University of Applied Sciences - Wismar

Faculty of Engineering
Department of Electrical Engineering and Computer Science

Master - Thesis

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Engineering (M.Eng.)

Course of Graduation:	Information and Electrical Engineering
Specialisation:	Communication Engineering and Information Technology
Topic:	Robust Software-Defined-Radar Cluster applied to the MMARIA approach.
Author:	B.Eng. Nico Pfeffer Mat.Nr. 118427
1. Supervisor:	Prof. Dr.-Ing. Ernst Jonas
2. Supervisor:	Prof. Dr. Jorge L. Chau
Date of Submission:	July 17, 2017

Statement of Originality

I, **Nico Pfeffer**, declare that I have authored this thesis independently, that I have not used other than the declared sources/resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources. This work has not previously been submitted for a degree or diploma in any university.

Signature:

Contents

List of Figures

List of Abbreviations

ACQ	Acquisition
ADC	Analog Digital Converter
ADJ	Adjust
ANA	Analyze
API	Application Programming Interface
ASP	Analog Signal Processing
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
BUF	Buffer
CFG	Configuration
CIC	Cascaded Integrator Comb
CORDIC	Coordinate Rotation Digital Computer
CPU	Central Processing Unit
CW	Continuous Wave
DAC	Digital Analog Converter
DDC	Digital Down Conversion
DFT	Discrete Fourier Transform
DNS	Domain Name System
DSP	Digital Signal Processing
DUC	Digital Up Conversion
EMC	Electro Magnetic Compatibility
ETH	Ethernet
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GNU	GNU's not Unix
GPS	Global Positioning System
GRC	GNU-Radio Companion
GUI	Graphical User Interface
GZIP	GNU ZIP
HDD	Hard Disk Drive
HDF	Hierarchical Data Format
HF	High Frequency
IAP	Institut für Atmosphärenphysik
IBP	Integer Band Position
IFFT	Inverse Fast Fourier Transform
INITD	Initializing Daemon
IP	Internet Protocol
IPP	Inter Pulse Period

JSON	JavaScript Object Notation
LAN	Local Area Network
LHB	Large Halfband Filter
LNA	Low Noise Amplifier
LOG	Logging
LTE	Long Term Evolution
LTI	Linear Time Invariant
LTS	Long Time Stable
NCO	Numerical Controlled Oscillator
NFS	Network File System
NPG	Noise Power Gain
NTP	Network Time Protocol
NTPD	Network Time Protocol Daemon
NUMPY	Numerical Python
MIT	Massachusetts Institute of Technology
MMARIA	Multistatic and Multifrequency Agile Radar for Investigations of the Atmosphere
MNG	Management
MOSH	Mobile Shell
OSI	Open System Interconnection
PA	Power-Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
PID	Process Identifier
PPS	Pulse Per Second
PRF	Pulse Repetition Frequency
PSD	Power Spectral Density
RADAR	Radio Detection and Ranging
RAM	Random Access Memory
RSYNC	Remote Synchronization
RTI	Range Time Intensity
RX	Receiver
SanDRA	Software Defined Radio in Atmospheric Research
SCIPY	Scientifical Python
SDR	Software Defined Radio
SEQ	Sequence
SHB	Small Halfband Filter
SIMD	Single Instruction Multiple Data
SNR	Signal to Noise Ratio
SSH	Secure Shell
SSHFS	Secure Shell File System
SYSTEMD	System Daemon
TPA	Third Party Access
TSIP	Trimble Serial Interface Protocol
TX	Transmitter

UDP	User Datagram Protocol
UFTP	UDP based File Transfer Protocol
UHD	USRP Hardware Driver
USRP	Universal Software Radio Peripheral
UTC	Universal Time, Coordinated
VPN	Virtual Private Network

1 Introduction

1.1 Brief History

SanDRA Since 2013, the Leibniz-Institute of Atmopsheric Physics (IAP) focused on radar system architectures with commercial software-defined-radio (SDR) hardware. This hardware is capable to use as a transmitter or a receiver. To proof the usage of SDR's in radar applications and form a statutory framework in sense of radio-frequency (RF) communications, a bachelor-thesis was written based on this topics ([27]). During this working period, the first meteor observation were made with this prototype system. This leads to another topic of special radar signal detection and identification techniques to extract signals of interest out of acquired raw data and this was in the focus of a second bachelor-thesis ([5]). Therefore an acronym called *SanDRA* (Software Defined Radar in Atmospheric Research) was defined.

MMARIA-Germany In a parallel working group, the extension of spatially measurement observations within a multistatic radar network was in focus. This concept of a multistatic radar network is the basis of the IAP called *MMARIA* (Multistatic and Multifrequency Agile Radar for Investigations of the Atmosphere). The measurement concept and fundamental ideas are described in different papers. In 2015, an article ([4]) was published that describes the wind estimation from radar measurements in a bistatic configuration with a system that uses the pulse-compression technique realized with barker codes of length 7 and also in comparison with a mono pulse configuration.

MMARIA-CW One problem with the pulse compression technique is the fact of using them in a multi-static network configuration on the same frequency. If using the same pulse compression sequence on multiple transmitters around a receiver with the same frequency the detections could not be classified from which transmitter the sequence was originally send, this results in ambiguity. To find a way out of this controversy, in 2015 IAP started a field campaign with Juha Vierinen from MIT (Massachusetts Institute of Technology) to test another measurement concept called *phase-coded continuous-wave* radar. This field campaign was successful executed in a bistatic radar configuration and another paper was published ([42]).

In the same year IAP advertised a topic for a master-thesis ([15]) to proof the concept of an extended multi-static radar network with low-power phase-coded continuous-wave radar measurements. This network was build up with three forward scatter links on the same frequency, realized with 3 transmitters and one receiver. This yields a lot of proofed fundamentals and ideas to start working on a design for a multi-static radar network and this project is named *MMARIA-CW*, which is part of *MMARIA*.

Based on the results of the field campaigns, plans for a multi-static radar network are made in collaboration with *MIT* and the *University of Tromsø*. The first network plan is, to built up 6 transmitters and 2 receivers in northern germany. This plan is illustrated in Fig.1 and shows the transmitter with red marked pins and the receivers with blue marked pins. In Fig.2, more details of the geographically positions are listed.

Figure 1: Planned Radar Network Topology

Location	Latitude / N	Longitude / E	Distance / km	Distance / km
Sehestedt	54.369001	9.817718	128,03	240,04
Goehl	54.283968	10.941189	54,51	172,20
Crivitz	53.573838	11.651366	61,11	112,59
Neudorf	53.170551	12.069858	113,40	63,55
Grammow	54.047276	12.630714	58,20	81,12
Siedenbollentin	53.733170	13.378839	115,89	46,07
Juliusruh	54.620717	13.371855	118,13	141,26
Kühlungsborn	54.146851	11.742003		
Neustrelitz	53.360115	13.073020		

Figure 2: Planned Radar Network Topology Parameter

Taking into account that the installation of this additional transmitters is combined with adminstrative and technical conditions, a mobile transmitter prototype is designed to test the new developed software and receiver acquisition. This test transmitter is shown in Fig.3, where the electronics are installed in a metal box and interfacing with the gloabl-positioning-system (GPS) antenna and the transmitter antenna at the side panel.

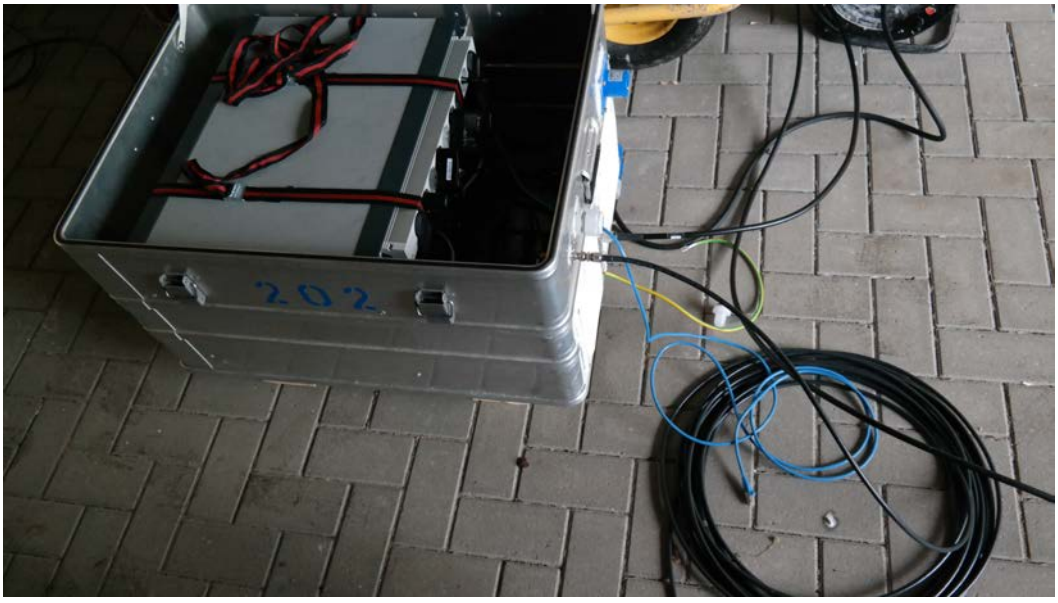


Figure 3: Prototype Mobile Transmitter Box

1.2 Motivation

Multiple prototype systems were installed to observe different atmospheric signals and analyzing them. During this working period the experience in sense of system behaviour and error-proneness grows and gives a lot of information to develop a strategy to design a system architecture, which is more robust, much easier to configure and accessible. During the development period many technical issues arised with the system stability. This is not a problem during field campaigns where technical engineers and scientists are monitoring the system behaviour around the clock at the sites. To realize a geographically distributed multi-static radar network with many transmitters and receivers, this easy monitoring by staff members becomes a big problem which will end up in circular trips for maintenance. To make this more robust and reliable a lot of attention for error handling during the measurement system runtime is in focus.

Based on the MMARIA-CW project, another main task is the signal processing on the same center frequency acquired raw data. Problems arised by trying to merge the different software parts, like unknown error messages in background executed scripts or interruption of raw data distribution with the file-transfer-protocol (FTP) within a local-area-network (LAN). This issues needs to be solved to be able to process the data with different transmitter sequence parameters in a multi-static radar network on the receiver side.

1.3 Interrogation

To solve the issues of the system behaviour, other concepts and principles needs to be designed and implemented. A possible solution is to apply more detailed software engineering techniques and a distributed computing model based on a client-server workflow. All the informations and advanced knowledge collected over the last two years impacts the design and implementation decisions in this thesis.

Since using Linux operating system for acqusition and signal processing, a deeper understanding of process management and the concepts behind them needs to be appropriated. After a lot of literature recherche, the focus is to implement system daemons. These daemons are controlled by a special software tool, called *systemd* ([57]). The systemd executes software scripts and observes the execution state. It also provides the possibility to use own developed software to start them automated at the boot intialization levels of the operating system. The configuration is intuitive and well documented by the developers.

Another implementation decision is the fact of processing multiple times the same raw data with different parameters. By the finite count of central-processing-unit (CPU) cores on a computer, the ressources gets quickly limited and therefore additional computers needs to be used. This is a combination of distributed computers in a LAN and *multi-casting* concepts which are managed up to open-system-interconnection (OSI) Layer-3 ([12]). The multi-casting technique is bandwidth efficient and distributes the same raw data in a LAN to specific computers only once at a time. The research for such implemented software tools leads to the user-datagram-protocol (UDP) based FTP (UFTP) software, developed by Denis Bush ([3]). This software applies file based transfer over network devices with an additional dropped package error handling, called *negative acknowledgement*. These additional features will be implemented in the current radar software development.

2 Theory and Basics

2.1 Operating System

For the current development of the radar stations, a decision of the used operating system is forced. This is the reference for the radar system installation, because the current development of the software is not operating system independent and version independent. The used operating system is:

- **Xubuntu 16.04.1 LTS**

The reason why using this operating system ([20]) is the capability to use an optimized desktop environment (XFCE) to be able to quickly handle some figures and operating system configurations in a *X11* based graphical environment. This avoids to do all the configurations by terminal and pure script manipulation.

Another fact is the usage of the newest stable release, to be in state-of-the-art developments and also be able to find quick supports when specific operating system manipulation or configuration problems arise. The following list covers some web based tutorials, necessary for the specific usage of the operating system in the radar applications:

- Networking Interfaces ([46])
- Network Time Protocol - NTP ([49],[14])
- RAM-Disk ([52])
- Secure Shell - SSH ([55])
- Mobile Shell - MOSH ([52])
- UDP based FTP - UFTP ([3])
- Hamachi LogMeIn ([44])
- Systemd ([57])
- Secure Shell File System - SSHFS ([56])
- Network File System - NFS ([48])
- GNU-Zip - GZIP ([43])
- Python-Virtual-Environment ([29])

2.2 Software Defined Radio

In the last years the capabilities on using digital-signal-processing (DSP) on high sampling rate hardware devices like field-programmable-gate-arrays (FPGA) and analog-digital-converters (ADC) provides a reasonable digital bandwidth to implement communication systems. There are different SDR devices on the market which could be used, but the universal-software-radio-peripheral network device (USRP-N200/210) from *Ettus-Research* ([32]) provides a high bandwidth of interfacing over gigabit ethernet with a personal computer. This is the reason, why this device is in the focus of the current developments.

2.2.1 USRP-N200

The USRP-N200 provides one transmission chain and two receiving chains. It is possible to use transmission and reception at the same time, but this implies to initialize them in the same process. Due to this fact it is assumed that the USRP-N200 is used as a transmitter or a receiver separated. This makes the software design and management implementation more easier and flexible.

Figure 4: USRP-N200 Hardware Device

The USRP-N200 transmitter and receiver architecture is detailed described in [27]. Therefore, only the illustrations of the transmission (Fig.5) and receiving chain (Fig.5) is shown. This gives a basic overview on technical implementation details to follow the concept and ideas behind the USRP-N200 devices.

