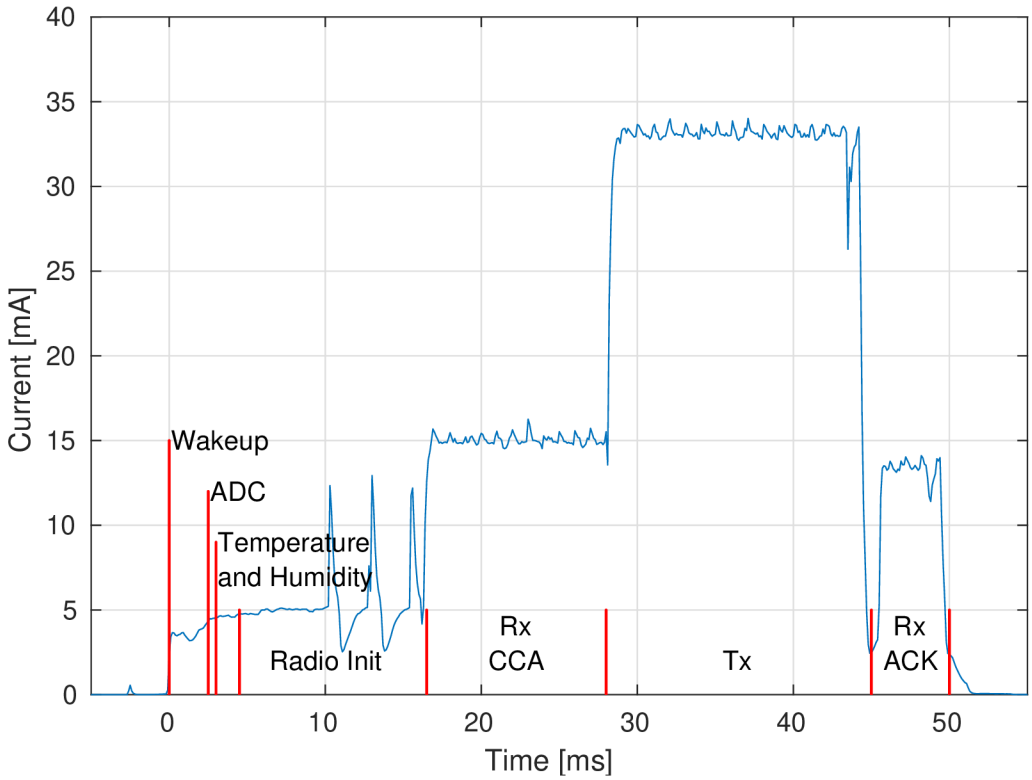

Figure 6.8: Detailed Consumption of Sensor Reporting to CU

by Power Profiler Kit II from Nordic Semiconductor providing 2 V to a sensor with all power sources removed. The switching power supply adds a lot of noise to the current measurements due to its pulsating nature. The consumption in Figure 6.8 is higher than with the switching power supply, but the overall shape of the curve and energy consumption should correspond to a complete sensor. The consumption of a full sensor could be estimated by using the ratio of voltages and switching power supply efficiency. This piece also doesn't have the AFE and the electrochemical cell. The ADC measures from a floating input. The removed AFE reduces the consumption by 10 µA [61].[5]

The firmware goes through these stages visible in Figure 6.8:

- 2.5 ms below 5 mA waking up the MCU and clock.
- 0.5 ms below 5 mA doing the ADC measurement.
- 1.5 ms below 5 mA temperature and humidity measurement.
- 12 ms at 5 mA initializing the transceiver. This time can be shorter if the radio is in standby mode instead of powered off, but it would increase the current consumed while the sensor is sleeping. It can be beneficial with a shorter report period.

[5] If it could run from 2 V.

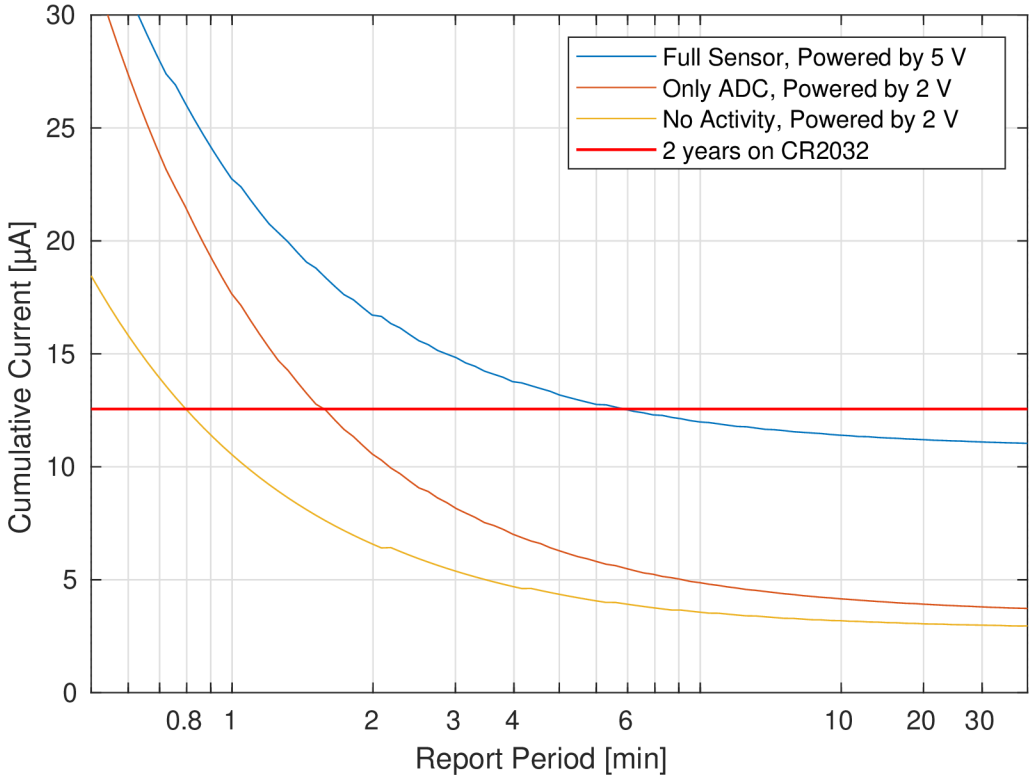

Figure 6.9: Sensor Cumulative Current Consumption

- 3 ms to 27 ms at 15 mA receiving a randomly selected CCA. These values come from the CCA mechanism in S2LP and can be different for a different transceiver or sensor network design.
- 5.5 ms to 17 ms at around 40 mA[6] transmitting data to CU. Duration depends on how much data is needed.
- 4.5 ms to 6.5 ms below 15 mA receiving and waiting for a response from CU. Depends on how quickly CU responds and on the amount of data. The method of encryption used by the sensor network may increase the size of the response significantly.

Apart from the activity of reporting to the CU, the sensor also wakes up to measure. It makes a bump in the consumption of less than 3 ms wide and less than 8 mA tall.

6.4.2 Cumulative consumption

Figure 6.9 shows approximate predictions of consumption depending on the period between two reports to CU. The cumulative current consumption was

[6]The actual value varies between 30 mA and 45 mA. It is probably caused by non-ideal 50 Ω coaxial cables. Transceiver Tx consumption can be influenced by a load on its RF output.

estimated as a mean of instantaneous current with the communication with the CU from Figure 6.8 in the center. Varying the duration of the averaged area with exactly one report to CU corresponds to a sensor that would change its report period. This way it is possible to estimate current consumption for up to almost twice the period used by this PoC. All data were measured by the Power Profiler Kit II.

The **full gas sensor in minutely mode** was powered by 5 V. It is the maximum possible with the measurement tool and corresponds to two CR2032 batteries in the middle of their life. The sensor was measuring 8 times a minute, averaging that to one sample a minute and reporting to CU once every 26 minutes. The cumulative current was measured over the 26 minute report period. Full sensor with the NO_2 detector in version with the 2.8 V LDO was consuming 11.2 µA. The version with LDO can also be seen in Figure 6.9. Full sensor with the NO_2 detector in version with the precise 3 V reference was consuming 13.9 µA. That is a bit over the limit for two years on two CR2032.

The **full gas sensor in secondly mode** was measured with the NO_2 detector in version with the 2.8 V LDO. It was measuring 2 times a second, averaging that to one sample a second and reporting to CU once every 26 seconds. The cumulative current for 26 second report period, averaged over 17 periods, was 48.7 µA. This is too much for coin cell batteries but could work as a more powerful sensor powered by four AAA batteries.[7] This version is not present in Figure 6.9.

A **generic sensor without the switching power supply and without AFE** was powered by 2 V. This could represent what would be possible for a different sensor. The ADC measurement timing was the same as with the full sensor in minutely measurement mode. The sensor was consuming 4.1 µA. It shows that there would be a reserve for a sensor running two years from a single CR2032. This combination can also be seen in Figure 6.9.

The last version is a **generic sensor that doesn't do periodic measurements**. It represents a sensor that guards a digital input or uses the analog trigger. It only needs to report to CU that it still exists. Due to low-power timer management in firmware, this variant was waking up once a minute only to immediately return back to sleep. This version of the sensor spent much less time in Tx because it didn't have any measured values to send. The cumulative current was 3.1 µA. It is the last curve in Figure 6.9.

[7] The limit for two years on AAA batteries would be somewhere around 50 µA.

7 Mixed Network with GMSK and OFDM

A more advanced solution for the problems of frequency agile sensors might be to use the high bandwidth camera as the all-channel receiver. The cameras have complex OFDM receivers which have a similar construction as the SDRs used above. This solution will require a larger investment into development and hardware than the previous option. Part of the Section 7.2 with initial evaluation was published in [32].

7.1 Proposed Network

The proposal is to build a network combining OFDM and frequency agile GMSK. High bandwidth communication uses radio transceivers that are very close to an SDR or entirely software-defined. That is why it would not require a lot of additional hardware to add GFSK communication into the OFDM transceivers in cameras and the central unit.

I have rejected the idea of incorporating sensor communication into the camera feed in an Orthogonal Frequency Division Multiple Access (OFDMA) fashion. Synchronization between individual transmitters would be complicated [33] and probably require much more advanced hardware in sensors. The sensors need to be very energy efficient and cheap which is fulfilled only by common GFSK transceivers.

FSK modulation can be used in OFDM-MFSK [31] where one of each group of M subcarriers contains a signal and the rest are silent. Its application to smart meters shows advantages in robustness [34] which might be beneficial in a security system. An OFDMA version of OFDM-MFSK with $M = 2$ or $M = 4$ could perhaps be received by the simple FSK transceivers. It would be a way to connect the sensor directly to high bandwidth communication and an interesting topic for future research. In the most basic layout, just the OFDM header could be FSK encoded with network information and hole position for the sensor.

The solution explored here is to make holes into the regular Phase Shift Keying (PSK) OFDM signal by zeroing several neighboring tones. The sensor will start transmitting as soon as one OFDM packet ends and the OFDM radio needs to detect that and create a hole in the next packet to not disrupt the sensor. The sensor can continue its GMSK Tx and follow with Rx of an

acknowledgment. The sensor packet can be received directly in the CU or one of the cameras and routed via the OFDM link.

Getting the information to the CU is the important part, but the sensor needs to receive an acknowledgment, so it can go back to sleep to save power. The CU or a designated router will immediately respond on the same frequency with acknowledgment and allow the sleep mode. This can also be used to give the sensor updated information about the network and perhaps to synchronize the sensor into an FHSS.

7.2 Signal Simulation

As a first step to determine whether the proposed system can be built, a simplified model was created in Matlab Simulink. For the sensor communication, I have selected GMSK with symbol filtering of $BT = 0.3$. Minimal frequency deviation of GMSK will have small bandwidth to fit into an OFDM hole. One GMSK symbol was simulated with 240 samples.

Simple quadrature PSK was used for the OFDM signal, initially with $n = 64$ sample FFT and later $n = 128$ and $n = 256$. The cyclic prefix was set to $n/4$. The OFDM symbols were not window mixed together to keep the model as simple as possible. There were 6 and 5 empty guard tones on the edges and no pilot tones. The receiver was simulated coherently with the transmitter so no pilot tones or packet headers were needed for synchronization. The signal was upsampled by 3 before being mixed with the sensor signal. There were 1, 2 or 4 GMSK symbols per one OFDM symbol to match the sample rates.

The initial FFT size was selected to fit GMSK channels to OFDM tones while satisfying the minimum number of GMSK channels for a hypothetical FHSS [45], [52]. The $n = 64$ OFDM signal has a distance between two tones of $1.25 \times f_s/240$, where f_s is the sampling frequency. The bandwidth of the GMSK signal can be again estimated by Carson's rule

$$f_{bw} = 2 \cdot \left(\frac{BT}{T_s} + f_{dev} \right) = 1.1 \times f_s/240 \tag{7.1}$$

where BT is the Gaussian symbol filter, $T_s = 240/f_s$ is the duration of one GMSK symbol and frequency deviation $f_{dev} = 1/4T_s$. It shows that with these parameters, one channel could barely fit in one tone.

The proposed composition of OFDM, GMSK and AWGN can be seen in Figure 7.1. The GMSK signal has a unit power, the same as the noise. The OFDM signal would have unit power if all 64 tones were used. Its power was not compensated for the empty tones. In the case of Figure 7.1, there are 5 empty tones centered on the GMSK signal. With guard bands, it is 16 empty tones and a power of $3/4$.

For $n = 64$, a full 3-dimensional matrix was simulated with the strength of both signals varying between $-20\,\mathrm{dB}$ to $30\,\mathrm{dB}$ plus the special case with no signal. The strength of the signals is relative to the situation in Figure 7.1.

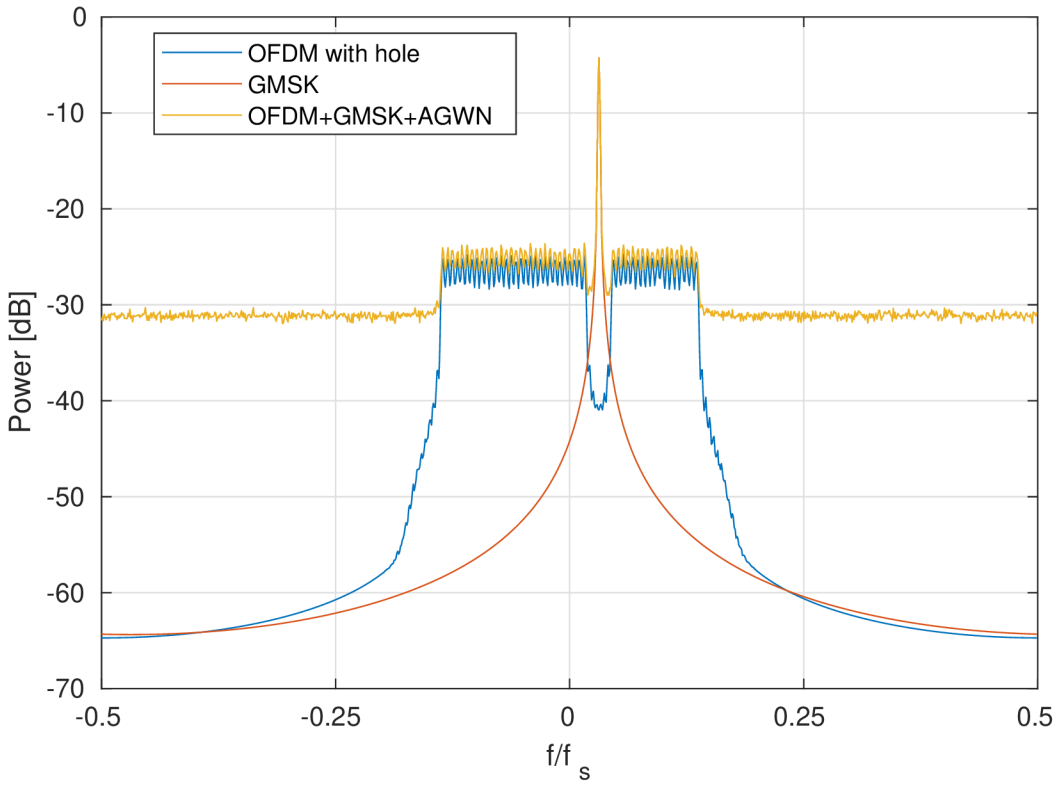

Figure 7.1: The Spectrum of the Communication

The third dimension was a number of empty tones. A portion of these was simulated needlessly to fill the matrix completely and ease later processing. For $n = 128$ and $n = 256$, only a cross was simulated as two slices of the complete matrix necessary for the figures.

7.2.1 Interference to GMSK

Figure 7.2 shows how the BER of the GMSK signal is affected by the OFDM signal for $n = 64$. The power of noise and the power of individual OFDM tones stays the same as in Figure 7.1. The power of the GMSK signal is varied relative to the unit power. The GMSK signal power can be related to SNR, but we have to consider the OFDM signal as noise and correct the value for relative bandwidth. The GMSK signal has a relatively narrow bandwidth, so only a small fraction of the AWGN and OFDM has any effect on the GMSK performance.

One of the curves in Figure 7.2 shows the performance of the GMSK without the OFDM signal as a reference for comparison. The rest are simulations with different hole sizes. We can see that a gap of one tone centered on the GMSK signal has the largest influence. Zeroing a gap of 3 or 5 tones can improve the signal a bit more, but has a much smaller effect. The effect of 3 or 5 empty tones is slightly larger when there is a larger difference between the

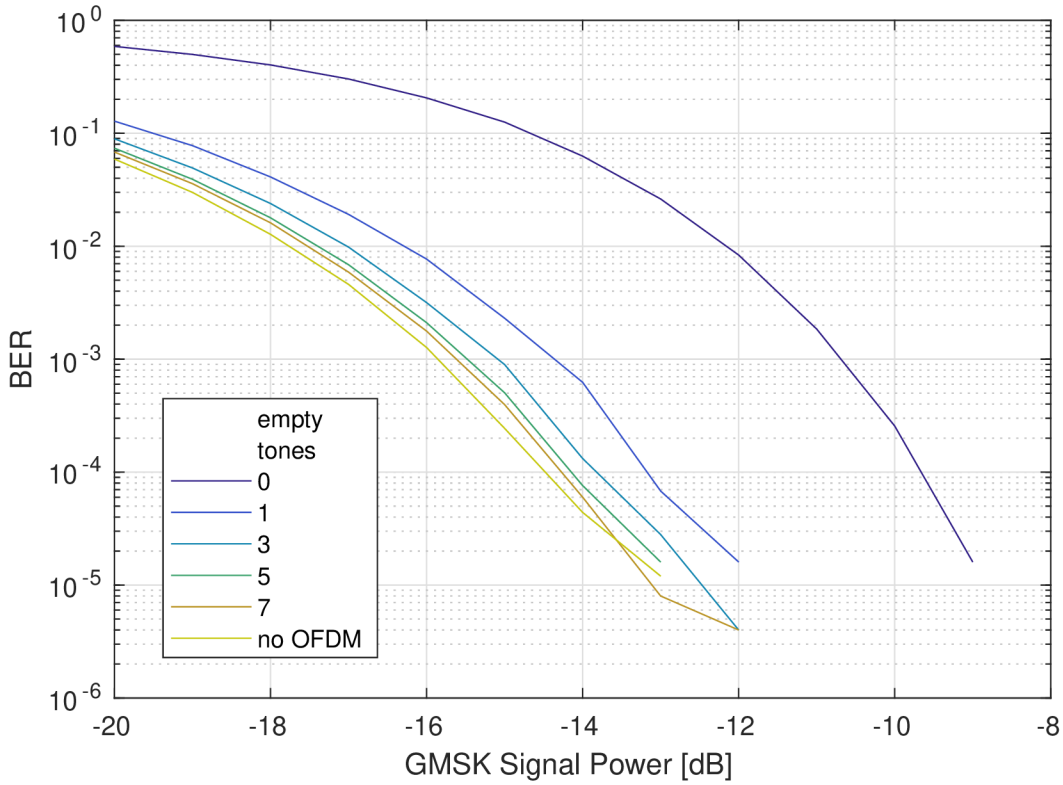

Figure 7.2: Bit Error Rate for GMSK Signal

power of the OFDM and the AWGN. The situation where one signal is overpowering the other will be shown later.

7.2.2 Interference to OFDM

Figure 7.3 shows how the BER of individual OFDM tones is affected by the GMSK signal. The power of the GMSK and noise is set to 1 while the OFDM power is varied. The tone number means distance from the GMSK signal. Tone 0 has the same frequency as GMSK, tone 1 is an average of the two neighboring tones and so on.

Same as before, the power of OFDM can be related to SNR, having in mind that AWGN power is spread to three times as much bandwidth while GMSK power is concentrated mostly in a bandwidth of a single tone. There is a reference curve of BER of all tones without any GMSK for comparison.

We can see that the tone with the same frequency as the GMSK signal is mostly unusable. The tones right next to it are still strongly affected. Tones further away come closer to the performance of the OFDM without a GMSK signal. Same as in the previous section, zeroing 3 or 5 OFDM tones would have the wanted effect. Zeroing 3 or 5 tones would be beneficial for both signals.

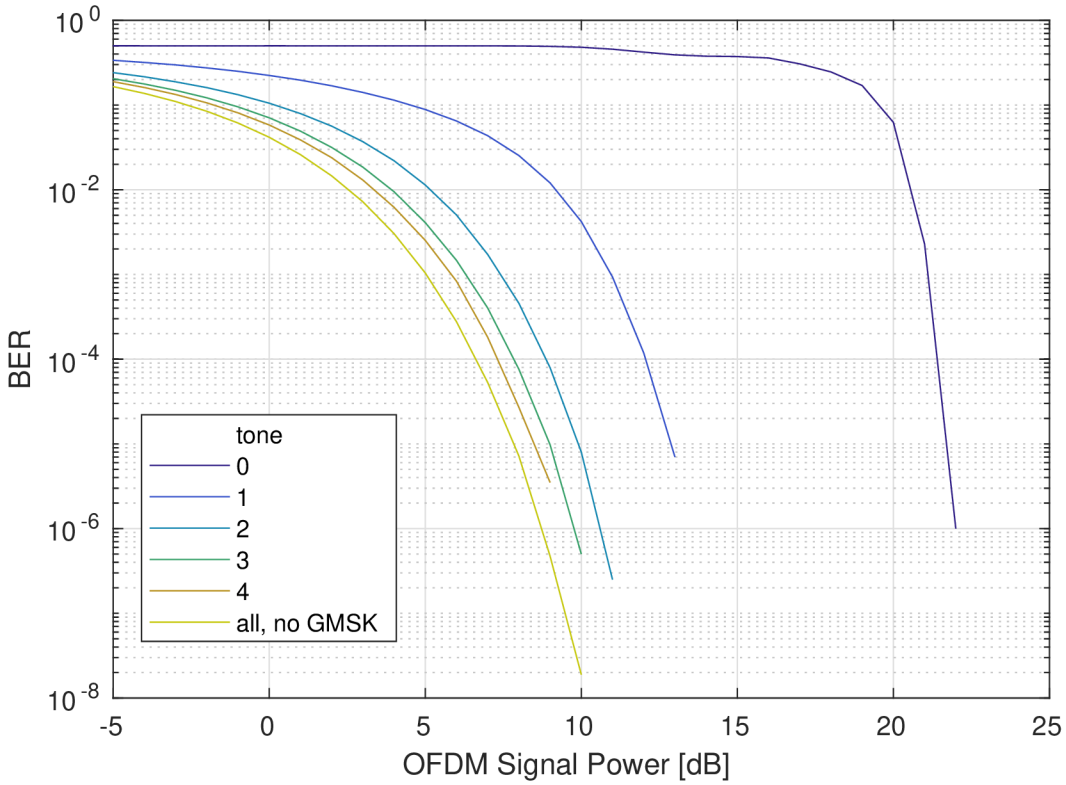

Figure 7.3: Bit Error Rate for Individual Tones of the OFDM Signal

7.2.3 Relative Signal Strength

The previous two sections failed to show the area in which both communication links can work at the same time. In Figure 7.4, simulated for $n = 64$, we can see the area where none of the signals is usable and where both are usable. The gradient is fit between BER of 10^{-2} and 10^{-6}. For the OFDM the BER means errors in bits transmitted on all used tones.

Figure 7.4 shows that in the case without empty tones, the two modulations cannot work together. The stronger signal would always disrupt the weaker signal or none of them would work. With an increasing number of empty OFDM tones, a corridor appears where both communication types can work at the same time. The corridor is about $20\,\text{dB}$ to $25\,\text{dB}$ wide for the case of 5 empty tones. More than 5 empty tones has a smaller effect on the corridor.

We can use the free space loss equation to compare the change in signal strength to a change in distance. The change allowed by the corridor in Figure 7.4 can be related to physically moving one of the transmitters relative to the receiver.

$$L_{FS} + L_c = 20log(d \cdot d_c) + 20log(f) + 20log\left(\frac{4\pi}{c}\right) \tag{7.2}$$

where L_{FS} is the original free space loss, L_c is the change in signal strength, d is the original distance, d_c is the change in distance and $20log(f) + 20log\left(\frac{4\pi}{c}\right)$ are constants in this consideration. We can see that a change of $20\,\text{dB}$ com-

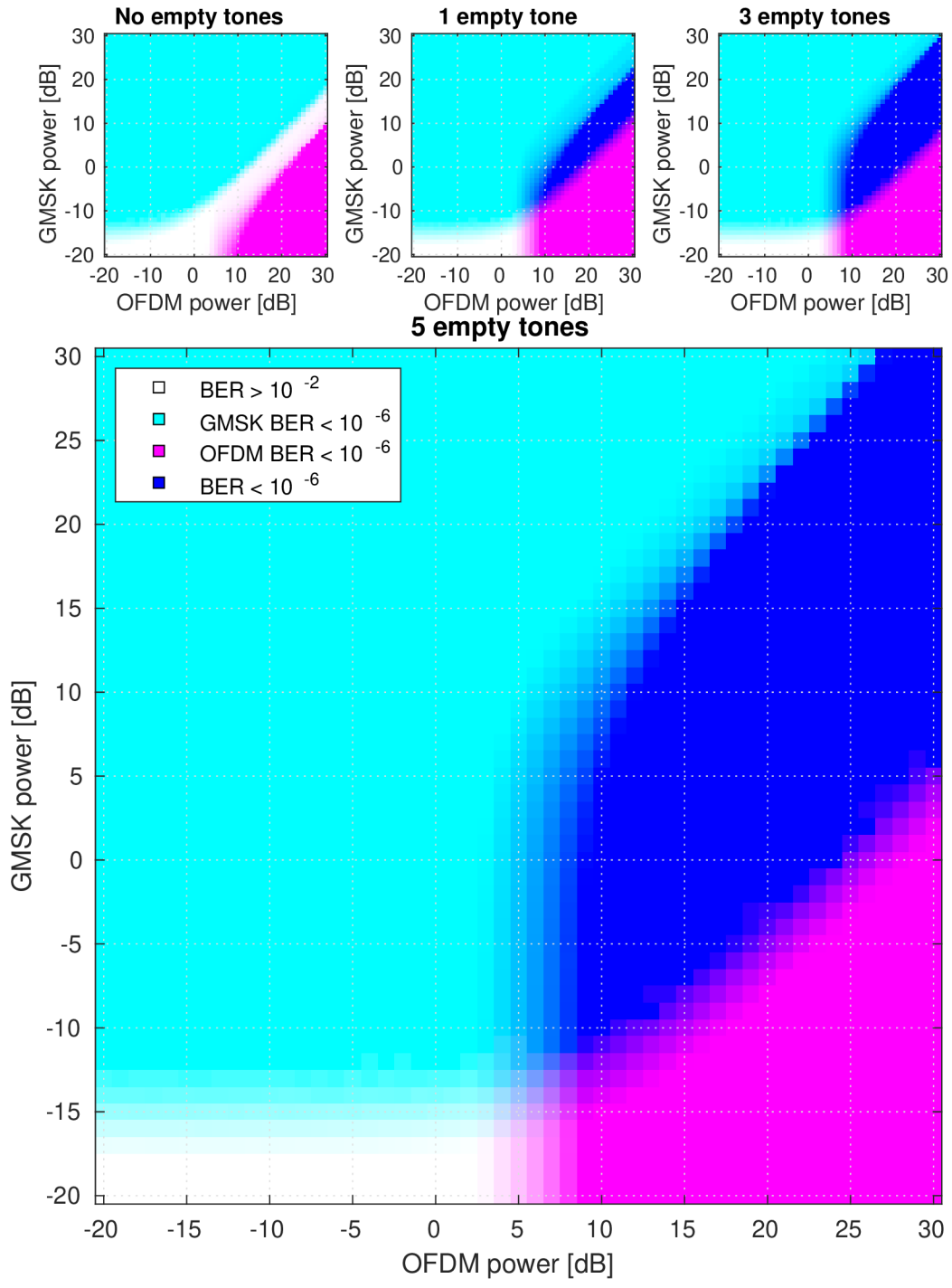

Figure 7.4: Bit Error Rate for interfering signals

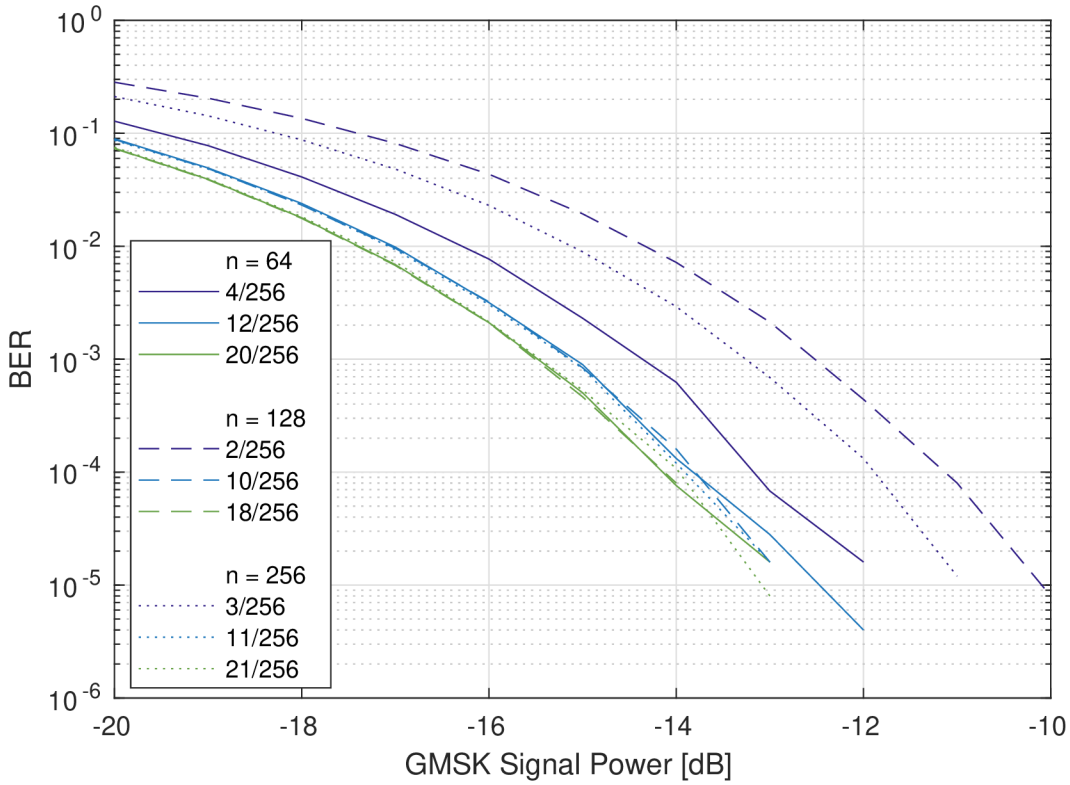

Figure 7.5: Bit Error Rate for GMSK Signal

pares to a $10\times$ change in the transmitter distance. For our model situation, if the farthest transmitter is 100 m from the receiver, then the closest has to be at least 10 m far. That is not enough for a practical network. The network will require controlling transmitter power to fit the right power at the receiver. Most devices in a security system are stationary, but a device such as a key fob is movable and will be problematic in this setup.