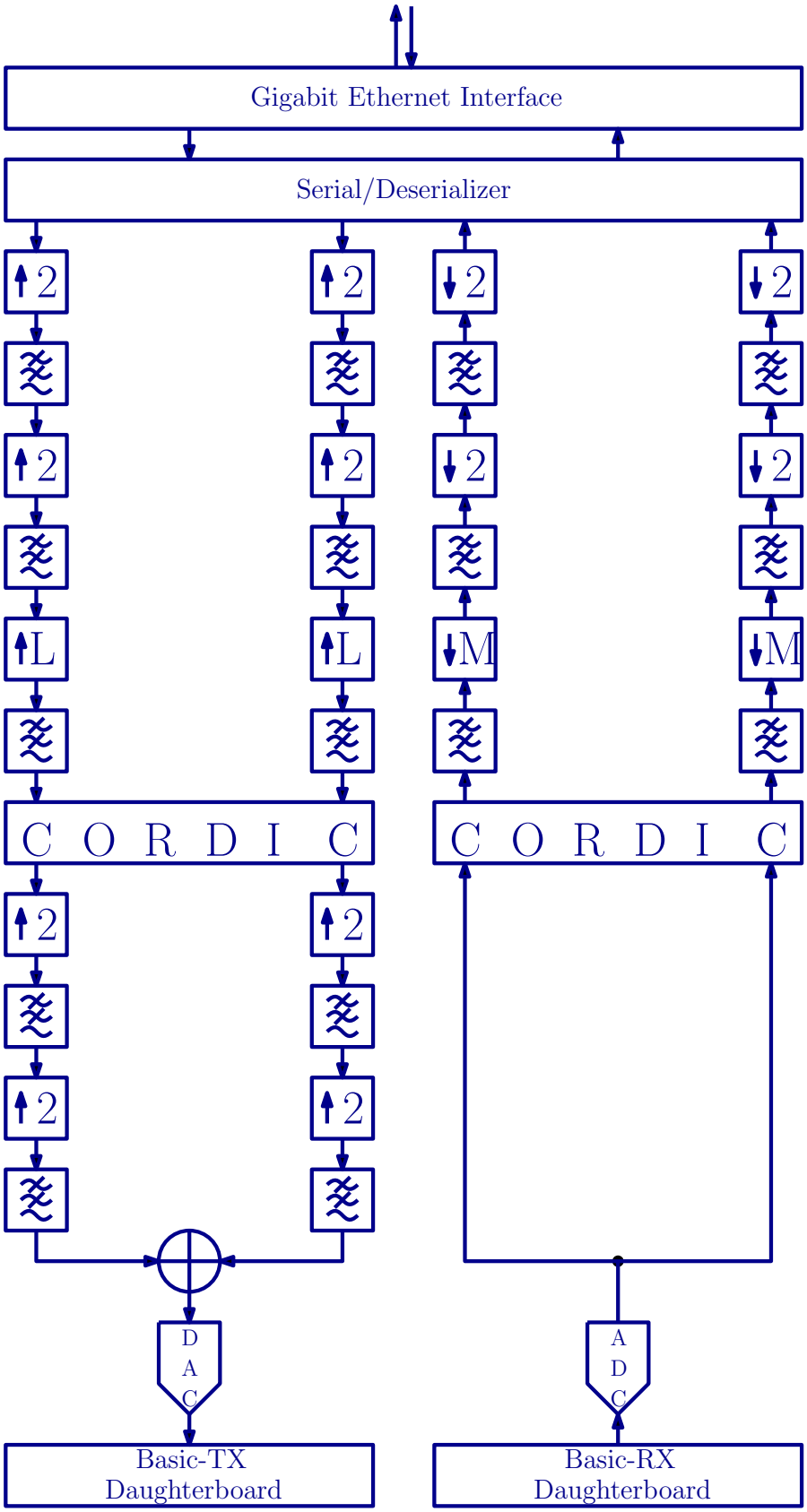


Figure 5: USRP-N200 Transmitter and Receiver Chain

As illustrated in Fig.6, the *interpolation* and *decimation* stages ([21]) are in an cascaded order for the transmitter and receiver. This cascade could be defined by the user and it is highly recommended to use the full interpolation and decimation cascaded processing chain ([33]), to avoid *frequency images* and *aliasing* ([21]).

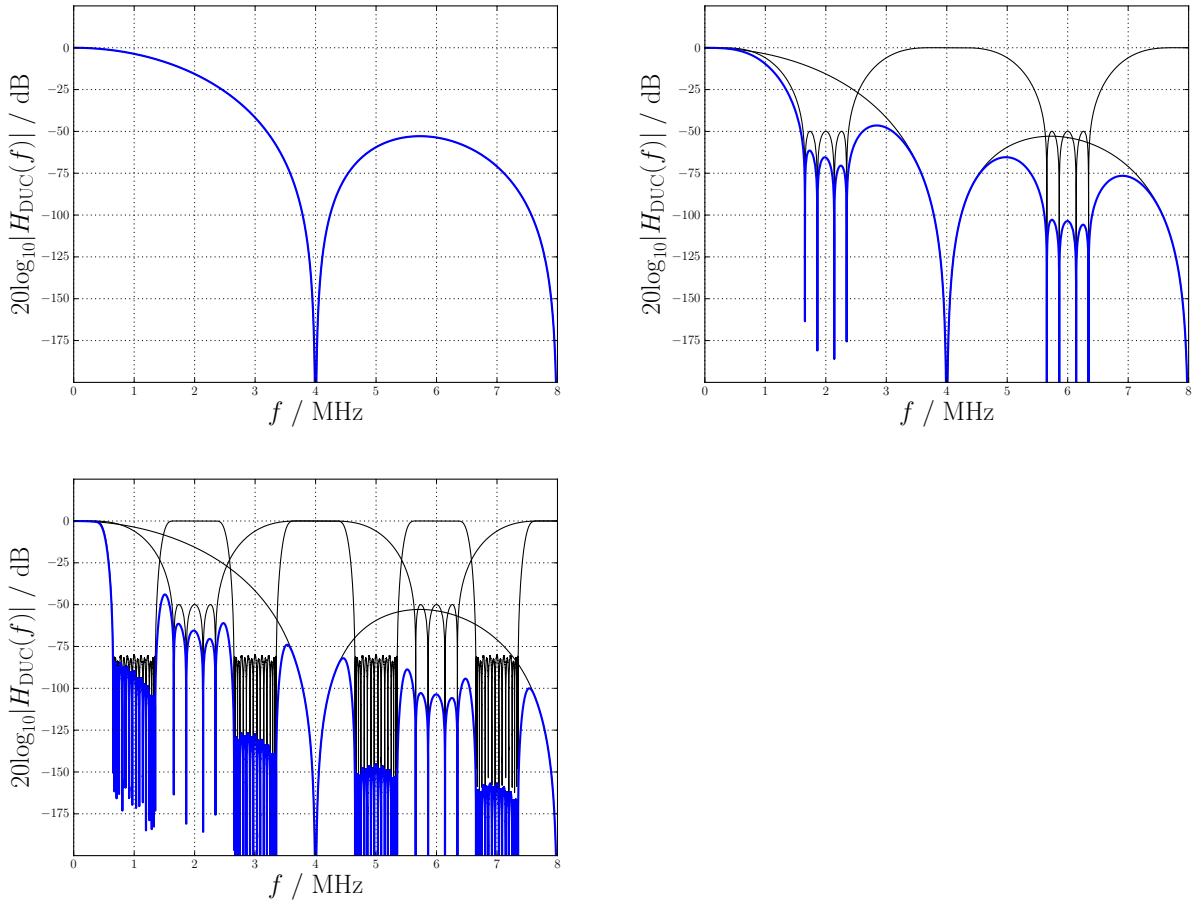


Figure 6: USRP-N200 Cascaded Interpolation/Decimation

At this point, one fact due to experience is the usage of a digital receiver near to a transmitter. When the center frequency of the transmitter is near to the receivers center frequency and the transmitting power is high related to the receiver dynamic range, then aliasing of the transmitter signal could occur in the receivers baseband signal. This could be avoided when using customized *notch filters* to reduce the signal power of the transmitter influencing the receiver.

The digital-analog-converter (DAC) used in the USRP-N200 is a special device to increase the sampling rate further to provide more bandwidth. Also the DAC is not using any image rejection filters to reduce the *first-order* model generated spectral images ([21]). Therefore the radar operator is forced to provide analog filters to remove

this spectral components of the transmitter signal, before feeeding the signal into the amplifier stages. If not, a lot of spectral components occur after the amplifier stage which are not easy to reject since the signal is on the higher power level. An example is given in the *statutory provisions* section.

2.2.2 GNU-Radio

To communicate with the USRP-N200 a soft-realtime DSP software tool, called *GNU-Radio* ([13]) is used. This software tool grows over the last decade in large steps and provides modularized implemented DSP techniques in a buffered streaming flow. The USRP-N200 use the USRP-hardware-drive (UHD) software to communicate and error handle the hardware device ([32]).

The main GNU-Radio software is written in *C++* ([10]) and provides a really good performance and *Python* ([30]) is used to interface the C++ application-programming-interface (API) functions as a wrapper ([16]). This makes it very easy to write software scripts, since GNU-Radio provides also a graphical-user-interface (GUI) to manage the DSP flowgraph it is not the aim to use the GUI to manage and control our radar applications.

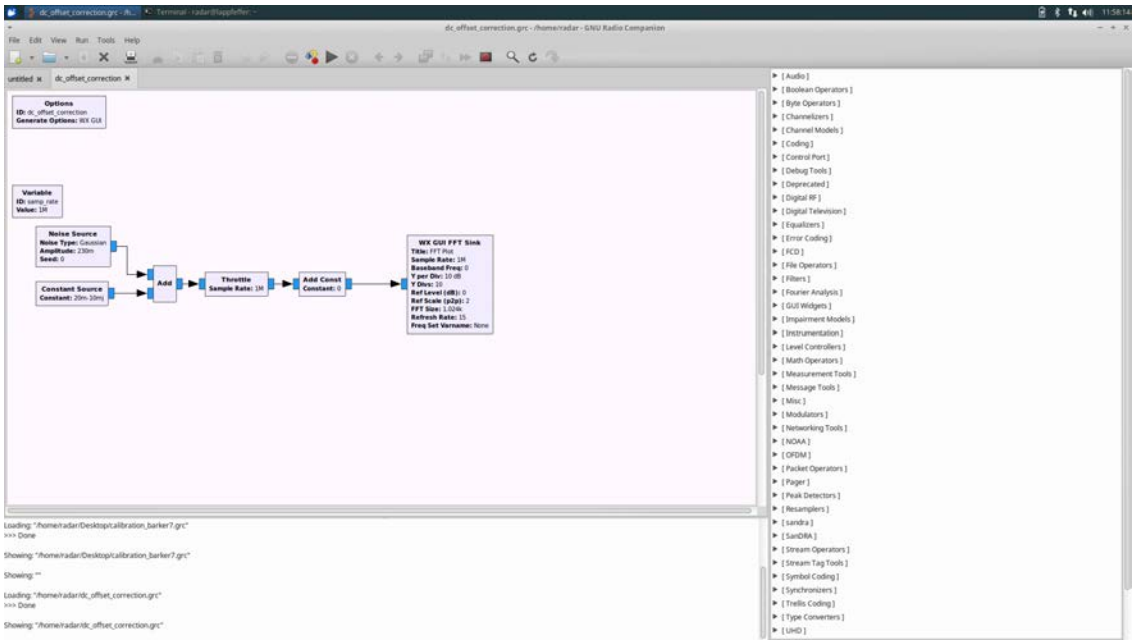


Figure 7: GNU-Radio Companion Workspace

2.2.3 Digital-RF

For raw data acquisition, a software tool called *Digital-RF* was developed from a working group of the MIT Millstone Hill Observatory. This group works on an *open radar* software ([24]) and one part of this software is this tool. The basic software is written in *C++* and *Python* to store and access digital generated raw data. An additional *out-of-tree-module* in GNU-Radio ([13]) is implemented to get the data from the GNU-Radio flowgraph and store them into files.

The features are the usage of the scientific hierarchical-data-format (HDF5), which is well optimized to store a large amount of data and also access in a performant way the sampled raw data. Since using the HDF5 libraries, the possibility to use different compression techniques is implemented in the Digital-RF software and the storage of different kind of variable types were the *complex 16-bit integer* format is the most interesting part in the implementation for this thesis.

Now only the core idea of the raw data acquistion directory and file structure, related to the raw data stream should be given and is illustrated in Fig.8 This makes it easier to understand some of the later design and implementation decisions.

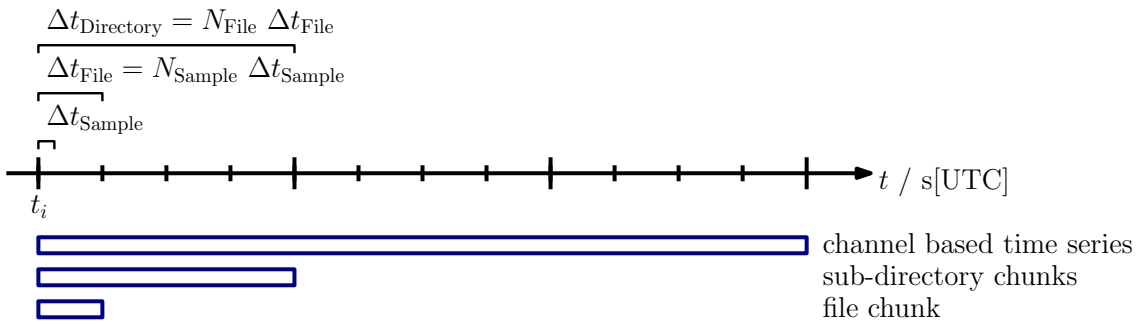
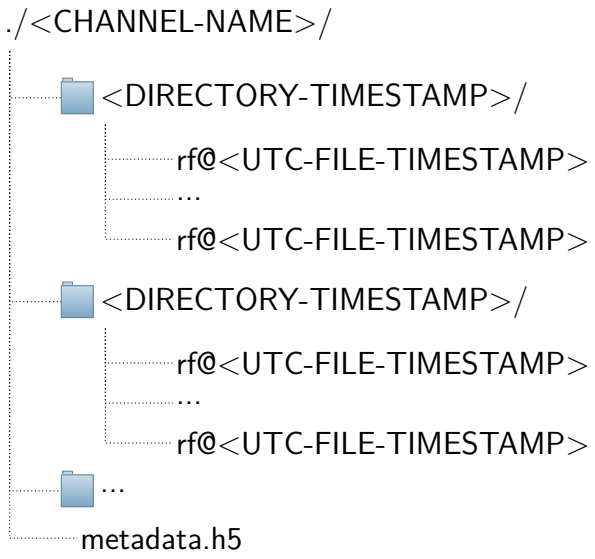


Figure 8: Digital-RF Parameter/Hierarchy

The timestamp t_i is given by the knowledge of the receiver starttime and this is related to the beginning of the coordinated-universal-time (UTC) format. This information is incremented due to the *sampling rate* and the user can define the time for the subdirectory and the file duration. From this parameters, the naming of the subdirectories and files are derived and given with timing information, which are also stored as a metadata file called *metadata.h5*. The acquisition file-structure for one channel is listed:


./<CHANNEL-NAME>/
<DIRECTORY-TIMESTAMP>/
rf@<UTC-FILE-TIMESTAMP>
...
rf@<UTC-FILE-TIMESTAMP>
<DIRECTORY-TIMESTAMP>/
rf@<UTC-FILE-TIMESTAMP>
...
rf@<UTC-FILE-TIMESTAMP>
...
metadata.h5

2.3 DSP Basics

The field of digital signal processing is widely spread. To cover the basic topics up to the advanced onces is not possible to describe them in detail. The literature for this discipline is also not possible to overlook, but in [21] the main concepts and techniques are very detailed described. The aim is to point out some special DSP concepts and techniques to follow some implementation and design details for our radar applications.