The simulated figures show that the OFDM signal is much less robust. The transmitted power allowed by government regulations usually stays the same, even though it is used to transmit much more data. In a physical system, this will lead to a shorter range for the security cameras compared to simple sensors. This is one more reason why having a secondary backup GMSK communication is necessary even in security cameras where Wi-Fi satisfies all criteria. The robust link can be used at least to report tampering and interference in the main high datarate link.

7.2.4 Different FFT Size

The effect of different FFT sizes should also be considered. Figure 7.5 shows the effect on the GMSK with OFDM of different sizes. Figure 7.6 shows the effect on the OFDM with different FFT sizes. The number of zeroed tones was chosen to get a gap of similar size for different FFT sizes. The same gap is not

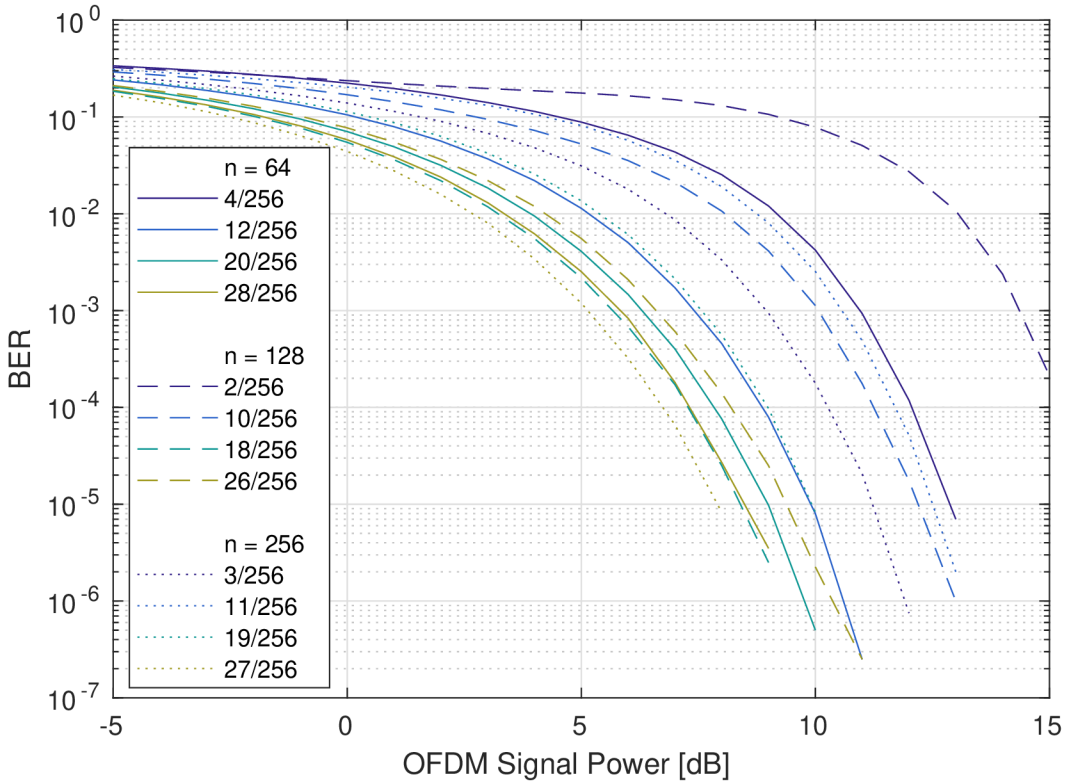

Figure 7.6: Bit Error Rate for Individual Tones of the OFDM Signal

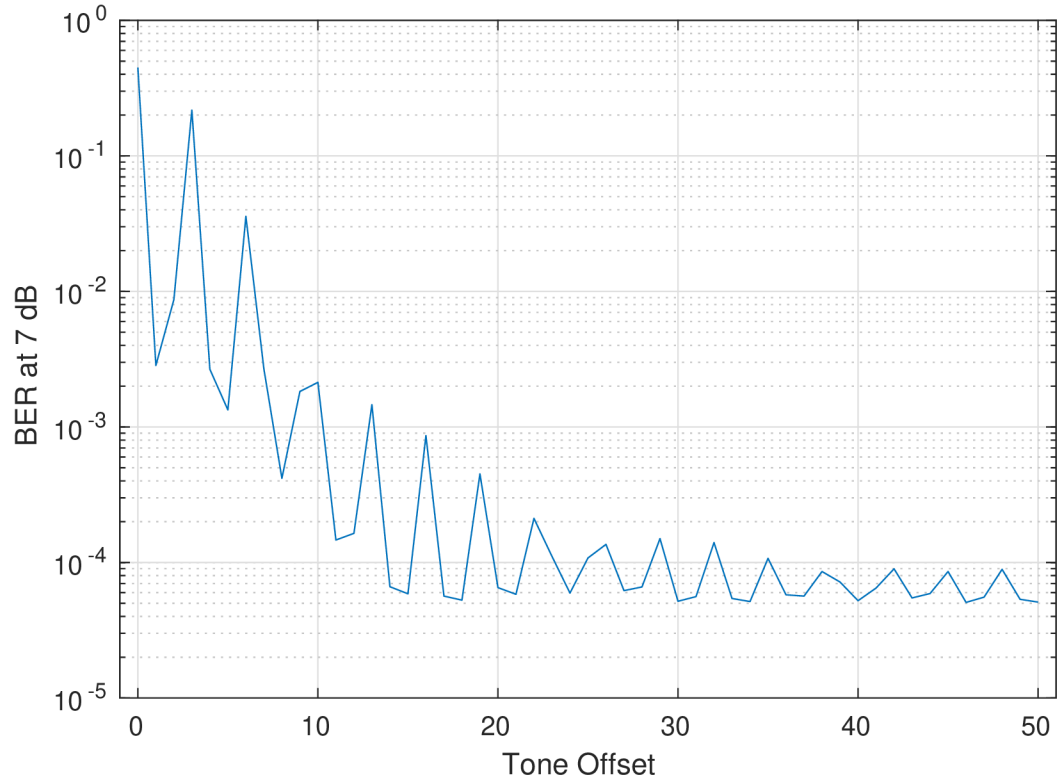

Figure 7.7: Bit Error Rate at 7 dB for Tone Offset

possible because the gap should be an odd number of tones, but also 2 or 4 times larger for the $1/2$ or $1/4$ FFT. The hole was set between 1 and 7 tones for $n = 64$ and a maximal still smaller value for bigger ns. The size of the gap in both figures is marked as parts of 256, so they can be compared.

Figure 7.5 shows that the size of the gap is important, but the size of the FFT is not. The first three lines show a difference because there is a large difference between the actual gap sizes. For more zeroed tones, the differences are negligible.

In Figure 7.6, the plots depict BER of tones that would be on the edge of a gap with the size written in the legend. The situation is more complicated here, as wider FFT show some suspicious behavior. In some situations, a larger gap is worse than a narrower one. More can be explained by Figure 7.7 which is made only for $n = 256$. It contains BER for multiple tones and fixed power. The GMSK signal is centered at tone 0. The plot shows peaks once every approximately 3.2 tones. Similar but a weaker effect can be seen with $n = 128$.

One explanation might be that peaks of the $sin(\pi x)/\pi x$ spectrum collide with some components of the GMSK only for some tones. The width of the $n = 256$ OFDM tone and distance between zeros in $sin(\pi x)/\pi x$ is $0.3125 \times f_s/240$. The ratio between GMSK symbol rate ($f_s/240$) and tone size would be $1/0.3125 = 3.2$ which might be a coincidence or an explanation. The question is whether the same effect will happen with realistic oscillator tolerances, symbol shift between the two modulations or windowing of OFDM symbols, but this effect was not investigated further. With the data in Figure 7.7, it seems reasonable for $n = 256$ to eliminate at least 7 tones and get rid of the first peak.

7.2.5 Limitations of this Simulation

We can question the validity of the simulation if we have no information about differences in the simulated GMSK demodulator and physical GFSK transceivers available on the market. From the available information, it seems that physical receivers digitize both IQ components at an intermediate frequency while the Matlab demodulator uses a baseband signal. Documentation of physical receivers mentions configurable low-pass filters and lots of specifications for blocking and selectivity, but Matlab block was receiving complex baseband signal with all the OFDM signal present on higher frequencies. A similar thing can be said about the Matlab's OFDM receiver.

The real receiver will also need to handle packet detection and synchronization which is not a part of this simulation. The differences need to be verified on a working prototype before a definitive answer can be given.

This simulation was done without symbol windowing. It might have been a wrong decision because it could make tone bandwidth narrower and reduce interference between the two modulations. Adding a symbol windowing could improve the cohabitation problem in exchange for less data transferred.

7.3 Implementation of an OFDM with a Hole

As a starting point, the liquid DSP library was used. There are two high-level classes for OFDM communication, `ofdmflexframegen` and `ofdmflexframesync`. These define complex packet structures for automatic selection of modulation, CRC or FEC. In our situation, it would be more of a burden than a simplification. Performance of the `ofdmflexframesync` as measured by liquid's author can be seen in Figure 7.8.

One layer lower, there are `ofdmframegen` and `ofdmframesync` which handle frame synchronization, but leave packet layout to the user. These two were rewritten to C++. It was less work than it seems, as liquid is object-oriented code made with C tooling, and it made making changes a lot easier. Three C++ classes were made, `HoleOfdmCommon` and two derived classes, `HoleOfdmGen` and `HoleOfdmRec`. This layout corresponds to the original split of common functions in liquid.

The goal is to implement an SDR OFDM transmitter and receiver which can make a hole in the communication. During the development, the OFDM was set to $n = 256$ with a 32-sample cyclic prefix and a 24-sample tapering between symbols. Default tone layout was used, which for liquid is $1/10$ of nulled tones as guardband above the signal and $1/10$ below. The DC tone is also nulled. Pilots are spaced one in every 8 tones. One hole was set 7 tones wide. Most of these parameters could easily be changed after the development.

From the example situation, the sensor GMSK communication should be sparse. Only a few packets are expected at the same time. The implementation was done for the possibility to add two holes and two GMSK signals. It would probably be more difficult to add more holes later.

7.3.1 Liquid OFDM Frame

The OFDM receiver needs to detect the start of a frame, estimate channel parameters and select a packet type if there is more than one.[1] The liquid receiver uses three OFDM symbols: two short `s0` symbols and one long `s1` symbol. Both `s0`s are used to detect the frame, do coarse and fine time synchronization and for carrier offset compensation. The `s1` is used to estimate the channel and prepare the receiver for data symbols.

Liquid OFDM implements coarse synchronization in the frequency domain which is in contrast with some common methods that use only the time domain [35]. The `s0` symbols have an m-sequence of 1 or -1 on even tones and 0 on odd tones. Instead of a regular cyclic prefix, `s0` symbols are composed of a ramp 2 cyclic prefixes long and two copies of the iFFT block. The ramp is another copy of the iFFT block with $sin()^2$ tapering at the beginning. It is the same function used to mix together consecutive symbols in windowing.

[1] Sometimes the term Physical Layer Convergence Procedure (PLCP) can be found, but Wi-Fi for example in its current specification does not use this term at all [62].

The receiver looks for a packet by processing one FFT every n samples. It compares the received frequency domain data with the expected `s0` symbol. From that, it calculates the `s_hat` metric by summing phase differences between all used tones and their $+2$ neighbors. The `s_hat` is used to set a coarse timing of the symbols. The receiver repeats the estimation two more times with $n/2$ samples offset. From the average `s_hat`, it readjusts precise symbol timing. In the end, it uses the time domain data and the original `s0` for maximum likelihood estimation of the carrier offset.

After `s0`, there is one `s1` symbol. This symbol is transmitted as an ordinary OFDM symbol with a regular cyclic prefix and windowing with `s0` at the beginning. The `s1` symbol is created from the m-sequence of binary PSK on all used tones. The receiver will put the received data through an FFT and compare them to the original `s1`. If there is a sufficient match, it will advance to channel estimation, otherwise, it waits for another $n/2$ samples and repeats this step. If the `s0` symbol and the `s_hat` metric are periodic after $n/2$, the `s0` match could have happened on any one of the periods. The `s1` match will ensure that the receiver will correctly start processing Rx symbols no matter on which `s0` period it detected the frame.

The last step is to estimate the channel response to rotate and amplify individual tones. The receiver compares the received symbol `s1` with the original and creates a 4th-order polynomial to compensate for tone power and a separate polynomial for the tone phase. These polynomials are used to create a complex vector with compensation for each tone.

During the Rx of data symbols, all tones are first compensated with the vector created from `s1`. Then the receiver compares pilot tones with their original value. Pilots are made with binary PSK from the same m-sequence in the transmitter and the receiver. The difference of pilot phases is approximated with linear function and its slope is passed through a simple IIR filter. The result is applied to the data tones additionally to the `s1` compensation. The constant value of the phase compensation is also used to tweak the carrier frequency compensation.

The resulting complex vector with compensated data tones is given via callback to upper layers. It might be `ofdmflexframesync` or any custom layer with any packet format. When the upper layer receives enough symbols, it signals back and the receiver is reset to wait for another frame.

The performance of the `ofdmflexframesync` was measured by the library author and is depicted in Figure 7.8. This was measured for $n = 64$ and is specific for the `ofdmflexframe` header modulation and coding.

7.3.2 Adding Holes

Both transmitter and receiver need to know the layout of tones in advance. In the case of the liquid library, the tones are fixed when creating the transmitter and receiver objects. That does not allow adding holes into the communication. One of the first changes was to make the tone layout changeable in the

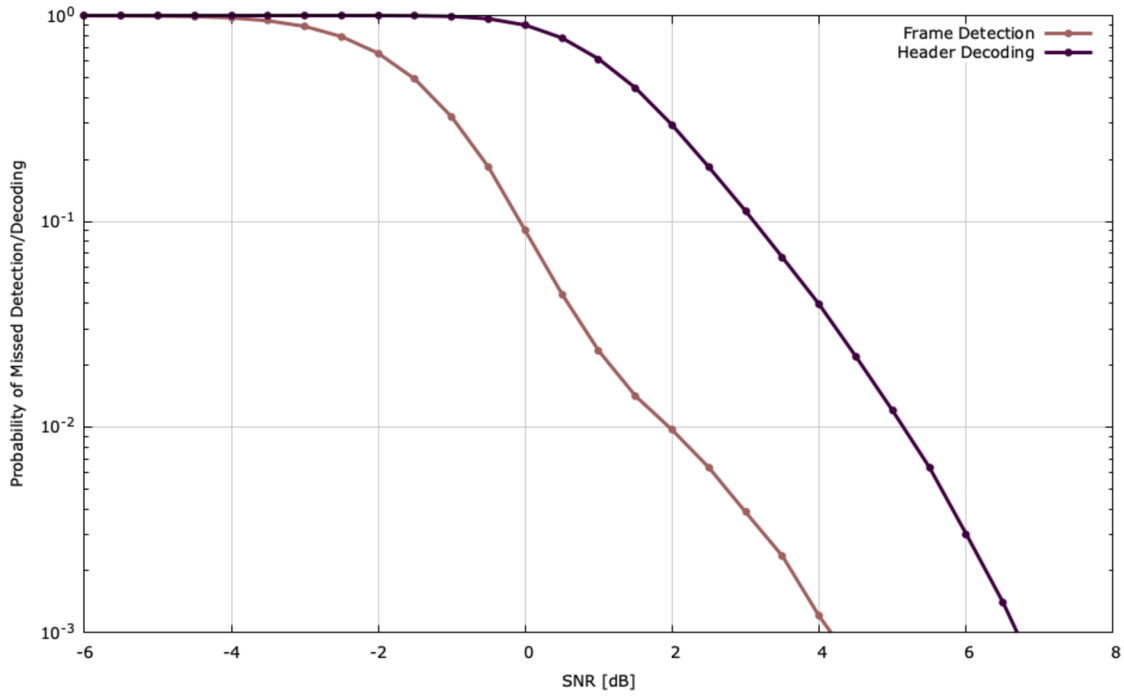

Figure 7.8: Liquid Flexframe Performance [63]

`HoleOfdmGen` transmitter when a packet is to be sent and not when the transmitter is created.

Much greater problems turned up when modifying the `HoleOfdmRec` receiver. Experiments showed that the receiver is unable to work if the tone layout is different from the one used for Tx. Synchronization with a hole in Tx and no hole in Rx works with high confidence, but the constellation of the output is as if it was passed through a mixer.

The first idea was to look into the `s0` synchronization. With an increasing number of holes, the synchronization gets worse and then stops working completely. When looking for the `s0` symbol, pairs of $+2$ neighboring tones are multiplied together and the sum is compensated by their count. There is small negligence in the count as there are one fewer neighboring pairs of tones than there are used tones. When a hole is added, this difference is increased. The offset was fixed to divide by the real number of nonzero products and the synchronization started to work even with small blocks of tones divided by many holes. Real performance would probably suffer with fewer tones used, but the fundamental problem was solved. Testing showed that the problem was not in this stage of the receiver and its influence was minor. In our situation of up to two holes, this fix would make almost no difference, so it was later removed.

A second wrong idea was to split the detection into blocks between two pilot tones. Synchronization symbols do not contain pilots[2], but missing blocks in

[2]Or are `s0` and `s1` made only of pilot tones?

synchronization would seamlessly continue into data symbols where the 7-tone hole for GMSK would be placed between two pilots. Each independent block could be calculated separately during the synchronization. Outliers can be eliminated from the set of metrics and then composed together to get one synchronization metric. This might make the synchronization work regardless of whether there is a hole. This method could be used for both `s0` and `s1`. Further testing showed that this also was not the main problem and it would restrict the GMSK channel placement, so it was not used.

The main problem was found in the channel estimation of `s1`. After a correct match, the receiver gets the difference between wanted and received `s1` and uses polynomial fit over all used tones. When there is a hole in the `s1` symbol, the phase received is completely random and the phase compensation is as well. The problem with the phase is that it is limited between 0 and 2π. The receiver checks if there is a step of phase compensation between two neighboring tones. A step indicates that the phase has wrapped over and needs to be unwrapped. If there is a difference larger than $+\pi$ or $-\pi$, it adds or removes 2π to make the phase curve smooth. The unwrapping algorithm will start correctly, but during the hole, it randomly adds several multiples of 2π and continues with that until the end. The 4th-order polynomial fit tries to smoothly link one part of the symbol with the other which is multiple turns above. The result cannot be used to compensate the channel, instead, it mixes all the received data beyond repair. Similar but less serious is the situation with power compensation. Unused tones have very low power, which influences the resulting polynomial around the hole. The same effect with phase and power was also happening later during the 2nd order polynomial pilot tracking.

The solution was to add a second `s1` symbol and encode DBPSK data into it before the channel estimation is made. This modulation should be tolerant to the channel parameters. Depending on whether the second symbol's phase is the same or inverted, we get as many bits of information as there are tones. Each empty tone will return a random bit, so the position of both holes needs to be encoded in the remaining tones and not influenced by the data on the empty tones.

The first idea was to mark both holes with ones on both sides. An algorithm can be made to look for the hole positions even when the holes are overlapping. The algorithm needs to find the longest consecutive sequence of zeros and then mark two or a single hole. There is a hard limitation on only two holes. Limitation on the number of used tones and size of the holes is satisfied for the selected parameters with a large margin. This solution has proven to be sensitive to those bordering bits which are also the ones most likely to flip.

A much simpler solution is to take hole positions and encode them with high redundancy into the available data while not considering which bits won't be transmitted. In the simplest, the position information can be repeated many times. One hole will overlap only one copy of the information. During Rx, individual bits are counted and the more frequent value is used. A 16-bit number

is enough for the selected parameters, one B for each hole position. If the hole is not used, it can be put to the guard bands where it has no meaning.

With $n > 256$, the hole position would have to be encoded differently. The GMSK signal can only appear on one of 64 channels and only 6 bits of information are needed. The other 2 could be used to configure the size of the gap.

Instead of a simple majority, a proper FEC could also be used. Or more elaborate non-standard FEC encoding could be created. The transmitter has the information about which bits will not be received, so it could encode the hole position only to bits that are not part of the hole. Design of an own FEC would be a lot of work and using an already existing and validated FEC should be a better option.

When the hole positions are decoded, a new layout of tones is known and can be used to finish the equalization. The phase and power corrections from the first `s1` can be used, missing tones eliminated and a correct polynomial fitted. After these modifications, the receiver started giving reasonable data.

7.4 Simulation of Frames with Holes

In parallel with the implementation, the receiver was continuously tested with a simulation. The same program was modified when the implementation was done to test the differences between the modified receiver and the liquid original. The simulation is a C++ program that runs an instance of the transmitter, adds AWGN and runs an instance of the receiver. Each run of the simulation was done for 4 sets of transmitters and receivers: the original liquid set, `HoleOfdm` set without any holes, with one hole and with two holes. The hole positions were selected randomly on one of the used tones.

Each run of the simulation used a combination of C++ `random_device` to seed a `default_random_engine` which should be enough to prevent any artifacts caused by some poor-quality pseudorandom generators. Before the simulated signal, there was a random delay in length between 3 and 4 OFDM symbols. Added noise was generated once and applied to all 4 signals. The only exception was the second `s1` symbol which is not used in the liquid OFDM frame. After the header, there were 256 symbols of data. Each symbol had binary PSK data on even tones and quadrature PSK data on odd tones. This layout was copied from a liquid example code. At the end of the frame, there was one more OFDM symbol of just noise to flush out any block that might still be stuck in the receiver.

The receiver callback had access to the transmitted data and the correct hole layout. It counted errors and marked frames with a wrong hole layout. Error in a symbol was counted as one error even in cases where the 4-state symbol would give both decoded bits wrong. Frames that were not detected were marked after the run.

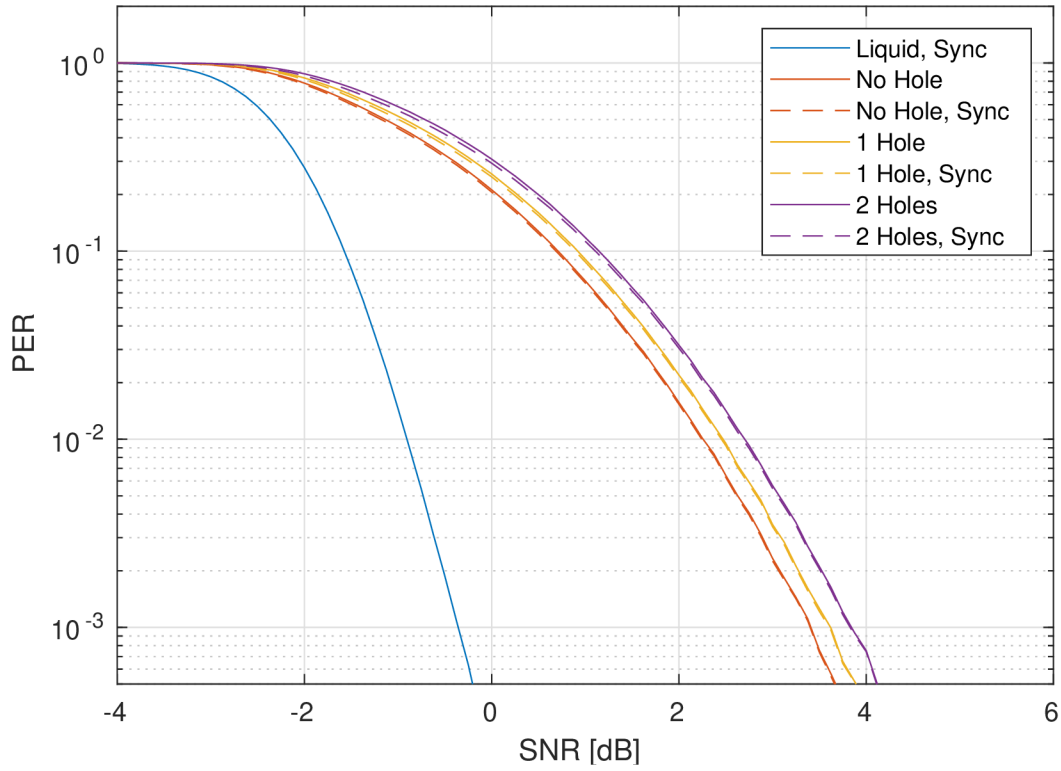

Figure 7.9: Comparison of OFDM receivers, Frame Detected

There was an attempt to speed up the simulation by using multiple parallel threads. Both liquid and `HoleOfdm` transceivers need FFT library to work. Both can either use liquid's built-in FFT or `FFTW3` from the operating system. `FFTW3` had decades of development, so it should be the most optimized and the fastest. It was optimized for multiple architectures and can easily run on x86-64 PC as well as on ARM which will be the target application of the transmitter.

As it turns out, `FFTW3` is not thread-safe when creating and deleting FFT plans. Usually, the plans are created in advance and then run many times. The problem is with `HoleOfdmGen` which needs to regenerate symbol `s0` on each Tx and at that time it creates, uses and destroys an iFFT. With some modification, the transmitter could probably use the same iFFT plan which is used for data symbols. To prevent this and more possible collisions, the simulation was instead split into different programs. Each running program has its own allocated memory and can simulate its own fraction of the overall simulation.

7.4.1 Results

The Figure 7.9 shows that the modified `HoleOfdmRec` is much worse at detecting the frame and synchronizing. The curves show packet miss rate or PER if success is the detection of the frame or packet and synchronization. The dashed lines show where the packet was detected, but the hole layout decoded from the second `s1` was wrong. Full and dashed lines are very close which means

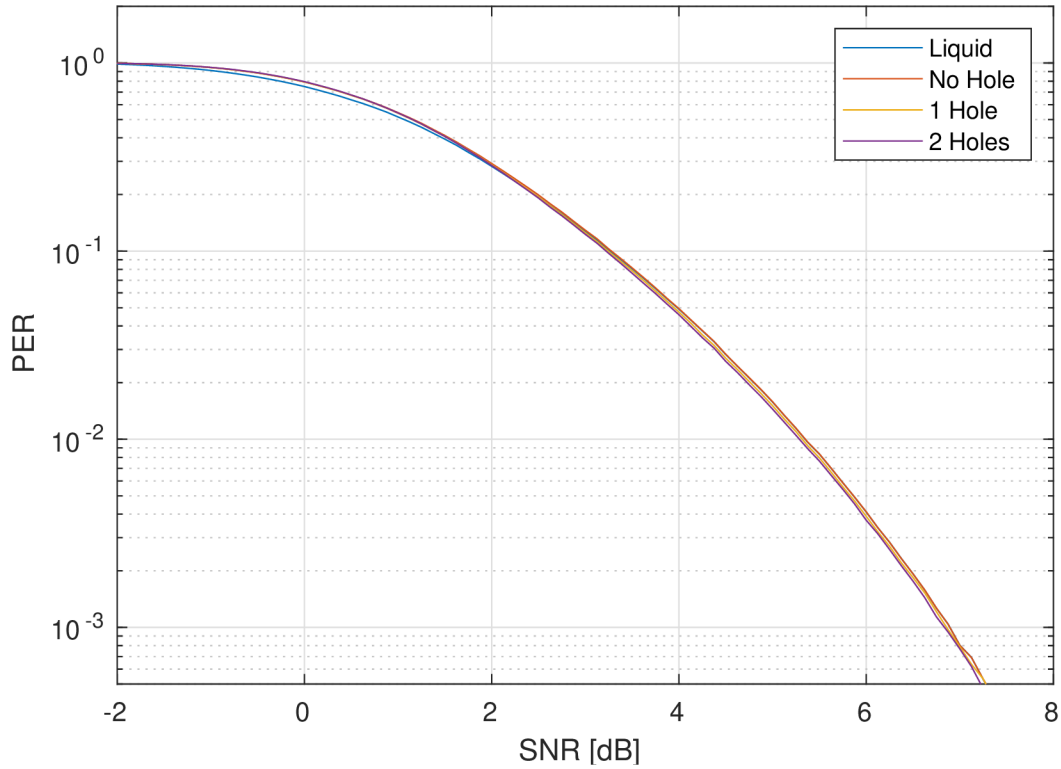

Figure 7.10: Comparison of OFDM receivers, 50% Symbols Correct

that when the packet is detected, the hole layout will be most likely correct. Detection and synchronization alone do not tell anything about the quality of the synchronization and the frame data. Even though the liquid receiver is able to detect many more packets, they will be mostly unusable.

Figure 7.10 shows that when the threshold for a successful packet is 50% data symbols correct, both liquid and `HoleOfdm` receivers are comparable, at least on this AWGN channel. These curves can also be compared with the header decoding curve in Figure 7.8. The `ofdmflexframe` header uses FEC and needs some portion of bits correct to work. The similarity between the figure created by this simulation and a figure provided by the liquid's author suggests that the results are correct and both receivers are in fact comparable. The last Figure 7.11 shows a performance of the receivers if success is 99% data symbols correct. At this level, the errors would be easily fixed with basic coding.

7.5 Prototype Network Implementation

The next step was to implement a basic camera, CU and a sensor for a prototype network. The camera needs `HoleOfdmGen` together with a source of video and a detector of the GMSK communication. The CU needs `HoleOfdmRec` with the GMSK transceiver from Chapter 6.