2.3.1 Integer-Band-Position

When using an ADC at some point of the signal chain, it is the interface from the analog-signal-processing (ASP) to the DSP domain. Since the USRP-N200 is used with an interfacing printed-circuit-board (PCB) called *Basic-RX Daughterboard* [32] it is important to understand that this board only interfaces to the ADC without any further filtering or amplification of radio signals. To plug an antenna direct on the input connectors it is not recommended because *aliasing* acts on the measured rawdata.

Aliasing is one of the most critical topics in DSP and is *always* present. The intensity of aliasing effects on the signal depends on the attenuation tolerance scheme of the digital filters that are used after the ADC and an analog *anti-aliasing* filter in front of the ADC is recommended to provide good radar signal quality.

This is the point to describe in more detail the concept of *integer-band-position* (IBP) sampling technique. The scenario is applied to the USRP-N200 technical specifications. The ADC and FPGA of the USRP-N200 is referenced with a 100MHz signal and the ADC amplitude response in frequency domain is a *lowpass filter* with a cut-off frequency at 300MHz. Within 300MHz bandwidth the assumption is to describe the ADC as a linear-time-invariant (LTI) system without attenuation.

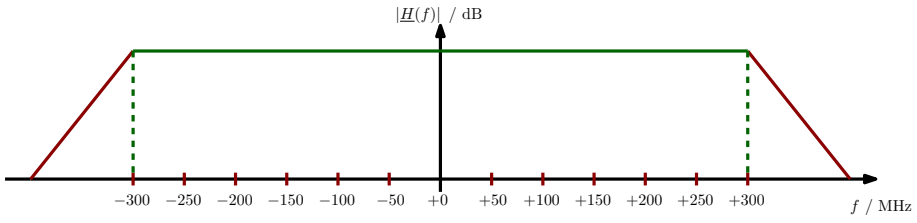


Figure 9: ADC Lowpass Characteristic

To understand the problem in detail, some scenarios are explained by examples. The easiest part for the anti-aliasing realization is an analog low-pass filtering to receive the spectral components from the first IBP. When using such analog filters, there is nothing to be aware of and the signal is in good quality after the ADC. This example is shown in Fig.10.

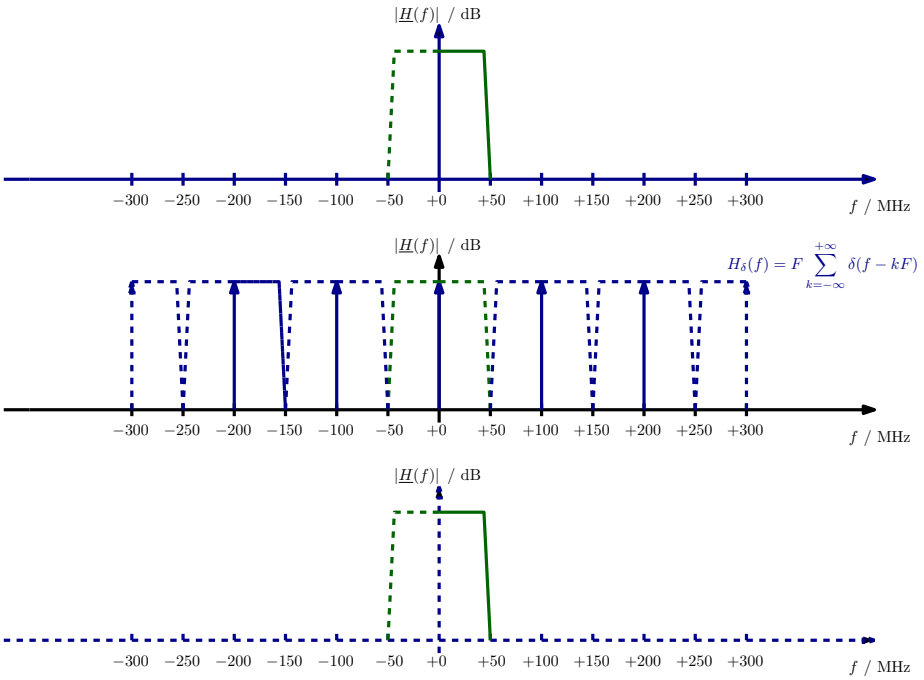


Figure 10: Integer Band Position, Baseband

Another example is related to the usage of our IAP used center frequencies and Fig.11 shows the usage of one anti-aliasing analog band-pass filter, to cover all the frequencies at once. As illustrated it is not possible to do this with one analog band-pass filter. Aliasing of the next IBP occurs in the first IBP and this causes problems in the signal quality.

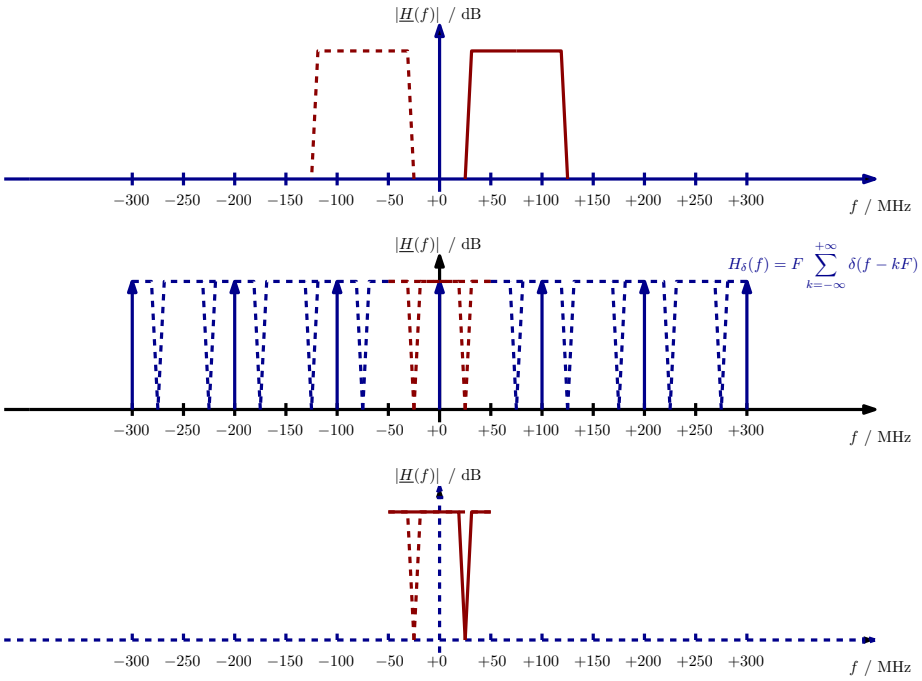


Figure 11: Integer Band Position, Aliasing

To use the IAP center frequencies above 50MHz another anti-aliasing analog band-pass filter is required in the analog receiver frontend. As seen in Fig.12, the signal is shifted to the baseband frequency spectrum without aliasing, but the frequencies are mirrored around the zero-frequency in the digital frequency domain. This is called *even* and *odd* frequency conversion and is directly related to the currently used IBP of the frequency spectrum.

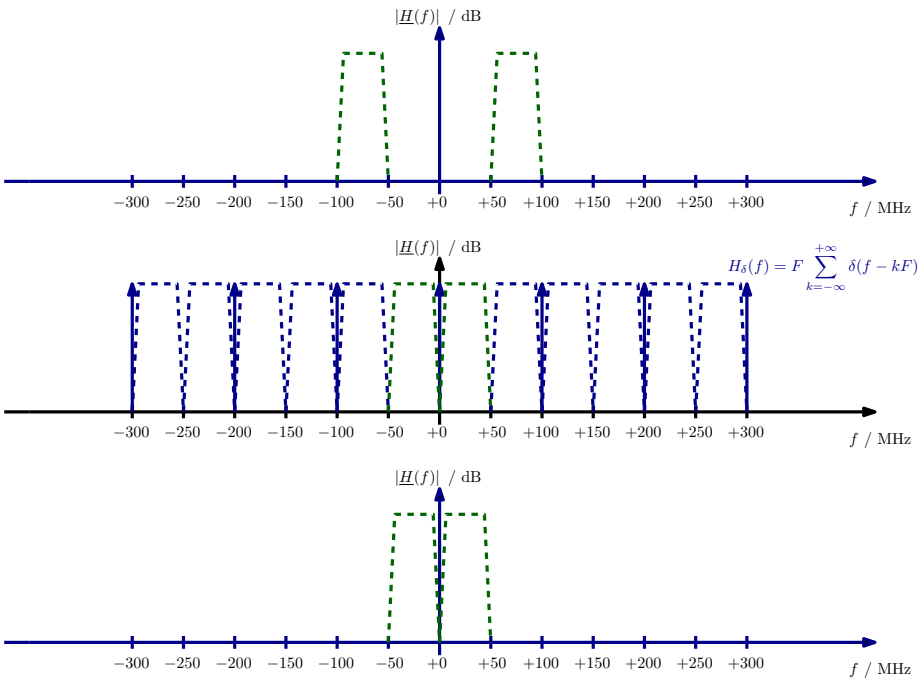


Figure 12: Integer Band Position, Bandpass

The last example (Fig.13) should demonstrate the wrong application of overlapping IBP's around the *nyquist frequency*. When the nyquist frequency is the center frequency, a total distortion of the signal occurs due to aliasing and is not usable for receiving a good quality of the radar signal. The dangerous part is, data is received, but the basic theory needs to be understand here to be able to design the right *receiver frontend* for the specific radar application.

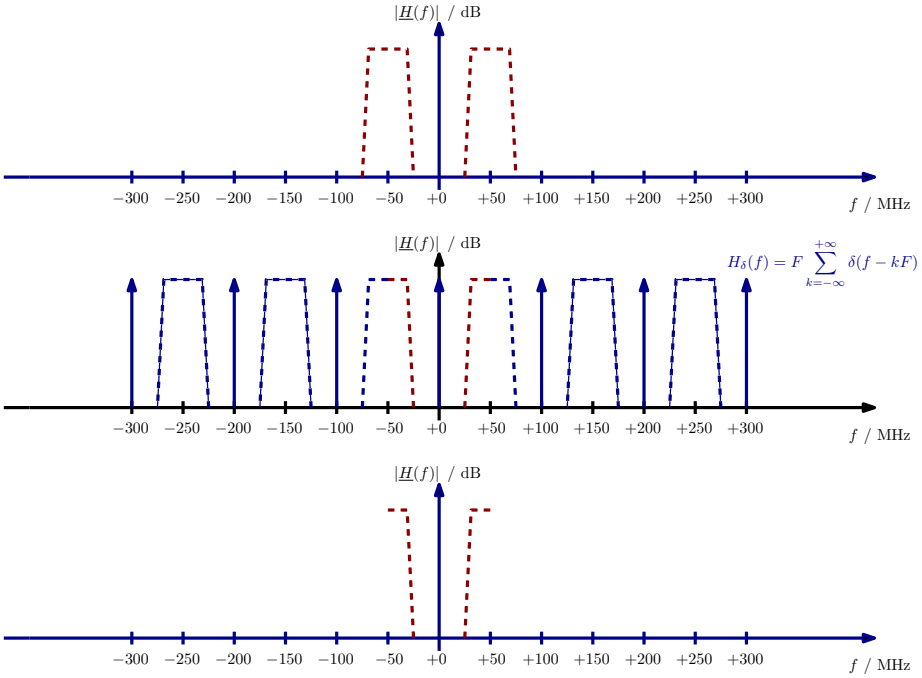


Figure 13: Integer Band Position, Bandpass Aliasing

2.3.2 Resampling

When using an arbitrary sampling rate from the USRP-N200 complex baseband signal, it is important to be able to increase or decrease the sampling rate on the computer with GNU-Radio to acquire the rawdata. Therefore, a technique called *rational resampling* is used to keep the processing as easy as possible and since the radar parameters are defined to match together it is enough to use this resampling concept.

The rational resampling is the combination of an *interpolator* and a *decimator* (Fig.14). An interpolator is an *up-sampler* followed by an *finite-impulse-response* (FIR) filter and the decimator is implemented by first apply a FIR filter followed by a *down-sampler*. The radar system parameters are defined intuitive easy that the possibility is given by using *interpolator* stages only on the transmission signal processing chain and the *decimator* stages only on the reception signal processing chain. This makes the design of the SDR radar transmitter and receiver very easy on configuration and parameter setup.