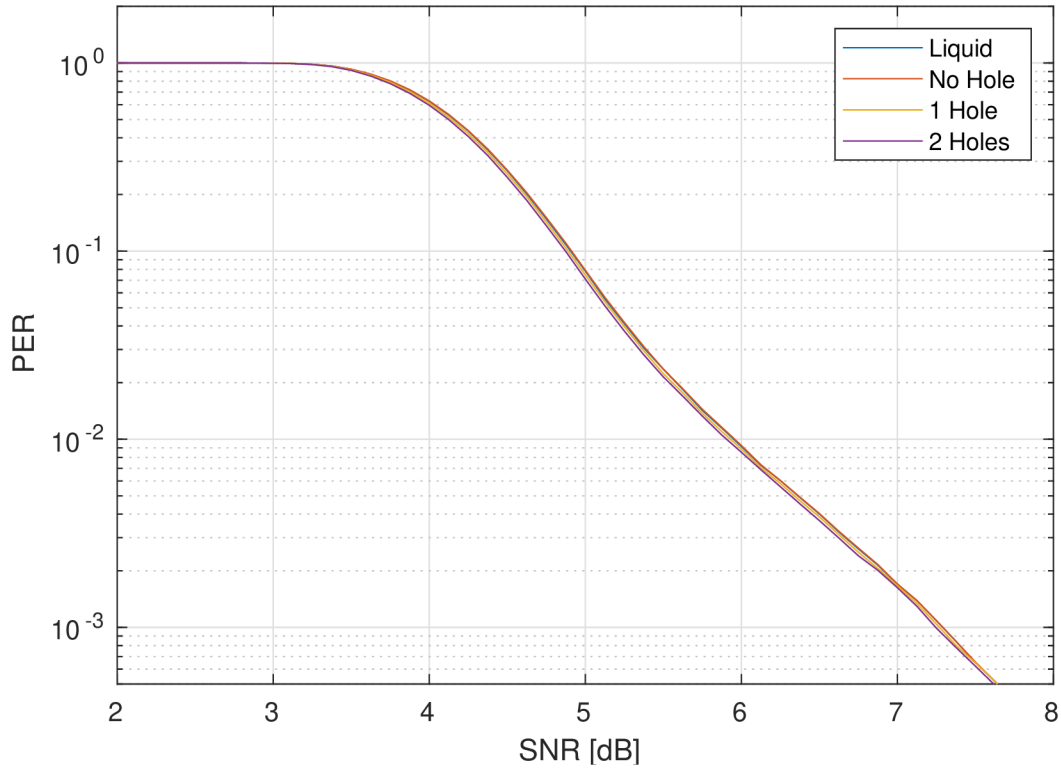

Figure 7.11: Comparison of OFDM receivers, 99% Symbols Correct

The development and testing of the prototype network were done entirely on 50 Ω coaxial cables. LimeSDR Mini transceivers had approximately 43 dB between Tx and own Rx while there were at least 3 dB more for each T-splitter in the path to other transceivers. RTL-SDR was used for the inspection of what signals are present. This setup was used instead of antennas for two reasons. It would not be polite to fill the entire sub-GHz SRD spectrum with the testing signal. Production devices need to keep track of their Tx and obey duty cycle limits, which is not true for tweaking and testing. There will still be some signal escaping the cables, but it shouldn't inconvenience nearby sub-GHz users. The second reason is that the first simulation showed that it may be necessary to control the Tx power. That functionality is regularly used today and there would be no benefit in implementing it yet again in this prototype. Fixed attenuation and manually set Tx power will simplify the design.

7.5.1 Clear Channel Assessment

In our model situation, it would be possible to keep a camera synchronized with CU and the network but the sensor will wake up unaware of the network state and timing. The GMSK transceivers in sensors will awake from sleep and will have to find a correct channel to Tx as soon as possible. Normally, a CCA algorithm is used to determine who will transmit first and who later. The device that wants to transmit will first scan the channel, try to detect the other

transmitter and if there is one, it will back off and wait before trying again. Depending on the implementation, either the scanning time or the back-off time can be random or both. In our situation, it should also be improved by retuning to another random channel for the frequency agility.

The first thing to notice is the detection of the OFDM signal as it is spread approximately 51 times wider than the GMSK.[3] Normally, the regulations provide rules for CCA. In EU at sub-GHz, it is 15 dB above the sensitivity level [52], but the receiver's filter will detect only $1/51$ or -17 dB of the OFDM signal. Modern transceivers should have sensitivity lower than the limit required by regulations, but it needs to be accounted for in the design. On the other hand, following the rules completely would prevent the sensor and the camera from transmitting together, so a little reinterpretation is necessary.

If the situation would be a continuous stream of video, such as DVB-T for example, the sensor would have no other choice but to scan every channel and find a hole. It would have to detect a drop in the power on one of the channels. Scanning for power should take less than the already mentioned 630 µs [22] which were for preamble detection, but scanning tens of channels will still require some time. The actual time will depend on the transceiver and its configuration. Random channel scanning might also be unreliable in a multipath environment if the remaining sidebands in the hole are stronger than a used channel on the other side of the spectrum.

For this network, there is a set of `HoleOfdmGen` and `HoleOfdmRec` which use frames and can add an arbitrary hole, including to the frame header. The CCA used here is almost the opposite of the correct algorithm. The sensor starts as a regular CCA and if the channel is clear, it transmits. If not, instead of a random back-off, the sensor will wait until the OFDM packet ends and start transmitting immediately after. The OFDM transmitter needs to sense the GMSK signal as a part of its CCA and automatically eliminate the necessary tones. There is a risk of multiple sensors transmitting both after the same end of the OFDM packet, but the chance for collision is smaller by the number of randomly selected channels.

7.5.2 Simplified Sensor

The sensor made in Chapter 6 was simplified to the bare minimum. Its only function is to send packets and wait for an acknowledgment. The sensor is sleeping most of the time and waiting for an interrupt from a button. If the button is pressed, the sensor does the modified CCA, transmits and waits in Rx for 100 ms. Another switch is used to select whether to randomly select frequencies or stay on one channel.

The packet format was also simplified. The length byte was removed and the packet size was fixed on both ends. An incorrectly decoded bit in the length byte would mean an entire packet lost. The packet has 18 B with 9 copies of

[3] A very approximate amount of 51 out of 64 channels of GMSK is covered by the OFDM with 20% guardbands.

a 1 B counter, to detect packets missed from sensor to CU, and 9 copies of a 1 B number obtained by the last acknowledge, to detect missing acknowledgments from CU to sensor. CRC was removed from the packet and the majority of 9 copies is used instead to detect how many bits were received wrong.

18 B would be a very short packet when network information and encryption are added. This could represent a binary sensor as a magnetic door detector or a smoke alarm.

7.5.3 OFDM Packet Format

The individual OFDM tones were modulated with quadrature PSK which puts one byte per 4 symbols. The packet was fixed to a length of 64 B per tone or approximately 15 ms on air including all synchronization symbols. The packet length is approximately equal to a 64 B packet of GMSK including its preamble and syncword. Without the pilots and guard bands, there are 178 data tones which gives something over 11 KiB.

The OFDM link should transport video. In the production device, the video would be encoded to save much of the available datarate. Camera devices would also have to track their Tx duty cycle which means limiting the transmitted video to only short bursts when there is some activity detected or on an intentional user request. In this prototype network, the goal was to continuously fill the link up to its limit without any restrictions on bandwidth usage. The raw video was mapped to the two-dimensional OFDM packet.

Each packet starts with one row of bytes storing 64 copies of a 16-bit video frame counter and a 16-bit vertical position of the first line. The next 63 rows are used to store 7 lines of the video data. The video picture frames at 432×768 pixel were split into lines. Each line of the picture is mapped to 9 rows of the data. Three colors times three neighboring pixels are stored in the same tone. The mapping disregards pilots and holes, so a hole in frequencies translates to a hole in the picture while the rest remains in the same place.

The picture data are not protected by any FEC or CRC, to make errors visible. Only the line position in a picture frame is protected by a majority to detect when a packet gets lost.

7.5.4 Central Unit

The CU software is written in C++ and runs on a PC with LimeSDR Mini connected to one of its USB 3 ports. The development started from the software used in Chapter 6 and uses the same GMSK receiver and transmitter. Few improvements and fixes were made during this development. The SDR classes were split into a submodule to allow easy transfer of changes between this software and the one from Chapter 6.

This software uses three threads and even more are spawned by the LimeSuite library. One thread is dedicated to the user interface which is mainly a console, but it handles also an output of the received video. This thread also

handles debug facilities such as printing a log of events and exporting debug data to be plotted. Two other threads use maximal priority and process SDR and application layer behavior. The second thread handles the LimeSuite library, GMSK Tx, FFT and the first stage of GMSK Rx and stores the incoming frames to FIFO. The threads are linked with a simple statically allocated FIFO for the 1020 sample frames. The third thread uses data from this FIFO to receive OFDM packets. It also handles the application layer of communication with the sensor.

The software uses the OpenCV library. No other OpenCV tools were used besides basic data manipulation and `imshow()` to show the received picture on the screen. The library was selected for convenience as it is also used in the camera, but it could be easily replaced. The video frame is built line by line as the packets are received and a thin red line is drawn underneath to show what part of the picture was updated. Data from missing packets are cleared from the image.

A small quirk of the setup creates a difference from the symbol simulation in Section 7.2. In the simulation, the symbol of OFDM including the guard interval was the same number of samples as the GMSK symbol or its multiple. In the SDR software, both modulations need to be baseband signals at the same sampling frequency. The all-channel GMSK needs to have one symbol per 2×64 samples, exactly at the 2^x size. The OFDM is set to $n = 256$ with a $n/8 = 32$-sample cyclic prefix which cannot be equal to 2^x. This difference should have a similar effect as a realistic shift between symbols from different transmitters.

7.5.5 Wireless Camera

The camera prototype is based on a Raspberry Pi 4 with a CSI camera module. Raspberry Pi OS operating system, now in standard 64 bit, based on Debian bullseye, was used to run it. The LimeSDR Mini was connected to USB 3, an Ethernet cable was plugged in for SSH access and an LCD was connected via the DSI for convenience.

The SDR software is written in `C++` and compiled from the same sources as the software for the CU. It is compiled on the host PC with `aarch64-linux-gnu-g++` with different options enabled. Initially, there were some problems with cross-compilation and the solution was to use the same version of Debian as is used in Raspberry on the PC used for development. Having newer Debian testing was not possible as it contains a newer `glibc` used by the libraries.

This software uses a similar layout as the one in CU. One thread is for the console output, one thread handles the LimeSuite library and detects incoming noise on one of the GMSK channels and the third thread composes the OFDM packets and puts the data to FIFO to be sent.

The main SDR loop was shortened, compared with the CU software, to 4 frames. It still causes a delay of at least 1.04 ms to react to an incoming signal.

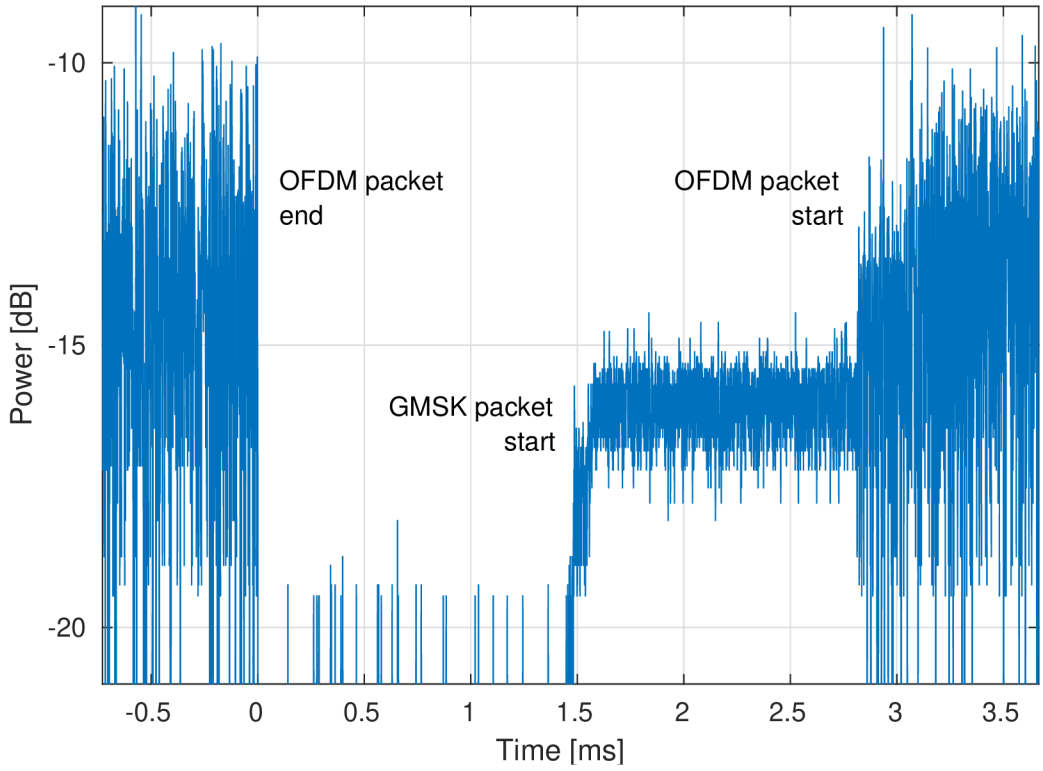

Figure 7.12: Timing of OFDM and GMSK Packets

The delay between two OFDM packets needs to include the time for the sensor to start transmitting and the reaction time of the camera. The delay is shown in Figure 7.12 which was measured by RTL-SDR at 1024 ksample/s and passed through an RMS filter in GNURadio. The power values are only for illustration as they depend on attenuators used to connect all devices and on the OFDM signal which does not fit into RTL-SDR bandwidth.

Figure 7.12 shows that the sensor takes approximately 1.5 ms to start transmitting. From personal experience with S2LP, I know that this time can be almost 5 times less, but it would require not using the S2LP library, reorganizing most of the sensor firmware and would have a little benefit in this prototype. In this situation, the delay between packets was chosen to be 13 LimeSuite frames or approximately 2.7 ms. The figure shows that the next OFDM packet starts approximately 2.75 ms after the first one ends which would be somewhere between 13 and 14 frames.

The camera data are obtained by OpenCV and a GStreamer pipeline with the advantage that the CSI camera or any other USB camera can be used almost without any changes to the code. It also rescales the picture to 432×768 pixel to fit nicely over the selected number of tones with a common aspect ratio.

7.5.6 Communication Problems

The first attempts with OFDM on the LimeSDR Mini showed that the signal is spread and the holes are barely visible. Two fixes were made to make the transmission work. Both are probably linked together and with high PAPR of the generated OFDM signal. First suspicion was to wrongly set Tx power amplifiers in the SDR hardware. The hardware has too many options to attempt custom settings and the only option is to trust the LimeSuite driver. The author of the liquid library writes [63]: "Most hardware have highly non-linear RF front ends (mixers, amplifiers, etc.) which require a transmit power back-off by a few dB to ensure linearity, particularly when many subcarriers are used." The solution was to reduce both the amplitude of the generated OFDM signal by $1/7$ and the driver Tx power by -18 dB. The Tx power could probably be increased significantly with proper tuning of the entire SDR chain. Another fix was adding a whitening of the transmitted data. Both transmitter and receiver flip each transmitted bit by the same m-sequence. With those two fixes, the OFDM transmitter and receiver started to work correctly.