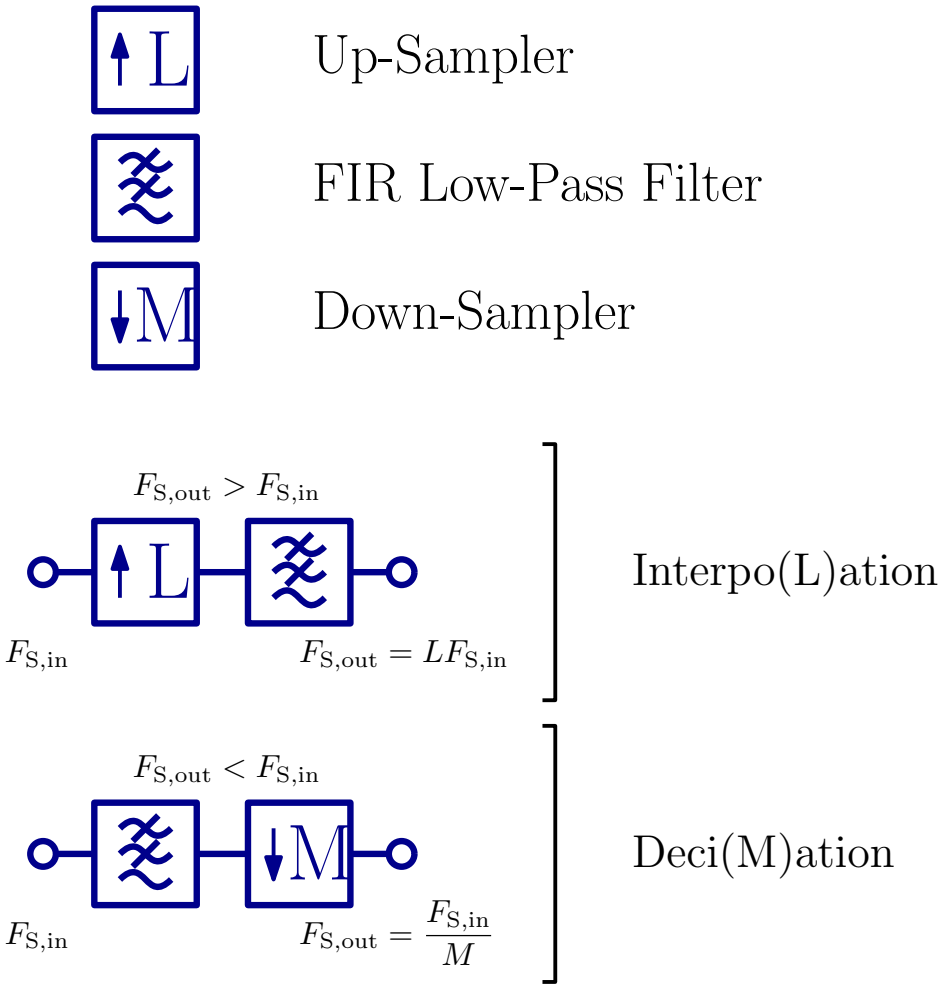


Figure 14: Interpolation/Decimation Structure

When using *center frequencies* near to the sampling rate *nyquist frequency* a complication of designing extremely sharp analog anti-aliasing filters is the result. One possibility is to change the receivers ADC sampling rate to shift the *nyquist frequency* away from the center frequency or to use an analog oscillator to shift the center frequency away from the nyquist frequency.

Both techniques implies a simplification on anti-aliasing filter design, but both techniques also imply other negative properties to take into account. This is not in the focus of this thesis because the usage of the first IBP is in focus. The possibilities are only explained, to keep it in mind when extending the radar application to such scenarios.

2.4 GPS Receiver

This devices is a good choice to realize a synchronized multi-static radar network. On the current market a lot of GPS receivers are ready to buy, but not all of them are capable for using with the USRP-N200 since the USRP-N200 needs a pulse-per-second (PPS) and a 10MHz reference signal ([33]) which is not provided by all GPS receivers. Further important parameters for buying GPS receivers is the accuracy on frequency *long-time* and *short-time* stability.

This accuracy parameters are take into account on the radar baseband signal and the radar center frequency. Since the synchronization of the baseband radar coherency is very important, the accuracy should be at least within this parameter conditions like complex baseband sampling rate and bit duration. For the occuring frequency jitter impacted by the used center frequency on separated transmitter and receiver it should be at least smaller than $f_{err} \leq 0.01\text{Hz}$ to fullfill a reasonable interval of the resulting *doppler-frequency* error.

2.4.1 Trimble Thunderbolt-E

With the previous requirements for a GPS receiver to synchronize a radar network, the used GPS receiver is the *Trimble Tunderbolt-E* device ([58]). This devices provides the necessary reference signal output and provides also enough accuracy to use this devices for our specific radar system parameters. The device is communicating over a *RS-232* serial interface, which can be connected to a computer with a *USB/RS-232* adapter. Fig.15 shows the used GPS receiver.

Figure 15: Trimble Thunderbolt E Hardware Device

2.4.2 Timestamp Formats

GPS Timestamp The most important parameter from the GPS receiver is the timestamp to trigger an interfacing computers datetime. This timestamp provided by the GPS receiver is given by separated time values:

- N_{week} .... Week Number since 1980.01.06 00:00:00
- T_{week} .... Week Seconds
- N_{leap} .... Leap Second

The *leap-second* time value is very important because this value is take some time to be fetched by the GPS receiver and this parameter needs to trigger permanently by the computer that should be synchronized. When the leap second occurs, then the GPS time values are ready to be converted to a UTC timestamp to update the computers datetime.

UTC Timestamp The UTC timestamp is an incremental timestamp since 1970.01.01 00:00:00 and could be calculated from the GPS time values provided by the GPS receiver. This is converted with the following formula:

$$T_{\text{utc}} = 315964800 + 7 \cdot N_{\text{week}} + T_{\text{week}} - N_{\text{leap}}$$

2.4.3 Reference Signal Splitter

When using one GPS receiver, a possibility to distribute the reference signal to the different devices is necessary. Fortunately there is a device produced by *Ettus Research* called *Octo-Clock* ([32]). This is an active 8-way splitter to provide the 1PPS and 10MHz reference signals. When using more than 8 referencing devices it is possible to build a tree structure of the splitters. The ratio of the price from the reference signal splitter related to the GPS receiver is 1 over 3 and therefore a good argument to use and buy it.

Figure 16: Ettus Research Reference Signal Splitter

2.5 Statutory Provisions

During the working period of this thesis, the arrangement to find places for transmitters and receivers is not only decided by the institution that uses the radar systems and the ground owners. In germany, a special authority (*Bundes-Netz-Agentur*) ([2]) takes the position to proof the legal operation of radio-wave applied systems. The main focus is here on the transmission, because this is the instrument that creates a spectra and adding this spectral component into the frequency spectra. The frequency spectra could also be seen as natural resource that is used by people, military and providers. Therefore some special laws existing to fullfill the specifications without interrupting other users of the frequency spectrum.

Frequency Certificate One part is the compliance of the assigned center frequency and the bandwidth used to transmit information in any way with electromagnetic waves and is detained in a *frequency license*. When generating spectral components out of the specified interval, forced from the authority institution, is extralegal. Examples are the usage of wrong pulse shaped sequences or the generation of higher harmonics due to non-linear behaviour of transmission power-amplifiers (PA) or DAC generated frequency images. The operator needs to take all of this conditions into account to match the statutory provisions.

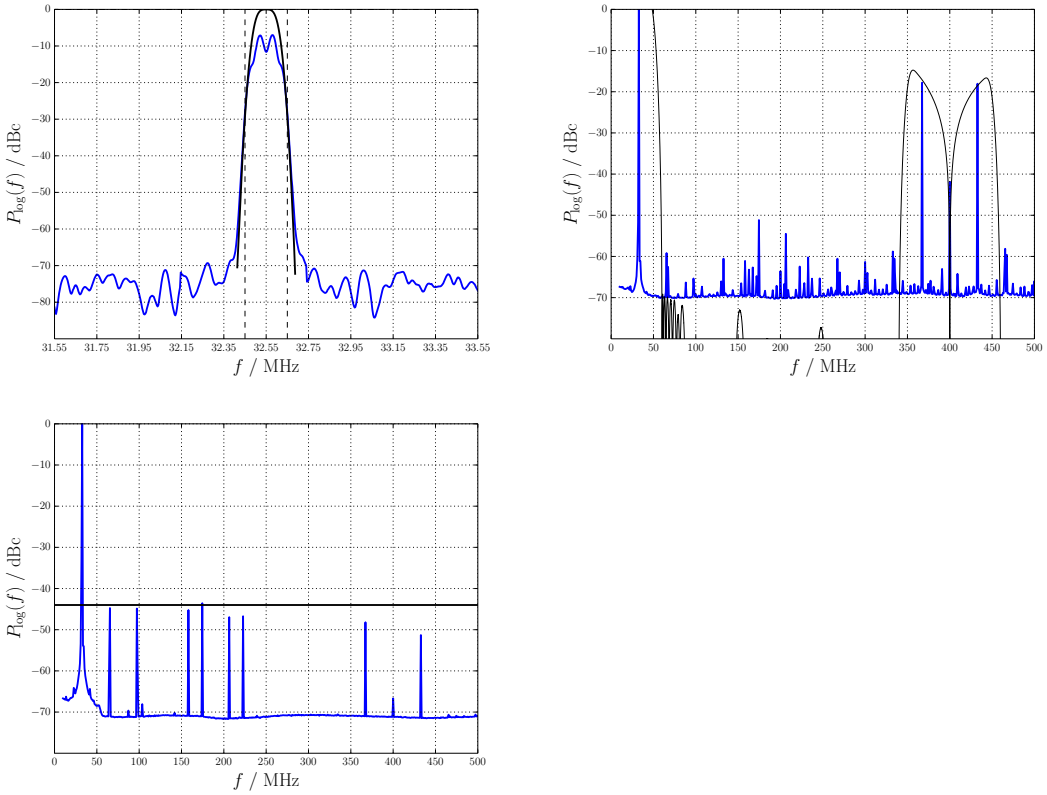


Figure 17: USRP-N200 Transmitter Signal Output

An example of generated higher order harmonics is shown in Fig.17 and the left figure on top shows the spectra in the bandwidth of interest, but when zooming over a wider frequency range, the higher harmonics produced from the DAC are present. These needs to be analog filtered before feeding this signal into the next ASP stages. The bottom figure on the left shows a filter output spectrum of the DAC signal output. In [27] a special designed pulse shape filter is used to operate transmitters within the specified bandwidth interval. In Fig.18, two types of pulse shaping filters are shown. On top, the classical *root-raised-cosine* filter response in *time* and *frequency* domain is illustrated. In comparison to this pulse shape filter, the designed pulse shaping filter with less filter coefficients is shown on the bottom figures.

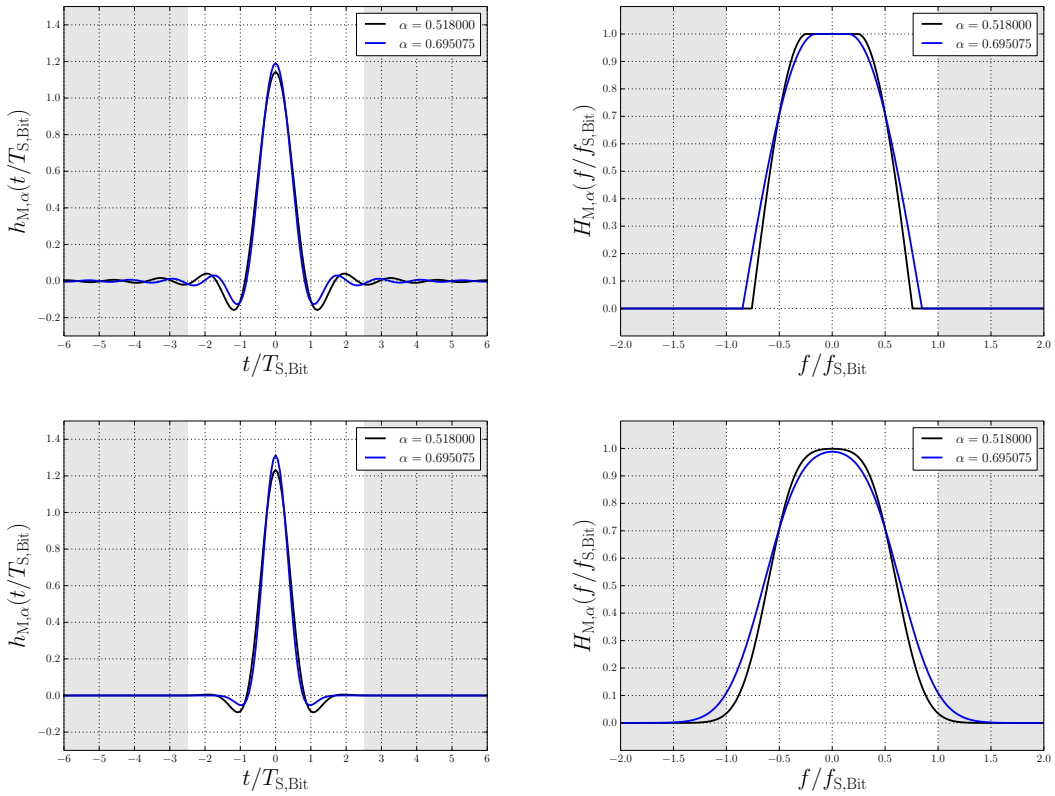


Figure 18: Root-Raised-Cosine/-Gaussian Pulse Shaping Filter

A higher-order harmonic rejection filter is used to remove the higher order spectral components, generated by the power amplifiers, before feeding the signal into the transmitter antenna.

Location Certificate Another important part is the *electromagnetic compatibility* (EMC). There are special limitations for *electric* and *magnetic* field strengths to avoid the harm of people. These limitations define the barrier distance to the transmitter antennas. This transmitter location properties are detained in a *location certificate*. These certificated documents needs to be present at all operating stations.

2.6 Radar Sequence Synthesis

The radar applications at IAP are focusing on *Impulse-Doppler-Radars*. This type of radar allows to extract physical parameters like *distance*, *doppler-frequency* and with the concept of *interferometry* it is also possible to determine with multiple channel *phase-differences* the conversion from *distance* to an *altitude* information of a scatterer ([5]). To achive this parameters, different transmission sequences are used.

Power Amplifiers After the low power sequence generation, the signal is amplified with an *amplifier*. For the specific radar application, two different type of amplifiers are used to provide enough power of the transmitted signal to be able to detect echoes on the receiver site.

- Pulse Amplifier
- Continuous Wave Amplifier

Both Amplifiers are different in design. The *Pulse Amplifier* is optimized to produce the highest signal output power when operating at a specified *duty-cycle*, which is a constant and forced the sequence parameters to match as best as possible to the duty-cycle. The *Continuous Wave Amplifier* is designed to amplify a signal that is permanent present in the time domain which is equal to a 100% duty-cycle. Both sequence types are related by a formula and is illustrated in Fig.19:

$$\frac{P_{\mathrm{CW}}}{P_{\mathrm{Pulse}}} = \frac{T_{\mathrm{Pulse}}}{T_{\mathrm{CW}}} = \text{Duty Cycle}$$

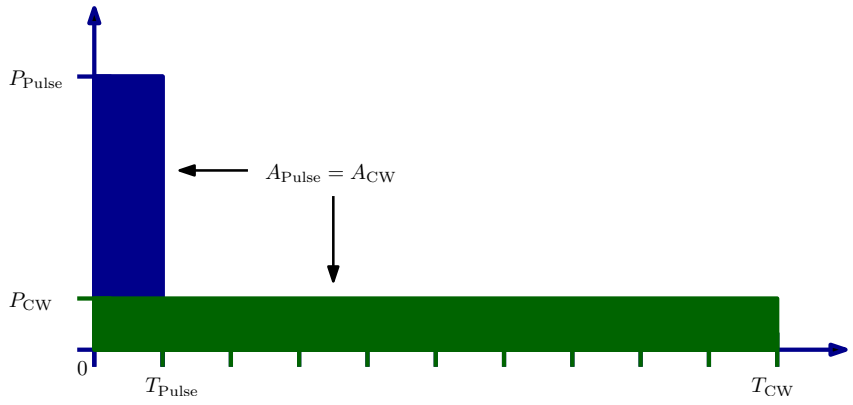


Figure 19: Peak-Power and Average-Power Relation

In Fig.20, the amplifier type is related to the sequence type that will be transmit. The operational *class* of the amplifier is also a factor in power consumptions related to the electrical efficiency. To avoid a lot of other technical issues like high non-linearities, *Class-AB* amplifiers are in operational use ([18],[34]). This class is not overall perfect in sense of linearity, but a good compromise to focus with the development on other currently more critical parts.

Sequence Type	Amplifier Type
Mono Pulse	Pulse Amplifier
Barker Codes	Pulse Amplifier
Complementary Codes	Pulse Amplifier
Continuous Wave	CW Amplifier
Pseudo Random Codes	CW Amplifier
Customized Sequence	???

Figure 20: Transmitter Sequence and Amplifier Relation

2.6.1 Mono Pulse

The easiest way to operate a pulsed radar transmitter is the *mono-pulse*. This sequence is intuitive and covers one single information bit. This implies no decoding efforts on the receiver side. This sequence is shown in Fig.21, where the left side shows the samples and pulse shape envelope of the mono pulse and the right side shows a full sequence at an example duty-cycle of 10%.

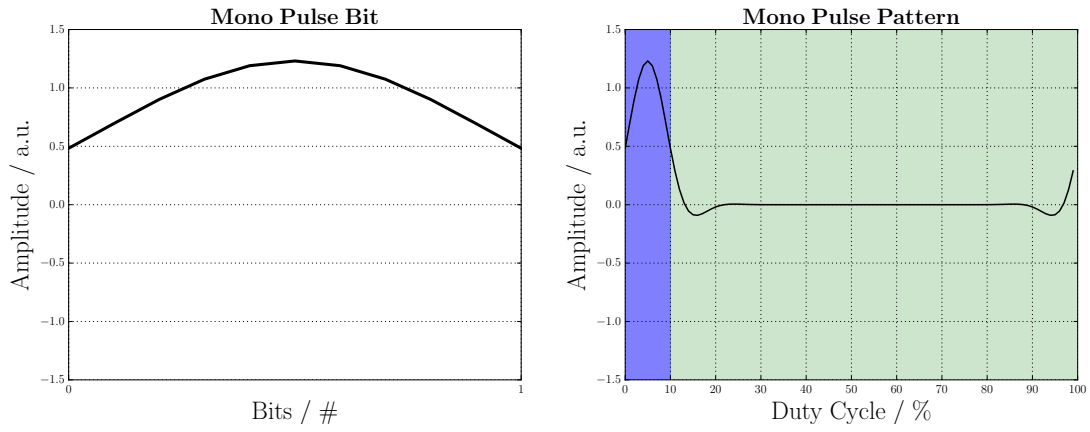


Figure 21: Mono Pulse Sequence

Now it is clear, that the sampling resolution for the ranges is not really high. One possibility is to reduce the duty-cycle to a reasonable value or to use other techniques for sequence transmission.

2.6.2 Pulse Compression Sequences

The technique used for increasing the range resolution and detectability from signals covered in atmospheric noise is the *pulse compression* technique. To demonstrate the principle, the duty cycle stays at 10% for all sequences. Not all the possible sequences are covered here, only the pulse shaped baseband patterns are illustrated.

Barker Codes One type of pulse compression sequences are the *barker codes* ([1]). These codes are repeating every pulse. In Fig.22, the top figures shows the barker-7 code in a zoomed graphic and the whole pulse sequence. The bottom figures show the same for the barker-13 code.

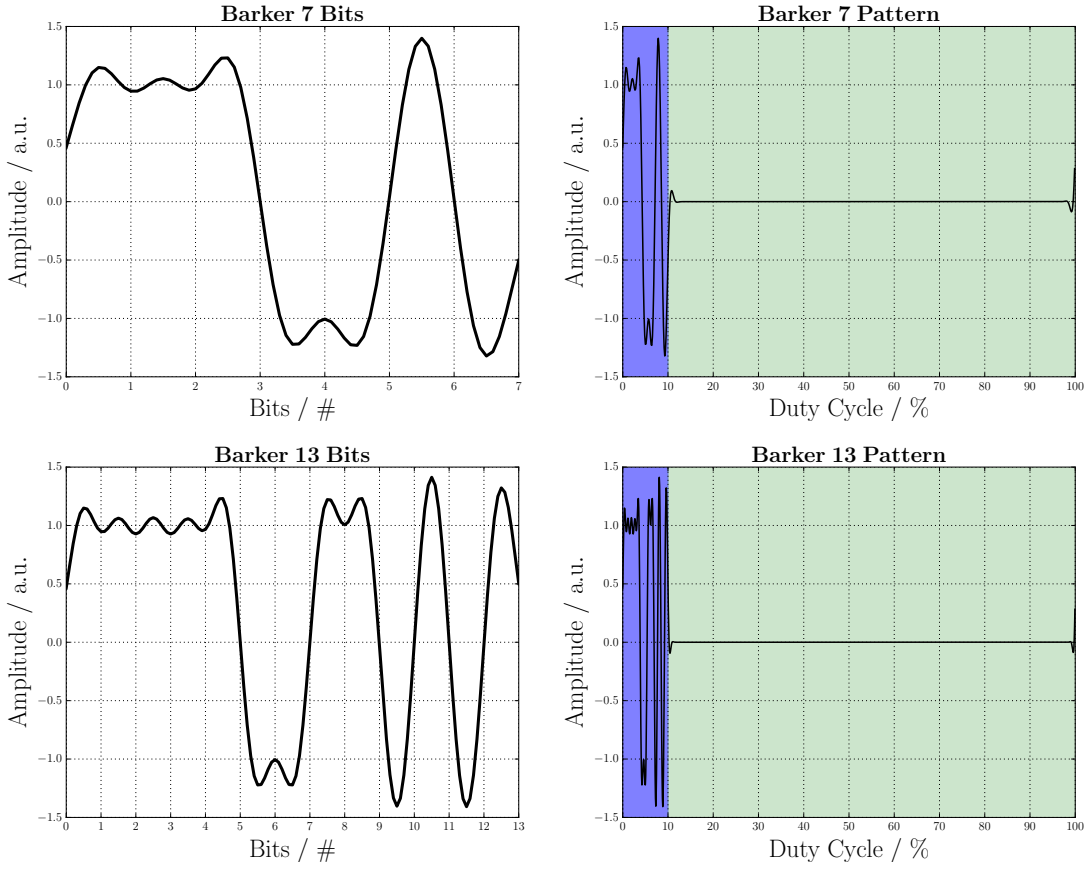


Figure 22: Barker-7 and Barker-13 Sequences

Complementary Codes Another type of pulse compression sequences are the *complementary sequences*. How these sequences are look like and what is the concept behind is given by [17]. These sequences are composed of two pulse patterns (pattern A and pattern B), which are transmitted in a concatinated way and then start repeating again. In Fig.23 an 8-bit complementary sequence is illustrated to understand the concept.

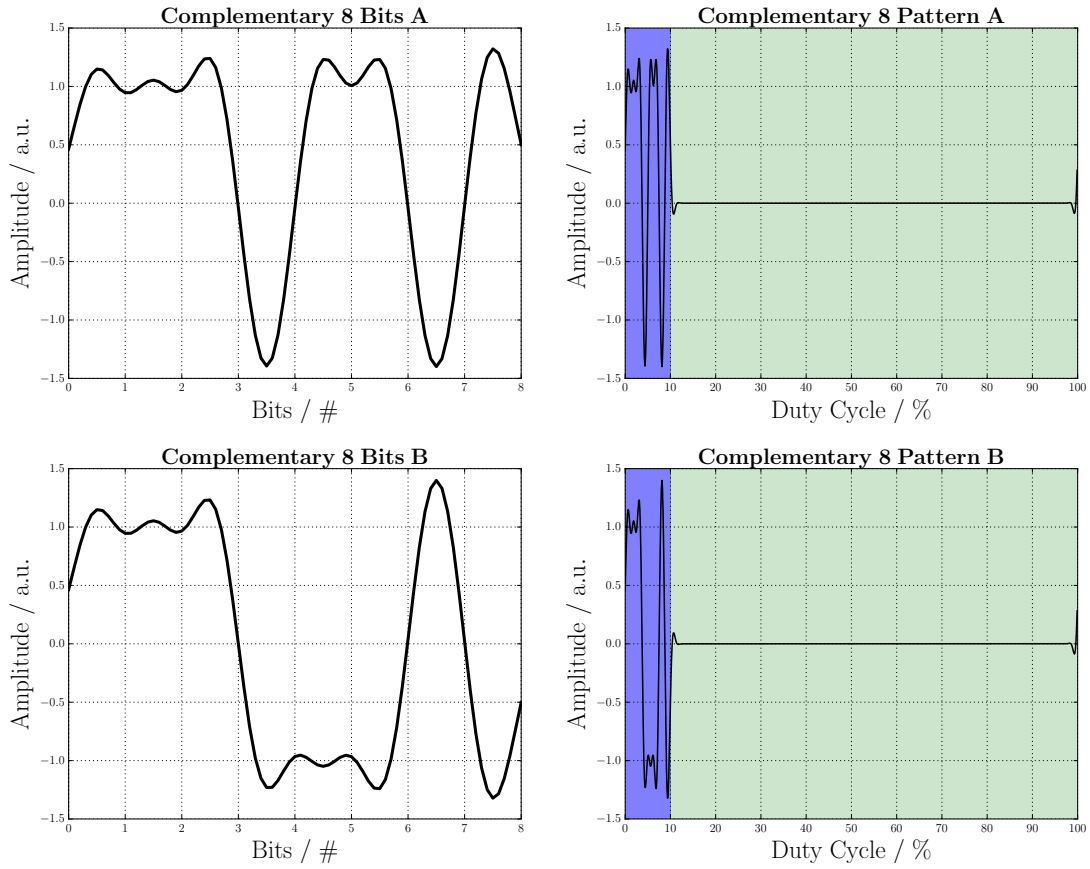


Figure 23: 8-Bit Complementary Code Sequence

2.6.3 Pseudo Random Codes

With *pseudo random codes*, the principle of pulse transmission sequences is not given anymore. This sequences are continuous. The concept is explained in [42]. These sequences are easy to generate as long the used pseudo random generator and the initial seed is known. In Fig.24, two different seed intitialized sequences are illustrated.

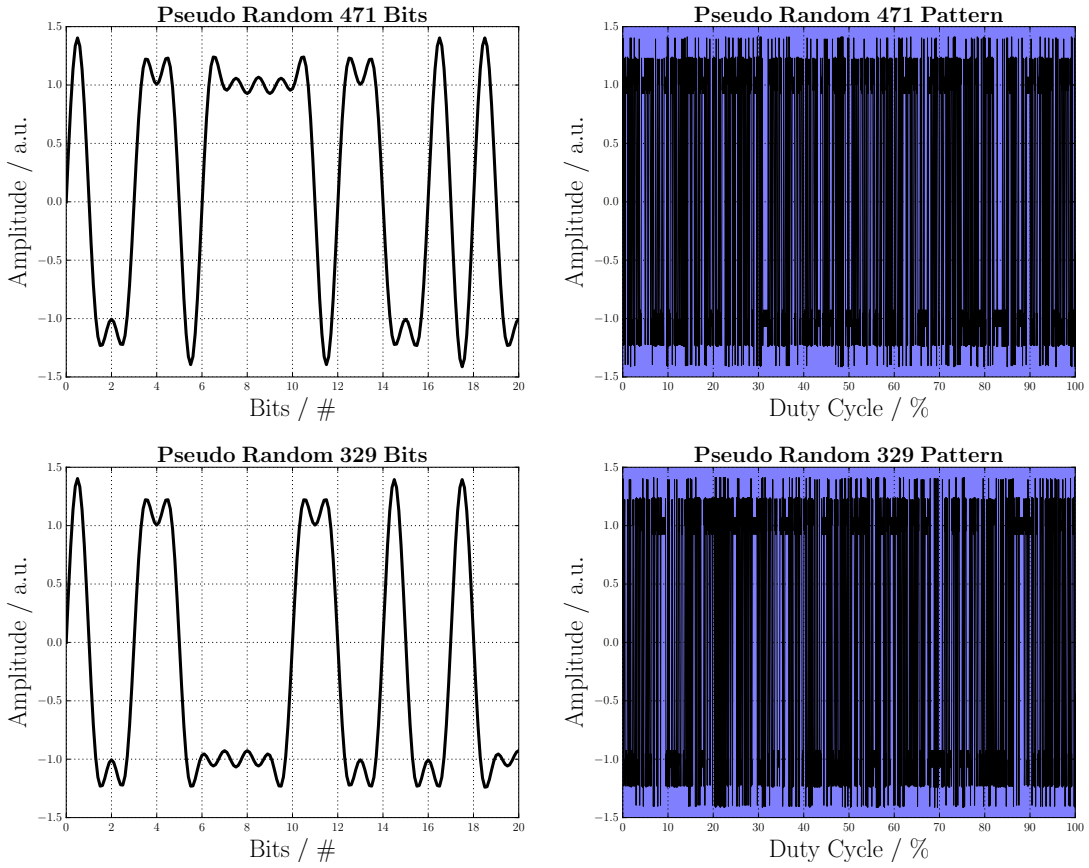


Figure 24: Seed 471/329 Pseudo Random Sequences

2.7 Radar Sequence Analysis

In the previous section, the typical radar sequence synthesis is given. Now, the analysis in sense of decoding is in the focus. Also the decoding concepts are explained by the given literature for signal synthesis. Only the differences between the decoded signals are explained and illustrated.

2.7.1 Pulse Compression Sequences

Barker Codes The barker codes are decoded via correlation analysis. After decoding, the correlated signal arised out of noise with one mainlobe and multiple sidelobes (left side ficures in Fig.25) related to the used code length. In Fig.25, the decoded barker-7 and barker-13 are shown. From this figures, the meaning of *pulse compression* is clearly seen on the right side.