The GMSK signal proved to be quite disturbing to the OFDM receiver. The signal from the sensor is problematic only slightly, which was expected from the simulation results, but the acknowledgment transmitted by the CU is problematic very. The situation could be probably improved by a proper separation between the SDR Tx and Rx paths. A circulator could be used instead of two common attenuators. Manually tuning the power of both GMSK signals was enough to make the system work. In a real situation with antennas where the separation between camera and CU will be much higher than between Rx and Tx antenna, the circulator might be necessary.

Another option would be to circumvent the problem with a clever design of the network. The video link would require a back channel from the CU to the camera. Higher OSI layers need to send acknowledge packets and CU will need a way to control the camera. The acknowledgment to the camera might be sent sooner than required and allow the CU to switch its SDR to Tx mode. The CU, as a master of the network, has the option to take priority and transmit right away. Both acknowledgments to the camera and the sensor can be sent at the same time. This would make the circulator in CU unnecessary and also allow the use of much simple SDR with a switch between Rx and Tx instead of both in parallel.

7.6 Results

One OFDM packet with the delay between two packets as seen in Figure 7.12 totals approximately 18 ms. One packet adds 7 lines of a picture which results in one frame of raw video in 1.1 s. It can also be converted to a datarate of 618 KiB/s or 5.1 Mbit/s. The same numbers were also observed by counting the received data in CU. This datarate is without holes which will remove 6 or 7×64 B from each packet, depending on pilot positions.

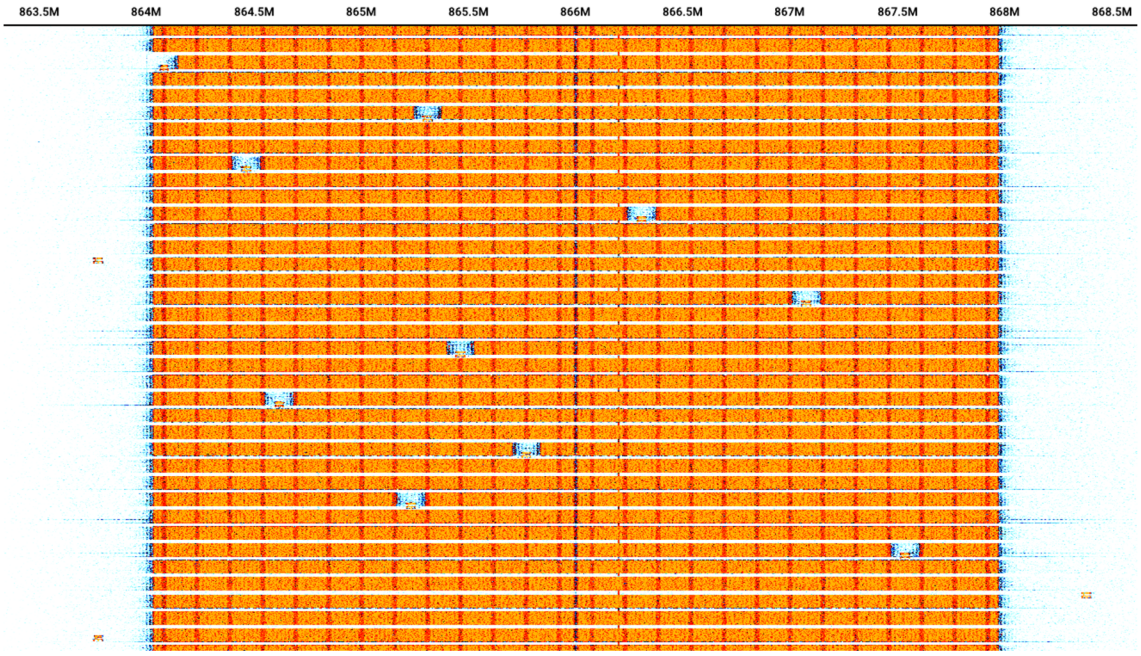

Figure 7.13: Waterfall of OFDM with Holes

Figure 7.13 was gathered by LimeSDR Mini and SDR++. Time flows up on the figure, the oldest data are at the bottom. The colors of the waterfall image were inverted and level-shifted for better illustration on white background. The center frequency of the receiver was put slightly higher than the transmitter. It can be seen as an artifact on the waterfall made by the DC spike. Other artifacts may be caused by missed samples. This setup was only for illustration purposes and not for serious measurement.

In Figure 7.13 the GMSK packet can be seen to start almost immediately after the OFDM packet ends. A short while after that, another OFDM packet starts with a hole around the continuing GMSK packet. It is the same situation as in Figure 7.12. Normally, another GMSK packet would follow as a response from CU to the sensor, but in this figure, the CU's radio was used to gather the waterfall instead. The same situation can be seen in Figure 7.14 which is a video output from the CU with sensor transmitting. Light gray are the pilot tones and darker gray are the unused tones and holes made for the sensor's signal.

Some GMSK packets are also visible on channels that are in the OFDM guard bands. The real network doesn't have to use these GMSK channels or use them only for key fobs and other mobile devices. That would circumvent the problem with configuring Tx power of mobile devices.

Figure 7.15 shows a spectrum of the OFDM signal. It was measured with an RBW of 5 kHz. Pilot tones modulated by binary PSK and the DC spike make small dimples in the spectrum. Figure 7.16 was measured with RBW of 5 kHz as well. It shows the sensor GMSK signal put inside a hole in the OFDM.

Error rates were measured with the power of both signals as set manually during the development. It is the case of GMSK power set to -23 dBm.

Figure 7.14: Video Output from CU

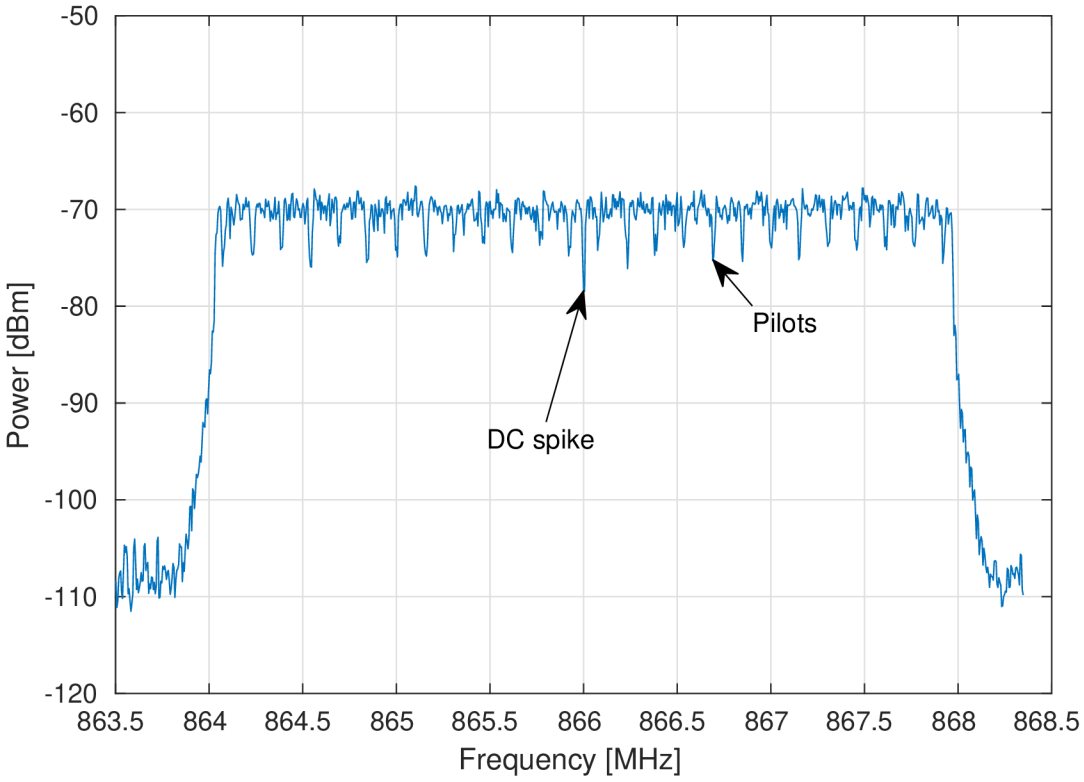

Figure 7.15: OFDM Spectrum

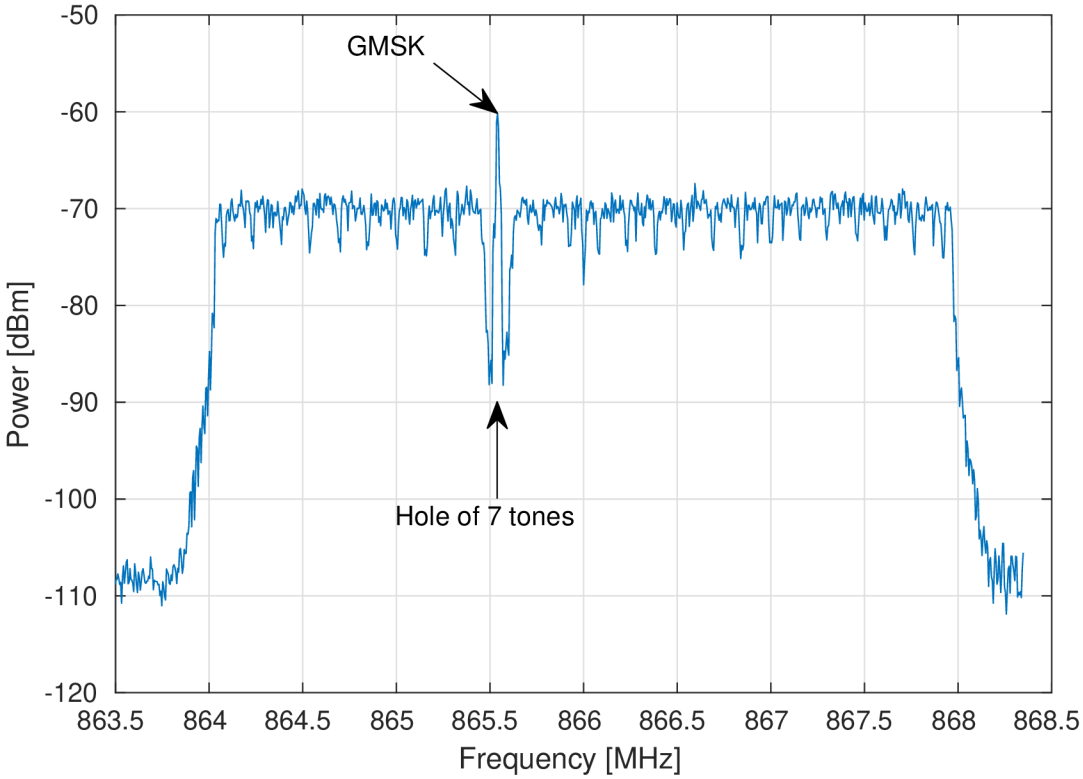

Figure 7.16: Spectrum of OFDM with Hole and GMSK

The sensor transmitted one GMSK packet approximately once on every third OFDM packet, so only $1/3$ of the OFDM was affected by the GMSK.

- When only the OFDM was running, no packet out of 32962 was lost. Out of more than 358 MiB, only 2583 bits were received wrong ($0.9 \cdot 10^{-6}$).
- When only the GMSK was running, no packet was lost out of 12549 sent. Out of almost 221 KiB, not a single bit was received wrong.
- OFDM with holes filled with GMSK has worse parameters.
  - None out of 32976 OFDM packets was lost (few usually do), and 6893 bits were wrong out of 355 MiB ($2 \cdot 10^{-6}$).
  - 2268 out of 11523 GMSK packets were lost (0.2) and $14\,778$ bits were wrong out of 163 KiB (10^{-2}).
- Without holes in the OFDM, the communication is unreliable.
  - 749 out of 32976 OFDM packets were lost ($2 \cdot 10^{-2}$) and 0.1 out of 350 MiB were wrong ($4 \cdot 10^{-4}$).
  - The GMSK communication wasn't usable at all and the CU software wasn't able to track the Tx packet counter to count lost packets.

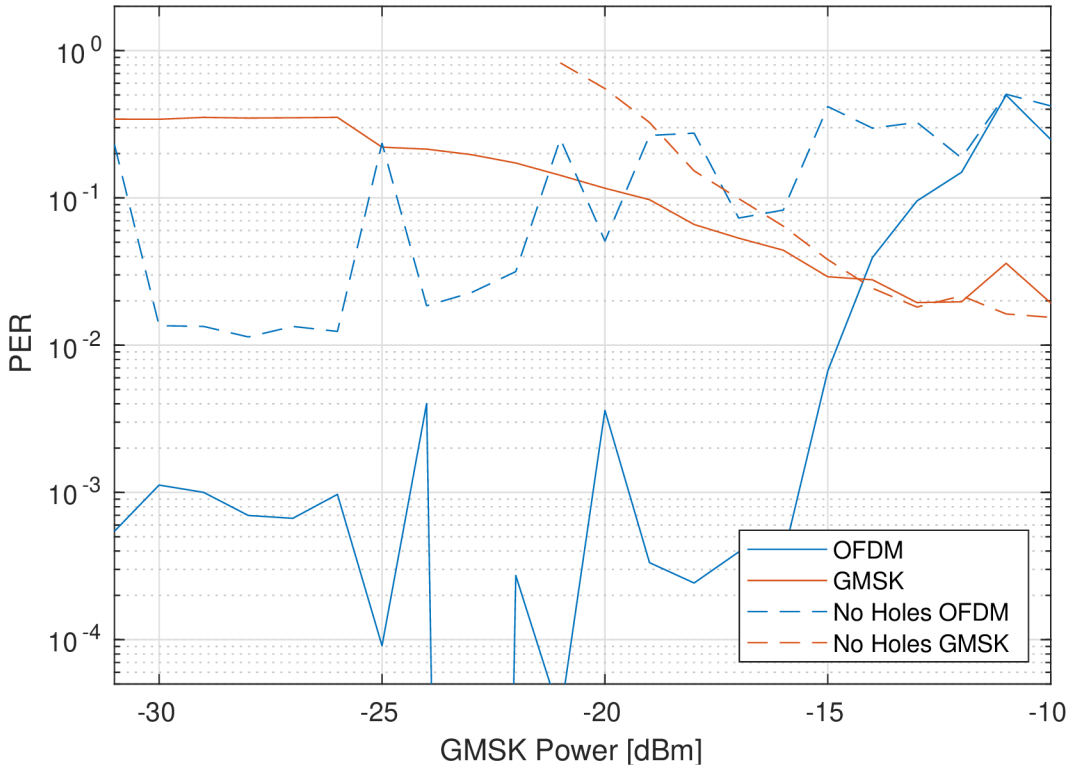

Figure 7.17: PER Test of the Prototype

These numbers show that the situation does favor the OFDM. The GMSK connection would be usable, but with drawbacks. In a low-power device, the frequent packet repetitions would increase its current consumption. The last test shows that if the OFDM link does not provide holes for the GMSK, the communication suffers a lot. Both PER and BER of the OFDM drop at least two more orders of magnitude. It should be beneficial to be polite even if the GMSK communication is not a part of the same network. European norms do not require the use of CCA if the duty cycle is limited, but waiting or making holes should be preferred. The GMSK link is not usable at all without holes.