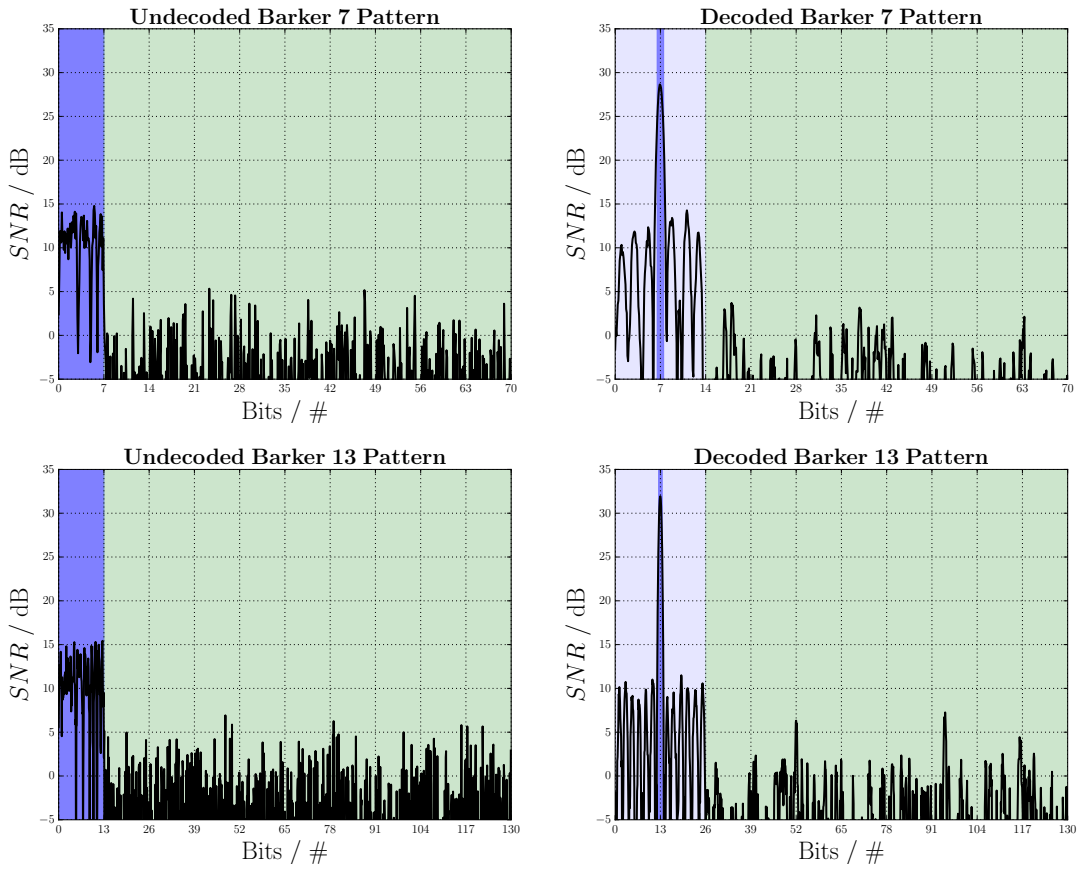


Figure 25: Decoded Barker-7 and Barker-13 Sequences

The fact, that a static sidelobe attenuation is given, the distortion between detected signals in different ranges is related to the amplitude of the scattered signal, if weak enough the sidelobes are still covered in noise and the mainlobe is still there.

Complementary Codes When using complementary sequences, the advantage is the vanishing of the sidelobes by coherent integration of both correlation decoded sequences. The disadvantage is the reduced time resolution by a factor of two. The decoding of complementary sequence with length 8 is illustrated in Fig.26 and the compression property is the same as for barker codes when increasing the code length.

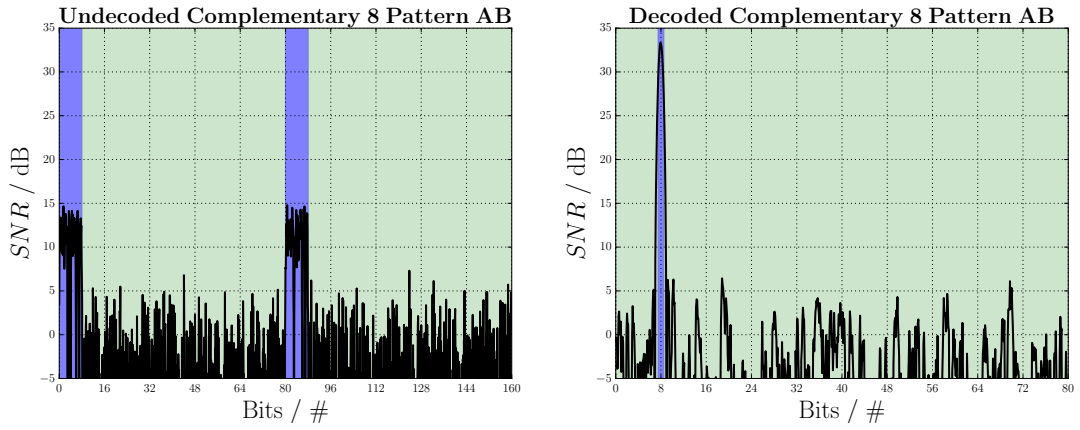


Figure 26: Decoded 8-Bit Complementary Code Sequence

2.7.2 Pseudo Random Codes

Decoding of pseudo random sequences is different. Here the concepts of *deconvolution* is used and the result after a lot of assumptions is a *matrix vector multiplication*. When applied to the received raw data, this multiplication finds the signals in the noise and a mainlobe spike is seen from the noise. This is illustrated in Fig.27 for two different pseudo random sequences covered in noise, and to clearly see the decoded mainlobe in the noise, a shift of 20 bits is applied.

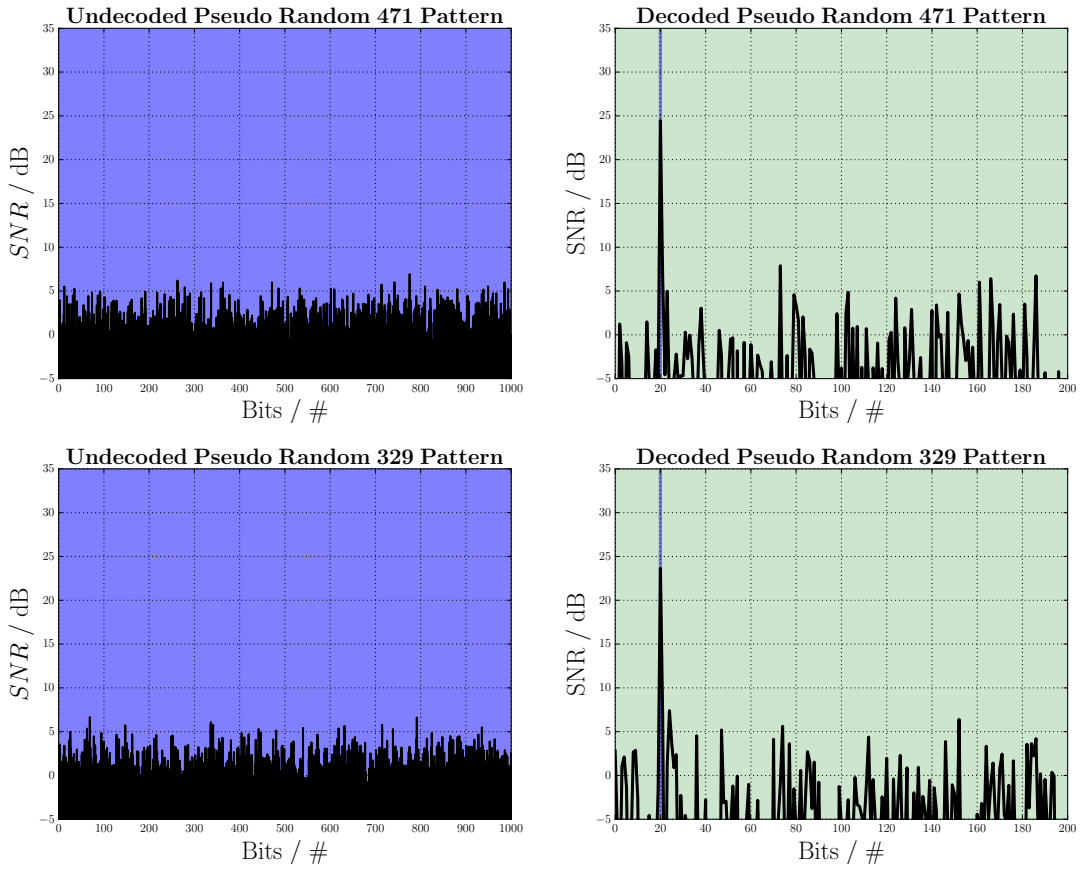


Figure 27: Decoded Seed 471/329 Pseudo Random Sequences

2.8 Radar Signal Detection

At IAP, the development for signal processing of the raw data to detect and identify specific atmospheric signal types is done and also still in progress for further extension. So far, the source code provides the detection and identification of *specular meteor echoes*. This is described in detail in [5] and this software tool is used to let it run under realtime conditions.

The additional software modifications to have a *online* processing capability is described in the implementation section of this thesis. Now, only the block diagram of the signal processing chain is shown in Fig.28.

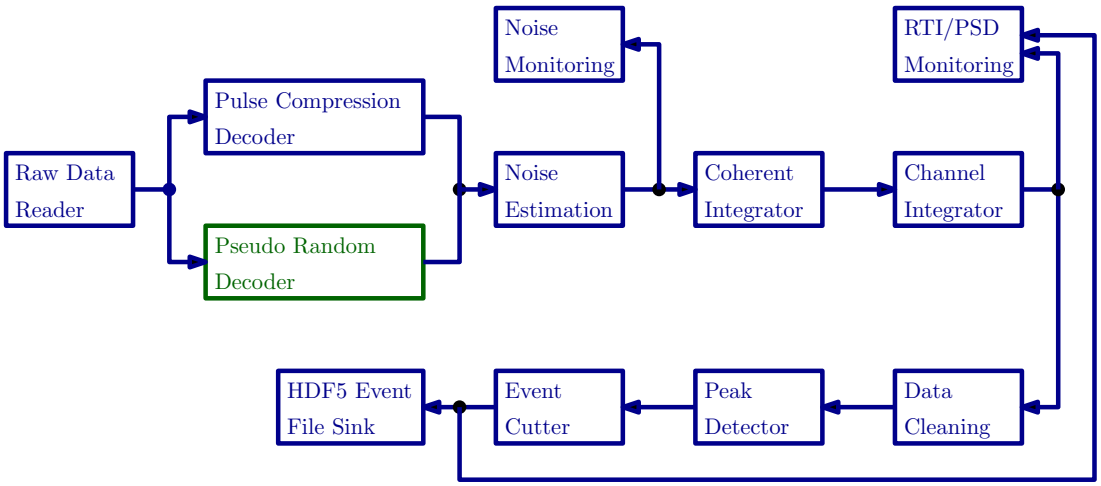


Figure 28: Detection Software Block Diagram

3 Design

This section contains the objectives, ideas and design decisions to provide ground-based measurement systems to study and extract atmospheric science parameters in a multi-static radar network. The essential elements of each radar measurement system is the combination of one transmitter and a receiver, which is called *monostatic* system if the transmitter and receiver are at the same location. By separating the transmitter and receiver spatially it converts to a *bistatic* system. If multiple combinations of *monostatic* and *bistatic* systems are in the network, the system is called *multistatic* system. The difference between the radar network types ([40]) is illustrated in Fig.29:

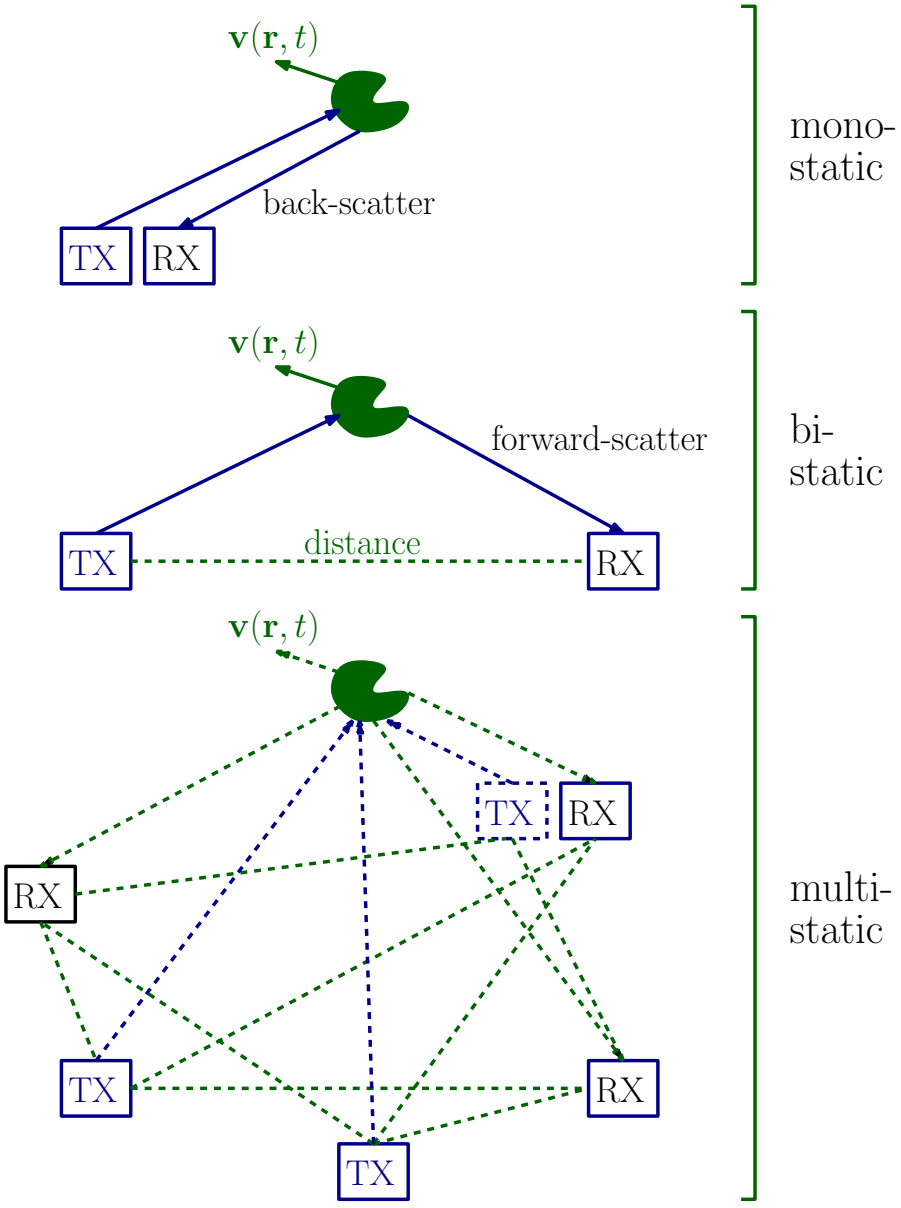


Figure 29: Radar Static Configurations

3.1 Objectives and Requirements

Multistatic Radar Network Architecture The design of multistatic radar networks is very flexible, since a lot of possibilities comes up to place the transmitters and receivers. It is fact to have less receivers as possible. The reasons for this argument is the larger size of the receiver antenna array and also the cost of interferometric receiver systems and their installation. This ends up to focus on building multiple transmitters around the receivers, because transmitters are less expensive than receivers. The structure of a *receiver centralized multistatic radar network* is shown in Fig.30:

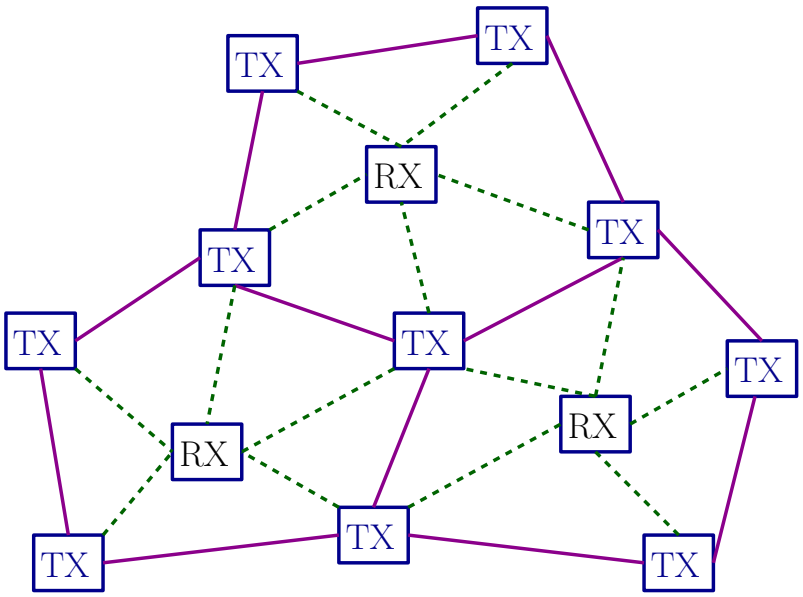


Figure 30: Receiver Centralized Multi-Static Radar Network

Radar Network Synchronization For geophysical measurement systems, especially in atmospheric science, the most important feature is to have a spatially distributed synchronization to keep the time of the measurement and the coherence ([40]) of the periodically transmitted sequences. If this is not given, a clear measurement with a known state of time is impossible. This problem is solved by using GPS receivers to synchronize technical devices like transmitter, receiver and computers.

Radar Network Accessibility A really important property of a multistatic radar network is to reach the distributed stations. This is clearly realized by using the internet. The fact that the transmitter and receiver installation is not always done on IAP owned ground results in a dependency to have a third-party-access (TPA) to a network ([37]) which is mainly managed by network administrators or public users. In some cases this could be a long and hard way of progress to get the wanted access. Therefore a mechanism is designed to have an independent network access and control of the radar. Since the buildout of mobile communication systems of different providers in most industrialized countries, it is a good opportunity to match the visions and ideas to intercommunicate with the multistatic radar network. Therefore, hardware devices called *mobile routers* are usable and an example device ([7]) is shown in Fig.31:

Figure 31: Mobile LTE Router

Computer Nowadays computers are standardly equipped with multicore processors and are relative cheap. This makes them suitable to also cover more heavy processing tasks in sense of software implementations, instead of bring most of the algorithmic calculations to hardware devices like FPGA's. To use computers in a multistatic radar network the hardware is forced to provide a mechanism to power cycle again after a system failure or a power off implied by power supply instabilities. Not all computers have this essential feature. An example of a *Mini-PC* ([62]) is shown in the picture of Fig.32:

Figure 32: Example Personal Computer

Transmitter/Receiver As mentioned before, one fundamental component is a transmitter. For our current applications they are very easy in architecture because only one or two antennas are in use. The requirements are very basic to fullfill. The transmitter should be configurable in some parameters and log information of the status to a file. The overall receiver architecture is more complicated to realize which depends on the case of application. But the application to observe meteors in the earth's atmosphere is in the main focus. At first only one frequency will be used in the receiver architecture and a channel based interferometer measurement system that is able to process the acquired raw data for different forward scatter links. The logging of status, easy configuration and pre-detected datasets are required.

Radar Signal Processing Another issue is the amount of receiver raw data collected by the acquisition. The raw data size of bytes depends on the system sampling rate and the number of channels to acquire. The higher the size of bytes become the more difficulties occur to share raw data to a central file server to process the raw data. Therefore it is necessary to process the raw data locally at the receiver stations. By using many digital receivers to fullfill the requirements of an interferometric measurement system, the processing power needs to take into account. To realize a receiver station that acquires data for one specified center frequency and process this data for different types of transmitter sequences, the processing power of a normal computer is not enough. One possibility to solve this problem is to use an expensive computing server with many processing units. But a granularity is given by processing the same raw data for different transmitter sequences. For one so called *forward scatter* link between a transmitter and receiver, a normal computer is enough to do the work of data

reduction by using detection techniques to get the signals of interest out of the raw data. This leads to a computer cluster ([28],[61]).

Radar Signal Archiving Since the detection and identification is currently at the state to extract *specular meteor echoes* from the raw data there could also be observed other atmospheric signals of interest with the radar system. An example is the measurement of *E-Region Echoes* from the ionosphere. This could be easily seen in the daily summary plots and the radar administrator is in position to define a time range when the echoes of further interest occur. Therefore a raw data archiving over a given time is needed. When such echoes are observed by the user these time series could be extracted and stored separated within the archive.

3.2 Architecture Design

3.2.1 Fundamental Structure

Computer Cluster The requirements from the previous section gives the rudimental approach to realize a multistatic radar network design. For operating transmitters and receivers, the synchronization has the highest priority and the SDR hardware is synchronized by 1PPS and a 10MHz reference signal, while the computer interfacing with the GPS needs to be synchronized via a UTC timestamp provided over a serial interface, if using a *Trimble Thunderbolt-E* GPS receiver. The next part is to have multiple computers available to do the acquisition and radar signal processing within a cluster. To use multiple computers in a cluster they need to be connected and communicate together. This can be realized in a LAN configuration. Since only one GPS receiver is needed to synchronize a radar station, a concept to synchronize a whole computer cluster needs to be applied. This could be realized by configure a local NTP server on the machine that communicates with the GPS receiver. It is highly recommended to use an original future-technology-devices-international (FTDI) USB-Serial adapter cable. Over the time a lot of communication problems occur when using cheap adapters and sometimes these devices are damaged before using them. After triggering the timestamp from the GPS receiver to the local NTP server, the computer is synchronized with the GPS receiver. This is the origin to configure the other computers in the LAN as NTP clients that listen and synchronize to the NTP server. After reaching the synchronization state, the cluster is ready to be used for radar runtime applications.

Mobile Router Another important interfacing device is the mobile router. This acts as the internet gateway ([36]) and allows the radar network administrator to have access to the main machine in the cluster and vice versa the radar station can provide logging information and datasets to a central file server. This mobile routers can be easily add in the LAN configuration. The realization to access the remote computers is done by using *Hamachi-LogMeIn* in combination with MOSH. Between the different radar adminstrators, a redundant virtual-private-network (VPN) ([33]) server is handling the sub-networks and VPN client addresses provided by *Hamachi-LogMeIn*. The accessability chain is shown in Fig.33 and gives an overview on how the computers are connected.

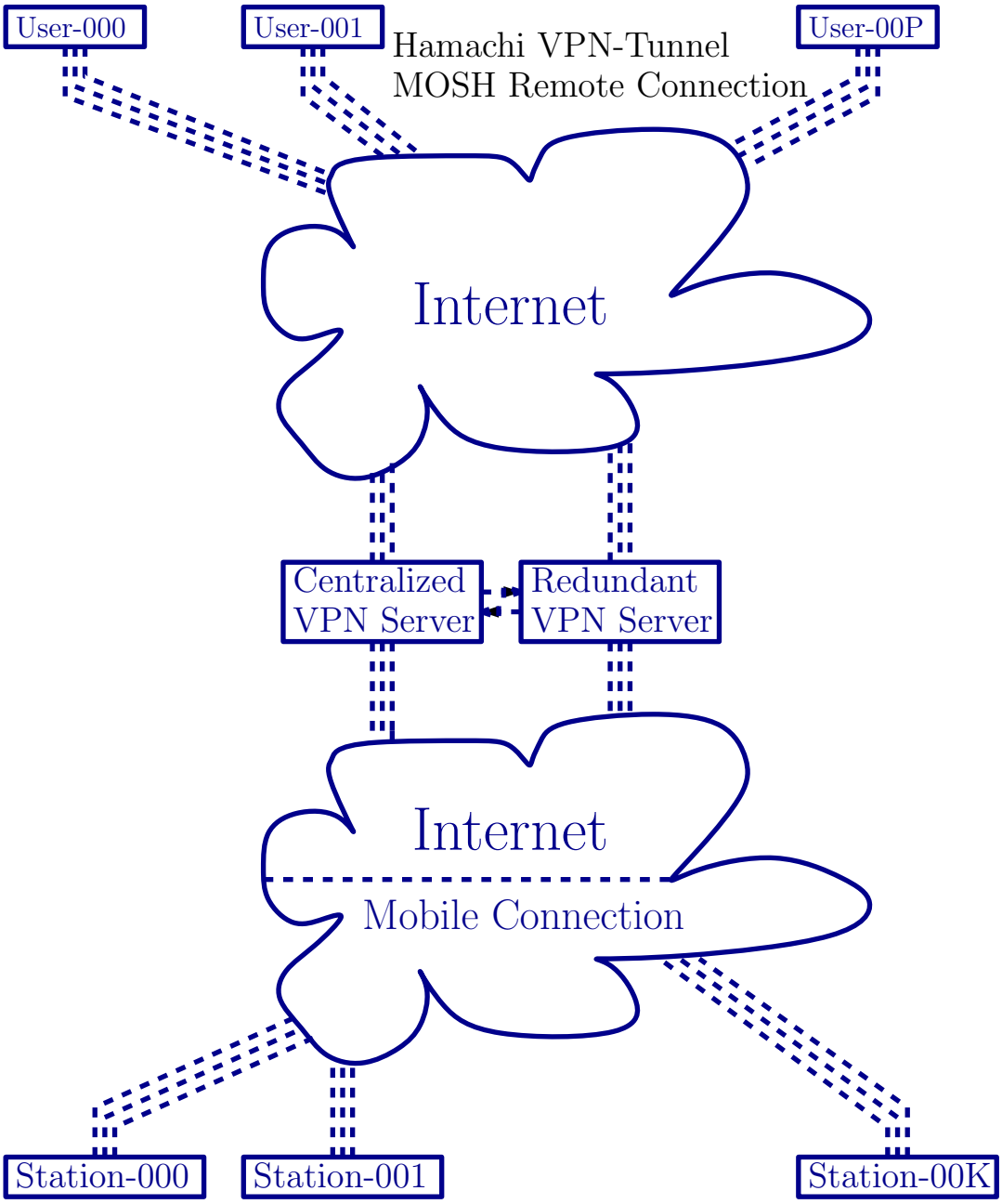


Figure 33: Radar Network Internet Connection

Layer-3 LAN Architecture During experimenting with archiving and distributing raw data within a network it figured out that using filesystem mounts like SSHFS or NFS are not good to use them for raw data sharing in a LAN. The reason is the network bandwidth limitation in the LAN. The file system mounts do not transfer the data once, instead the computer that keeps the data waits for requests from other computers and starts transferring the data to each computer in a peer-to-peer connection. When transfering the same data to multiple computer, the bandwidth in the LAN is not optimal used and can rapidly limited. This is the reason for the decision to use multicasting with a software tool called UFTP. When using multicasting techniques it

is necessary to use a more special type of network switch, called *Layer-3* switch ([22]) which should have at least gigabit ethernet connection with non-blocking capabilities.

LAN Configuration Now it is of interest to have a unified scheme to build up a computer cluster for the network stations. The LAN configuration should be implemented static which gives always the possibility to have control on the cluster and the radar administrator knows which software is executed on a node within the cluster. This static configuration is shown in Fig.34:

Class-C LAN-IP Address Space		Device Addres Dependency
$X_0Y_0Z_0.X_1Y_1Z_1.X_2Y_2Z_2.$	0	Network-IP
	1	Internet Gateway
	2	Master-Client Node
	3	Master-Client Node (Redundant)
	4 ⋮ $4+N-1$	Slave-Server Nodes
	$4+N$ ⋮ 253	Other Network Devices
	254	Guest/Control Computer
	255	Broadcast Address

Figure 34: Static IP-Address Device Mapping

3.2.2 Transmitter

Sequences The concept of a radar transmitter are periodically repeating sequences of different types. The focus in this thesis is to provide mono pulse, pulse compression (barker and complementary codes) and pseudo random sequences. These sequences are generated by pre-processing with a computer and stores a single-bit sample pattern in a binary file. The sequence is further online processed with the Gnu-Radio API's and interfacing with the USRP-N200. The online processing modifies the signal by interpolate and pulse shape the baseband signal. Then the signal is transferred over gigabit ethernet to the USRP-N200 which up-converts the signal to the wanted transmitting center frequency with a given bandwidth around the carrier.