Figures 7.17 and 7.18 show how PER and BER change when the power of the GMSK signal is varied. The drop of OFDM BER at the right edge is probably a quirk of the test as the inaccuracy of BER rises with high PER and less data received. The optimum of BER would be for GMSK power above -16 dBm, a little higher than the value manually selected during development. The PER of the GMSK signal stays unreasonable high even with increasing power.

It seems that the hole should be larger than the selected 7 tones. A hole of 11 tones would carry approximately $4/178 \approx 2\%$ less data but would improve the interference. Figure 7.19 shows how BER is influenced by the hole size when GMSK power was set to -16 dBm. The PER stayed almost constant in this situation for both OFDM and GMSK. Perhaps the bad PER of GMSK (over $4 \cdot 10^{-2}$) is not caused by an interference of the signals but by saturation of the SDR receiver.

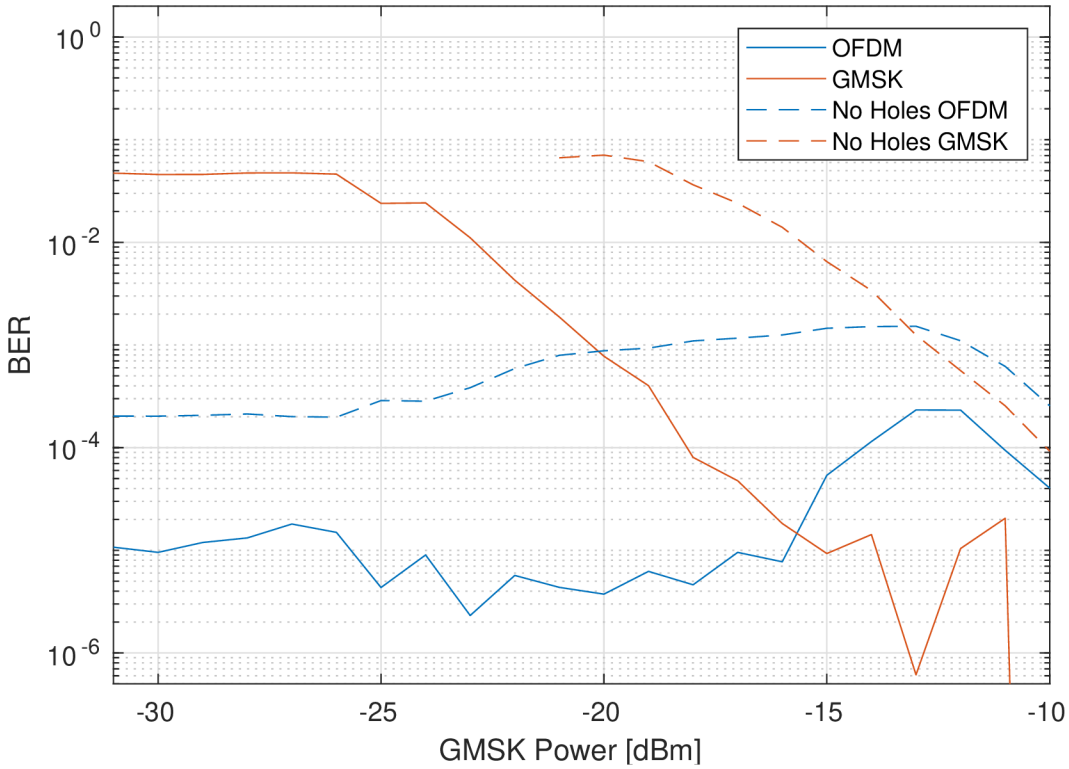

Figure 7.18: BER Test of the Prototype

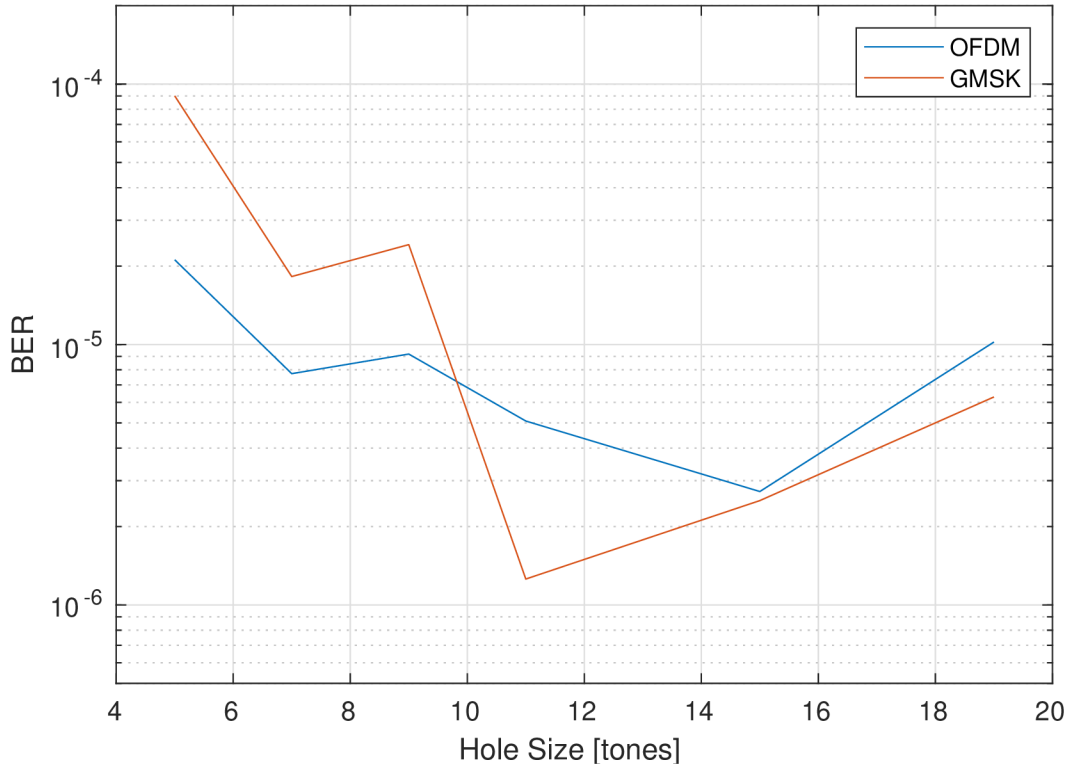

Figure 7.19: Hole Size Test of the Prototype

8 Conclusion

Current security and automation systems are often stuck on single-frequency communication which is susceptible to jamming, no matter if intentional or caused by an external signal from poorly designed electronics. The main reason for a single-frequency system is the delay from a sensor waking up to information getting to a CU. This work proposes two methods on how to improve wireless communication in home automation and security systems.

One method improves sensor communication, is affordable and could be implemented with proper investment in development. The other method adds video cameras, but wouldn't currently fit into a budget of a CU of a security or home automation system. However, it might be available in near future.

Both methods allow sensors to wake up from sleep and immediately start transmitting on a random channel. A network can be designed based on these principles. The sensor hardware, power consumption, range and communication delay can stay the same as for the simple single-frequency solution that is currently available. The main improvement is increased robustness by frequency agility. If one channel is occupied, the sensor can randomly select another. With some invention, these principles could also be used to synchronize the sensor into a slow frequency hopping without the usual disadvantages.

The second method allows coexistence between the signals of a sensor and a video camera. It allows the security camera to have only one hardware radio for both the low-power sensor network and the high datarate video link.

8.1 Affordable All-channel Receiver in Cortex-M

This receiver allows receiving all GMSK channels at the same time. The construction uses a cheap demodulator from a DVB-T tuner connected to the MCU's ADC, so the additional complexity of the CU is acceptable for home automation or security.

The prototype hardware is not optimized and its analog design could use a lot of attention from an RF engineer. The final prototype had only a range of several meters. A receiver of a similar structure running on PC with RTL-SDR with a proper antenna had a range comparable to or greater than a standard GMSK receiver. That might indicate that there is nothing fundamentally wrong with the concept, just an imperfect design of the Rx chain.

Despite the prototype parameters, it demonstrates the possibilities of software radio in small embedded microprocessors. Apart from the direct application, the design explores a new interesting area of embedded electronics. A simple SDR receiver doesn't need an expensive FPGA to work. This potentially opens a whole new market of consumer electronics to innovative and experimental designs in RF communication, radars or similar. A receiver without the need for FFT would fit into the MCU easily. For more complicated designs, even faster Cortex-M microprocessors are already available.

8.2 Mixed Network with GMSK and OFDM

This set of OFDM transmitter and receiver allows making holes several frequency tones wide. These holes can be used by the GMSK communication. The OFDM receiver will automatically detect the hole layout. It could be useful for a home security system with wireless cameras. In this case, the previous all-channel GMSK receiver can be run on the same hardware in parallel with the OFDM. The results for the GMSK receiver while the OFDM is running are not good even when power and hole size are varied. More research might be needed to get the PER reasonably low.

The implementation presented here is only one of many possibilities of the entire network design. For example, a much simpler solution for the cohabitation would be a time division multiplex between the two modulations. In this specific setup, it would reduce the OFDM datarate to $2/3$. It would be the preferred solution if the sensor packets would be sparse enough. However, if the expected situation is multiple sensors reporting at the same time as the video cameras start transmitting, then the holes can be a significant improvement.

Currently, common SDRs available on the market are too expensive for consumer electronics, but the price could be pushed way down with proprietary chips used in the mobile phone segment. Security cameras will need both a processor to encode the video stream and an SDR for communication. Both components are cheaply manufactured for every mobile phone but unfortunately kept secret to protect the design. The situation could change rapidly in the same way as embedded computing became easily available by Raspberry Pi and SDR became available by the leaked datasheet of RT820T2 and RTL-SDR.

8.3 Applications and More Research

Still, a lot of development would be needed between this work and a production device. The development of a real network should start with up-to-date hardware. There are new options instead of the LPC4370 as more powerful MCUs became available during the studies. For example, STM32H7 offers up to 480 MHz Cortex-M7 with M4 as a coprocessor. A complete CU could instead use something like an STM32MP1 with embedded Linux and radio running on

an M4 coprocessor. Both of these M4 coprocessors are comparable to the main core of the LPC4370.

Instead of the RT820T2 demodulator, there are also other options. With the upcoming IEEE 802.15.4 OFDM, more SDR-capable modems are expected. They can be used as a cheaper alternative between LimeSDR Mini and RTL-SDR. Recently, the CaribouLite[1] was crowdfunded to connect the raw IQ stream directly to Raspberry Pi's memory interface. It will be a nice and affordable concept for more SDRs which could be transformed into a CU. The IEEE 802.15.4 chips can already work with OFDM on their own which would allow time-multiplexed systems with cameras and sensors with relatively little computing power required.

An experiment showed that making the hole is advantageous in sense of correctly received OFDM packets. Even if the hole mechanism shouldn't be used for an own sensor network, it might be useful to avoid interference such as a narrow bandwidth LPWAN device. The device (eg. Sigfox) can use more than a second long, very narrow packets for its communication. Instead of waiting for the IoT device to end, it would be beneficial to eliminate a few tones and communicate. This might become even more important if the sub-GHz frequencies fill up with both LPWAN and IEEE 802.15.4 OFDM devices.

The OFDM prototype was able to communicate with $5.1\,\mathrm{Mbit/s}$. Theoretical datarate without spaces between packets would be closer to $6\,\mathrm{Mbit/s}$ which, given the bandwidth and quadrature PSK, is on par with the oldest IEEE 802.11a. Today, much higher speeds are expected. A lot of improvements could be obtained by dynamically assigning modulation to tones, simpler binary PSK for tones next to the GMSK signal and more complex Quadrature Amplitude Modulation (QAM) further away.

One path of future research would be the principles of OFDM-MFSK. The communication in the direction from the CU to the sensor should not be hard. Some tones would use regular PSK or QAM and some intended for the sensor would use two or four-symbol FSK. It would allow reducing the hole as the FSK signal would be orthogonal to the rest of the OFDM. In the direction from the sensor to the CU, it would be more difficult to synchronize the two. Perhaps an OFDM-MFSK preamble could encode the hole position and the sensor could precisely fit in. The rest of the OFDM packet could continue with regular modulation. It might again work only for stationary devices and many issues common with OFDMA synchronization would probably arise.

An important aspect of these methods would be the design of the entire network. The CU radio would be much simpler if it didn't have to transmit and receive at the same time. Timing of the OSI transport layer could ensure that the CU has enough time to compose a response to multiple sensors and the camera into one OFDM-MFSK packet. It would have to be balanced with the maximal delay allowed before a repeated packet gets to CU.

[1] Funded at crowdsupply.com/cariboulabs/cariboulite-rpi-hat.

Bibliography

[1] M. Starsinic, "System architecture challenges in the home M2M network," in *2010 IEEE Long Island Systems, Applications and Technology Conference*, 2010. DOI: 10.1109/LISAT.2010.5478336.

[2] V. K. Huang, Z. Pang, C.-J. (Chen, and K. F. Tsang, "New Trends in the Practical Deployment of Industrial Wireless," *IEEE Industrial Electronics Magazine*, 2018. DOI: 10.1109/MIE.2018.2825480.

[3] C. A. Trasviña-Moreno, Á. Asensio, R. Casas, R. Blasco, and Á. Marco, "WiFi Sensor Networks: A study of energy consumption," in *2014 IEEE 11th International Multi-Conference on Systems, Signals and Devices (SSD14)*, 2014. DOI: 10.1109/SSD.2014.6808887.

[4] P. S. Cheong, J. Bergs, C. Hawinkel, and J. Famaey, "Comparison of LoRaWAN classes and their power consumption," in *2017 IEEE Symposium on Communications and Vehicular Technology (SCVT)*, 2017. DOI: 10.1109/SCVT.2017.8240313.

[5] J. So and Y. Kim, "Interference-aware frequency hopping for Bluetooth in crowded Wi-Fi networks," *Electronics Letters*, 2016. DOI: 10.1049/el.2016.1773.

[6] D. Newell and M. Duffy, "Review of power conversion and energy management for low-power, low-voltage energy harvesting powered wireless sensors," *IEEE Transactions on Power Electronics*, 2019. DOI: 10.1109/TPEL.2019.2894465.

[7] N. H. Motlagh, "Frequency Hopping Spread Spectrum: An Effective Way to Improve Wireless Communication Performance," *Advanced Trends in Wireless Communications*, 2011. DOI: 10.5772/15482.

[8] R. Piyare, A. L. Murphy, C. Kiraly, P. Tosato, and D. Brunelli, "Ultra low power wake-up radios: A hardware and networking survey," *IEEE Communications Surveys Tutorials*, 2017. DOI: 10.1109/COMST.2017.2728092.

[9] R. Rondón, M. Gidlund, and K. Landernäs, "Evaluating Bluetooth Low Energy Suitability for Time-Critical Industrial IoT Applications," *International Journal of Wireless Information Networks*, 2017. DOI: 10.1007/s10776-017-0357-0.

[10] P. H. Kindt, M. Saur, M. Balszun, and S. Chakraborty, "Neighbor Discovery Latency in BLE-Like Protocols," *IEEE Transactions on Mobile Computing*, 2018. DOI: 10.1109/TMC.2017.2737008.

[11] K. Mikhaylov, "Accelerated Connection Establishment (ACE) Mechanism for Bluetooth Low Energy," in *2014 IEEE 25th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2014. DOI: 10.1109/PIMRC.2014.7136362.

[12] R. Liu, A. Beevi K.T., R. Dorrance, *et al.*, "An 802.11ba-based wake-up radio receiver with wi-fi transceiver integration," *IEEE Journal of Solid-State Circuits*, 2020. DOI: 10.1109/JSSC.2019.2957651.

[13] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario," in *2013 IEEE International Wireless Symposium (IWS)*, 2013. DOI: 10.1109/IEEE-IWS.2013.6616827.

[14] L. Flueratoru, S. Wehrli, M. Magno, E. S. Lohan, and D. Niculescu, "High-accuracy ranging and localization with ultrawideband communications for energy-constrained devices," *IEEE Internet of Things Journal*, 2022. DOI: 10.1109/JIOT.2021.3125256.

[15] C. Yao, Y. Liu, X. Wei, G. Wang, and F. Gao, "Backscatter technologies and the future of internet of things: Challenges and opportunities," *Intelligent and Converged Networks*, 2020. DOI: 10.23919/ICN.2020.0013.

[16] A. S. Arezoomand and M. Pourmina, "Prolonging network operation lifetime with new maximum battery capacity routing in wireless mesh network," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010. DOI: 10.1109/ICCAE.2010.5451679.

[17] J. Ding, M. Nemati, C. Ranaweera, and J. Choi, "Iot connectivity technologies and applications: A survey," *IEEE Access*, 2020. DOI: 10.1109/ACCESS.2020.2985932.

[18] G. Boquet, P. Tuset-Peiró, F. Adelantado, T. Watteyne, and X. Vilajosana, "Lr-fhss: Overview and performance analysis," *IEEE Communications Magazine*, 2021. DOI: 10.1109/MCOM.001.2000627.

[19] C. Gomez, J. Carlos Veras, R. Vidal, L. Casals, and J. Paradells, "A Sigfox Energy Consumption Model," *Sensors*, 2019. DOI: 10.3390/s19030681.

[20] R. Mozny, P. Masek, M. Stusek, K. Zeman, A. Ometov, and J. Hosek, "On the performance of narrow-band internet of things (nb-iot) for delay-tolerant services," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019. DOI: 10.1109/TSP.2019.8768871.

[21] H. Chougrani, S. Kisseleff, W. A. Martins, and S. Chatzinotas, "Nb-iot random access for non-terrestrial networks: Preamble detection and uplink synchronization," *IEEE Internet of Things Journal*, 2021. DOI: 10.1109/JIOT.2021.3123376.

[22] T. Jakubík and J. Jeníček, "Asymmetric Low-power FHSS Algorithm," in *Proceedings of the IEEE 13th International Workshop On Electronics, Control, Measurement, Signals and their Application in Mechatronics*, 2017. DOI: 10.1109/ECMSM.2017.7945892.

[23] T. Jakubík and STMicroelectronics, "Cortex-M Simulator," in *2020 International Conference on Applied Electronics (AE)*, 2020. DOI: 10.23919/AE49394.2020.9232712.

[24] T. Jakubík and J. Jeníček, "SDR All-channels Receiver for FHSS Sensor Network," in *2018 International Conference on Applied Electronics (AE)*, 2018. DOI: 10.23919/AE.2018.8501460.

[25] T. Jakubík and J. Jeníček, "SDR All-channels Receiver for FHSS Sensor Network in Cortex-M," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019. DOI: 10.1109/TSP.2019.8769064.

[26] M. A. Wickert and M. R. Lovejoy, "Hands-on Software Defined Radio Experiments with the Low-cost RTL-SDR Dongle," in *2015 IEEE Signal Processing and Signal Processing Education Workshop*, 2015. DOI: 10.1109/DSP-SPE.2015.7369529.

[27] M. Rice, *Digital Communications: A Discrete-Time Approach*. Pearson Education, Inc., 2009, ISBN: 9780130304971.

[28] M.-R. Awan and P. Koch, "Combined Matched Filter and Arbitrary Interpolator for Symbol Timing Synchronization in SDR Receivers," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, 2010. DOI: 10.1109/DDECS.2010.5491797.

[29] S. Rajan, S. Wang, R. Inkol, and A. Joyal, "Efficient approximations for the arctangent function," *IEEE Signal Processing Magazine*, 2006. DOI: 10.1109/MSP.2006.1628884.

[30] P. P. Ann and R. Jose, "Comparison of papr reduction techniques in ofdm systems," in *2016 International Conference on Communication and Electronics Systems (ICCES)*, 2016. DOI: 10.1109/CESYS.2016.7889995.

[31] M. Wetz, I. Periša, W. G. Teich, and J. Lindner, "Robust Transmission Over Fast Fading Channels on the Basis of OFDM-MFSK," *Wireless Personal Communications*, 2008. DOI: 10.1007/s11277-007-9395-8.

[32] T. Jakubík and J. Jeníček, "Feasibility of OFDM and FHSS in a Single Home Automation and Security Network," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020. DOI: 10.1109/TSP49548.2020.9163493.

[33] M. Morelli, M.-O. Pun, and C.-C. J. Kuo, "Synchronization Techniques for Orthogonal Frequency Division Multiple Access (OFDMA): A Tutorial Review," in *Proceedings of the IEEE | Vol. 95, No. 7*, 2007. DOI: 10.1109/JPROC.2007.897979.

[34] G. Al-Juboori, A. Doufexi, and A. R. Nix, "Feasibility study of ofdm-mfsk modulation scheme for smart metering technology," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017. DOI: 10.1109/ISGTEurope.2017.8260200.

[35] H. Zhou, A. V. Malipatil, and Y.-F. Huang, "Synchronization issues in ofdm systems," in *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, 2006. DOI: 10.1109/APCCAS.2006.342228.

Manuals, Datasheets and Online

[36] S. Kamkar. "OpenSesame." (2015), [Online]. Available: https://samy.pl/opensesame/.

[37] "JA-160PC Wireless PIR motion detector combined with a camera," Jablotron Alarms a.s. (2018), [Online]. Available: https://www.jablotron.com/en/produkt/wireless-pir-motion-detector-combined-with-a-camera-296/.

[38] "JA-151M Mini wireless magnetic detector," Jablotron Alarms a.s. (2018), [Online]. Available: https://www.jablotron.com/en/produkt/mini-wireless-magnetic-detector-213/.

[39] *EN 50131-1*, CENELEC, 2006.

[40] *SX1232 868 and 915MHz Ultra Low Power High Link Budget Integrated UHF Transceiver*, Semtech Corporation, 2013.

[41] *CC1200 Low-Power, High-Performance RF Transceiver*, Texas Instruments Incorporated, 2014.

[42] *S2LP Ultra-low power, high performance, sub-1 GHz transceiver*, STMicroelectronics NV, 2019.

[43] *LoRaWAN 1.0.4 Specification*, LoRa Alliance, Inc., 2020.

[44] *IEEE Standard for Low-Rate Wireless Networks*, IEEE 802.15.4-2020, IEEE Computer Society, 2020.

[45] *CFR Title 47, §15.247*, FCC, 2003.

[46] *Atmel AT86RF215 Device Family*, Atmel Corporation, 2016.

[47] *ATA8352 Impulse-Radio Ultra-Wideband (IR-UWB) Transceiver*, Microchip Technology Inc, 2021.

[48] *DW3000 Ultra Wideband Transceiver*, Qorvo Inc, 2021.

[49] *SX1302 LoRa Gateway Baseband Transceiver*, Semtech Corporation, 2019.

[50] *TS 103 357*, ETSI, 2018.

[51] *nRF9160 Product Specification v2.1*, Nordic Semiconductor ASA, 2021.

[52] *EN 300 220-2*, ETSI, 2017.

[53] A. Collins, *All Programmable RF-Sampling Solutions*, Xilinx, 2017.

[54] T. Leconte. "Playing with the Airspy R820T IF bandwidth." (2018), [Online]. Available: https://tleconte.github.io/R820T/r820IF.html.

[55] *LPC4370 Datasheet*, Rev. 2.3, NXP B.V., 2016.

[56] RTL-SDR Blog. "R820T2 Chip Discontinued: Low Cost R820T2 RTL-SDRs will Continue, Airspy Will Redesign." (2018), [Online]. Available: https://www.rtl-sdr.com/r820t2-chip-discontinued-low-cost-r820t2-rtl-sdrs-will-continue-airspy-will-redesign/.

[57] RTL-SDR Blog. "The R860 Will Replace The R820T2 - Same Chip Different Name." (2021), [Online]. Available: https://www.rtl-sdr.com/the-r860-will-replace-the-r820t2-same-chip-different-name/.

[58] *LPC4370 User Manual, UM10503*, Rev. 2.3, NXP B.V., 2017.

[59] *DT0087 Coordinate rotation digital computer algorithm (CORDIC) test and performance verification*, STMicroelectronics, 2017.

[60] *STM32L062K8 STM32L062T8 STM32L062C8 Ultra-low-power 32-bit MCU Arm-based Cortex-M0+*, STMicroelectronics NV, 2020.

[61] *LMP91000 Sensor AFE System*, Texas Instruments Incorporated, 2014.

[62] *IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE 802.11-2020, IEEE Computer Society, 2020.

[63] Joseph D. Gaeddert. "Ofdm flexible framing structure (ofdmflexframe)." [Online; accessed May 12, 2022]. (2022), [Online]. Available: https://liquidsdr.org/doc/ofdmflexframe/.

List of Published Results

[22] T. Jakubík and J. Jeníček, "Asymmetric Low-power FHSS Algorithm," in *Proceedings of the IEEE 13th International Workshop On Electronics, Control, Measurement, Signals and their Application in Mechatronics*, 2017. DOI: 10.1109/ECMSM.2017.7945892.

[23] T. Jakubík and STMicroelectronics, "Cortex-M Simulator," in *2020 International Conference on Applied Electronics (AE)*, 2020. DOI: 10.23919/AE49394.2020.9232712.

[24] T. Jakubík and J. Jeníček, "SDR All-channels Receiver for FHSS Sensor Network," in *2018 International Conference on Applied Electronics (AE)*, 2018. DOI: 10.23919/AE.2018.8501460.

[25] T. Jakubík and J. Jeníček, "SDR All-channels Receiver for FHSS Sensor Network in Cortex-M," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, 2019. DOI: 10.1109/TSP.2019.8769064.

[32] T. Jakubík and J. Jeníček, "Feasibility of OFDM and FHSS in a Single Home Automation and Security Network," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020. DOI: 10.1109/TSP49548.2020.9163493.

[64] T. Jakubík and J. Jeníček, "OFDM and FHSS Hybrid Network," in *Počítačové architektúry & diagnostika PAD 2017*, 2017, ISBN: 978-80-972784-0-3.

[65] T. Jakubík and J. Jeníček, "Mnohokanálový softwarový FHSS přijímač na Cortex-M," in *Počítačové architektury a diagnostika PAD 2018*, 2018, ISBN: 978-80-261-0814-6.

[66] T. Jakubík and J. Jeníček, "Asymmetric Low-power FHSS Algorithm," in 5^{th} *Prague Embedded Systems Workshop*, 2017, ISBN: 978-80-01-06178-7.

Software Created for this Work

All software here should be obtainable from GitLab, but its long-term availability cannot be guaranteed. Use any of this code only at your own risk and be sure to check local radio regulations first.

Matlab All-channel Receiver

These Matlab scripts and Simulink models were used to test the all-channel receiver in Chapter 5. Some are modified to work with RTL-SDR and some are intended for a simulation.

- gitlab.com/ratiafak/allch_sim

Cortex-M All-channel Receiver

This is a firmware for LPC4370 with the all-channel receiver used in Chapter 5. It uses fast ADC to receive GMSK packets demodulated by R820T2. These three projects of MCUXpresso also need CMSIS libraries from NXP.

- gitlab.com/ratiafak/allch1
- gitlab.com/ratiafak/allch1_app
- gitlab.com/ratiafak/allch1_intercom

Matlab Simulation of OFDM and GMSK

This is a set of Matlab scripts and Simulink models to simulate influence of OFDM and GMSK. They were used in Chapter 7.

- gitlab.com/ratiafak/ofdm_gmsk_ber_sim

Gas Sensor Firmware

This firmware for STM32L062 was used in the gas sensor in Chapter 6. It was developed using CubeIDE. Version `S01.00.0012` was used for the current consumption measurement in Chapter 6. For Chapter 7 a lot simplified version `S01.1m.0001` was used.

- gitlab.com/ratiafak/gas_sensor

It requires a submodule for controlling the S2LP transceiver. This submodule contains `S2LP_Library` provided by ST in STSW-S2LP-DK and a high level wrapper.

- gitlab.com/ratiafak/gas_sensor_radio

CU for Gas Sensor

This is a C++ Eclipse project for Debian Linux to work with LimeSDR Mini. It communicates with the sensor and can set its parameters. It was used in Chapter 6.

- gitlab.com/ratiafak/lime_response

It requires three submodules: transceiver library `gas_sensor_radio` is included for common packet format, submodule `lime_radio` contains most SDR code and `responder_userface` contains a console user interface to control the settings of the sensors.

- gitlab.com/ratiafak/gas_sensor_radio
- gitlab.com/ratiafak/lime_radio
- gitlab.com/ratiafak/responder_userface

CU and Camera for Mixed Network

This is a C++ Eclipse project for Debian or Raspberry Pi OS Linuxen. Both versions are used in Chapter 7. It works with LimeSDR Mini. A `Release` build configuration runs on PC and is for the CU. It shows the received video and simplified sensor packets. The second build configuration, `ReleasePi`, runs on ARM and is for the camera. It transmits video from a CSI camera and makes a hole if it detects a sensor packet. This project also contains a C++ simulation of the OFDM transmitter and receiver in folder `./experiments`.

- gitlab.com/ratiafak/lime_combi

It was cloned from `lime_response` and requires the same submodules. Technically, both `responder_userface` and `gas_sensor_radio` became unnecessary with simplifications of the sensor and communication. Both could be removed together with all code that was specific for gas sensor.

Source Code for this Document

The following are sources to make this dissertation and the shorter summary.

- gitlab.com/ratiafak/dissertation

---

v12; 2022-09-16T06:54:05+02:00; a3cadc7eab725b4e6f1816342a055567d78f2a1c

Tomáš Jakubík

europass

WORK EXPERIENCE

APR 2019 – OCT 2019

INTERNSHIP, CORTEX-M FIRMWARE DEVELOPER – STMICROELECTRONICS ROUSSET SAS

Sophia Antipolis, France

JAN 2017 – CURRENT

PART-TIME, CORTEX-M FIRMWARE DEVELOPER – JABLOTRON ALARMS A.S.

Jablonec nad Nisou, Czechia

EDUCATION AND TRAINING

DEC 2016 – CURRENT – Liberec, Czechia

DOCTORAL STUDIES OF TECHNICAL CYBERNETICS – Technical University of Liberec

ISCED 8

SEP 2014 – FEB 2017 – Liberec, Czechia

MASTER'S DEGREE IN MECHATRONICS – Technical University of Liberec

ISCED 7

SEP 2015 – SEP 2016 – Zittau, Germany

MASTER'S DEGREE IN MECHATRONICS – Zittau/Görlitz University of Applied Sciences

ISCED 7

LANGUAGE SKILLS

Mother tongue(s): **CZECH**

Other language(s):

	UNDERSTANDING		SPEAKING		WRITING
	Listening	Reading	Spoken production	Spoken interaction	
ENGLISH	C2	C2	C1	B2	C1
FRENCH	A1	A2	A1	A1	A1

Levels: A1 and A2: Basic user; B1 and B2: Independent user; C1 and C2: Proficient user