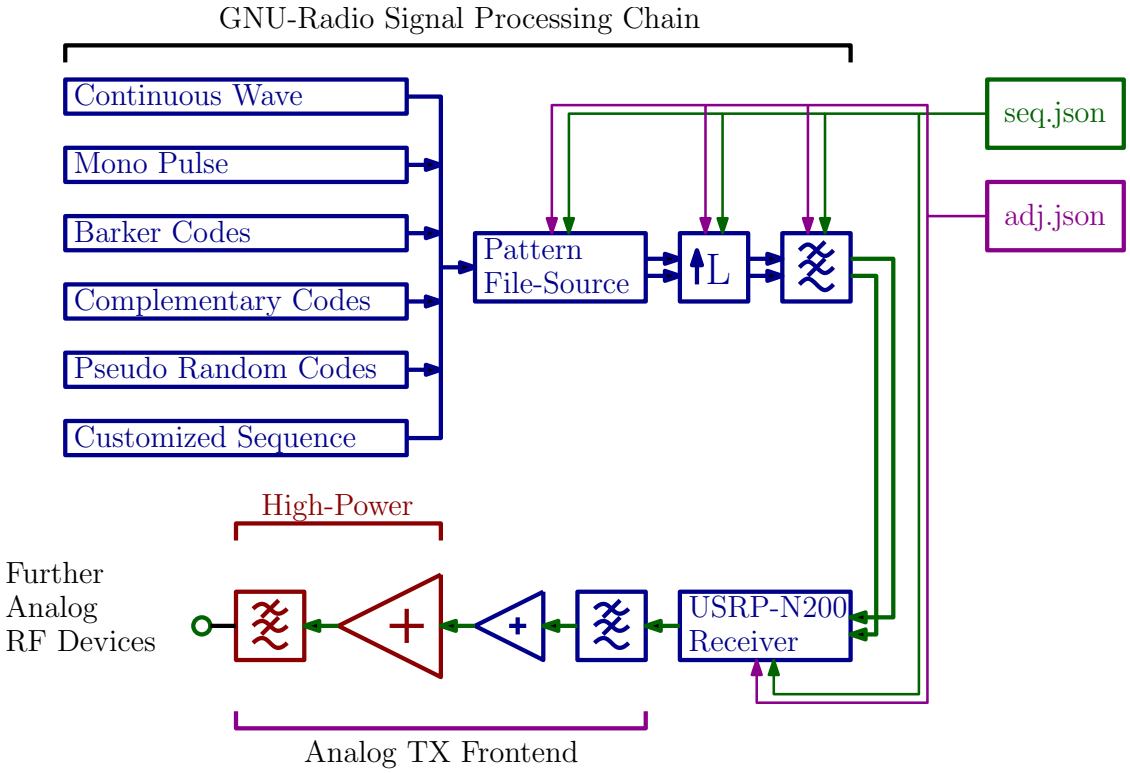


Figure 35: Transmitter Block Diagram

3.2.3 Receiver

Prototype Receiver Chain An interferometric receiver build up with multiple channels implies also the usage of multiple USRP-N200. Only a brief report of the acquisition is given because the architecture and implementation is described in [27]. The prototype receiver chain is illustrated in Fig.36 and was used for campaign based experiments, where no buffering or continuous sampling was in focus. The data processing of the acquired raw data was done after the campaign to extract the results.

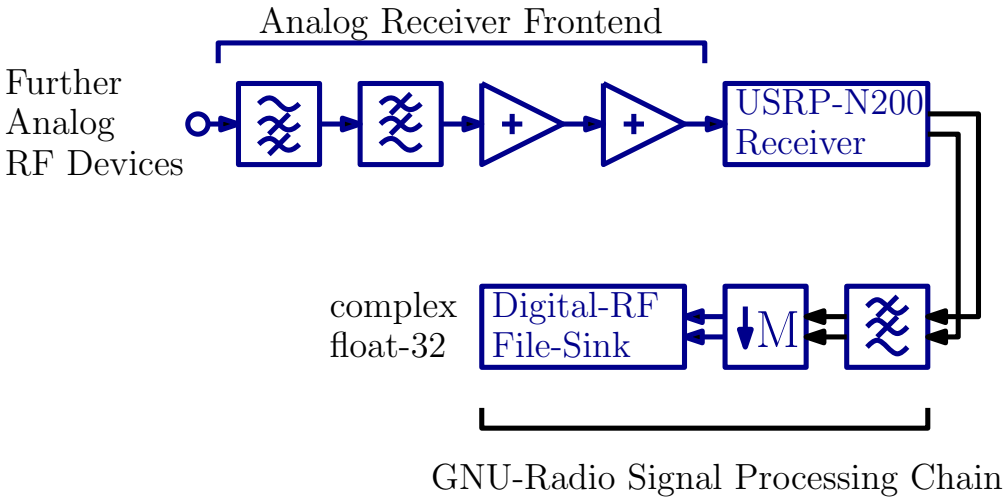


Figure 36: Prototype Receiver Block Diagram

This is the basis of proceeding with the further work on how to make the processing and data distribution more robust and have a better system configuration and management. Now the focus on one channel is take into account. This simplfies the idea and concept of the new design and features of the receiver system architecture. The raw data is sampled by the USRP-N200 and down converted into complex baseband signal and then decimated by an arbitrary factor to a reasonable sampling rate to transfer the complex samples over gigabit ethernet to an online signal processing computer which do further processing until the raw data is sampled to the Digital-RF.

Acquisition Design After type casting to 16-bit integer complex samples the raw data is sampled to the radio signal file system. The Digital-RF supports also lossless data compression in GZIP file formats and now the benefit of using 16-bit integer for data acquisition is omnipresent ([6]). Most of the radar raw data is noise and if correctly tuned by the receiver frontend and signal processing noise normalization, at least 3 or 4-bit for noise resolution are used to cover also diurnal variation of the noise level and keep the dynamic range as high as possible. If this is the case the integer complex samples in bit representation have less 1 toward 0. This is a perfect situation to achive high compression rates with the GZIP compression technique and allows a further reduction of data rates to distribute the raw data. Assume a static data rate is not possible since there could also be radio interference of other radar systems, passing cars or radio-wave reflections. This leads to a dynamic data rate, but always less than the full data rate needed without lossless GZIP compression. The raw data files are written to a computers ramdisk which supports fast writing and reading rates.

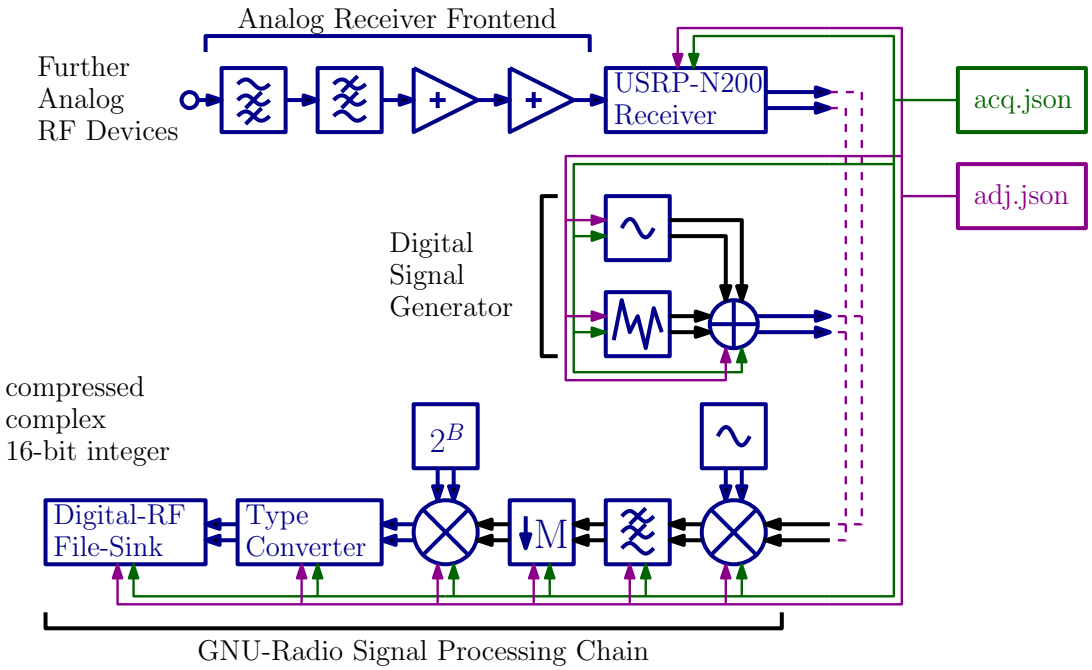


Figure 37: Receiver Block Diagram

Noise Power Gaining The new concept is to keep the data from the USRP-N200 in a 16-bit integer format and type convert the data stream to 32-bit float numbers since the range of the complex floating-point samples is in the set of $\underline{u}(n) \in [-1.0, +1.0]$ ([13],[33]) and this gives the possibility to design the receiver chain in a normalized signal processing ([21]). The standard normalization techniques are related to the signal amplitude or energy. But the maximum capable dynamic range and receiver sensitivity should be achieved. This points to keep the focus on the noise level and this is called noise-power-gaining (NPG), a non-trivial analytical procedure to apply ([39]). This technique is used to have a proportional relation of the input noise variance to the output noise variance of the acquired noise levels from the raw data. During filtering the digitized raw data signal-to-noise-ratio (SNR) increases and the noise power will be reduced. If not take care of the calculation, the noise amplitudes could be clipped at the end of the processing chain because the acquisition of 16-bit integer complex samples is forced. Therefore a pre-processing filter training is applied to calculate the proportional factor for same input and output variance to avoid intensive analytical calculations and clipping explained before.

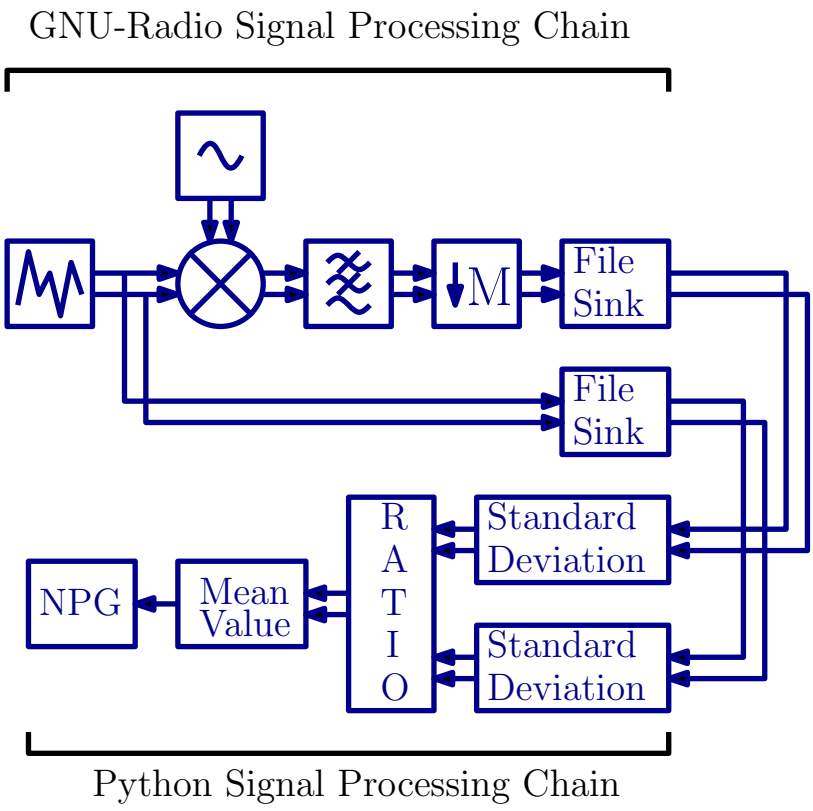


Figure 38: Noise Power Gain Training Block Diagram

Circular Buffer Design By acquire the raw data in a continuous stream of files the size of the filesystem space is a limitation factor and therefore a filesystem buffer is designed to avoid an overflow of the filesystem. The filesystem buffer is not only designed to avoid an overflow. This part of the acquisition takes the requirements of raw data archiving and distribution over UFTP into account. To realize the data distribution the structure of the Digital-RF is the key to make it more easy and efficient. The Digital-RF structure is intuitive implemented by separating the channels in directories and the sub-directory structure for one channel is to keep the timestamps in ordered configurable way to put a finite number of raw data files in a timestamp indicated sub-directory. This timestamp sub-directory needs to be compressed to reduce the hierarchical structure of the filesystem by one level. This reduces loops and sensitive triggers in the source code and is designed as a simple state machine. Also the design follows a strong hierarchy to have more control and a better opportunity to debug some contingencies.

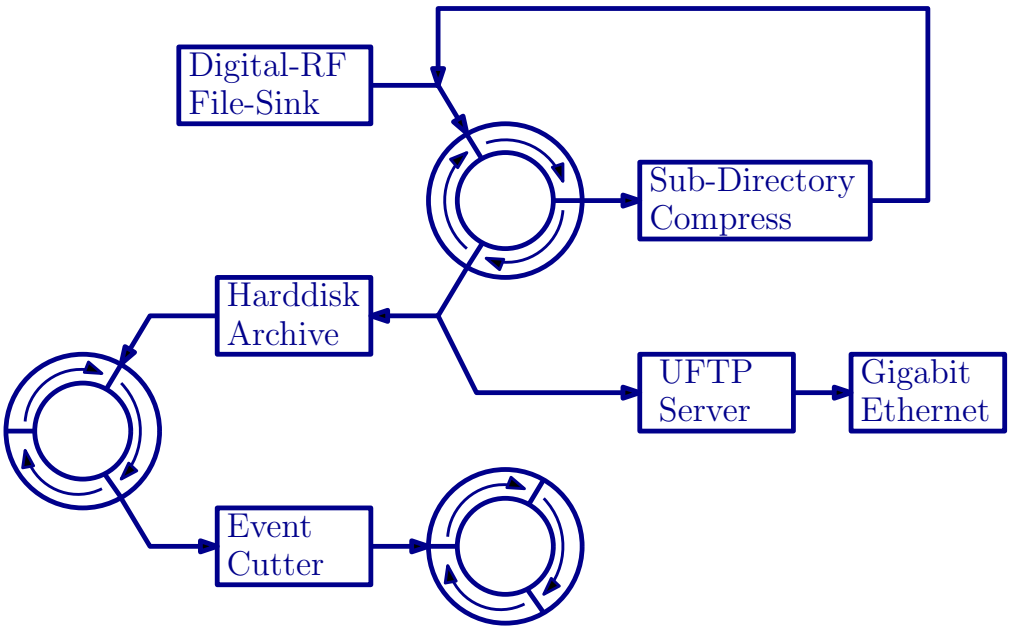


Figure 39: Raw Data Acquisition Distribution Block Diagram

Signal Detection Chain After distributing the raw data to a specific node in the computer cluster, the developed detection and identification software ([5]) is able to access the raw data to do the main data extraction procedures. Before putting the raw data to the signal processing chain, an inverted concept of the previous described circular buffer of the acquisition chain is needed. This basically takes care of the buffer overflow and de-compresses the raw data sub-directories from the Digital-RF. The detection software was modified to be able to read the raw data from the dynamically updating circular buffer and buffers also the produced detected datasets to avoid again possible overflows. The rest of the detection software is working as before with the same concepts and implementation details. Another feature is additionally included to the processing chain related to the decoding of the raw data, explained in the implementation section. The basic structure of the further data reduction chain is illustrated in Fig.40 to have an idea of what is the working flow to extract the data.

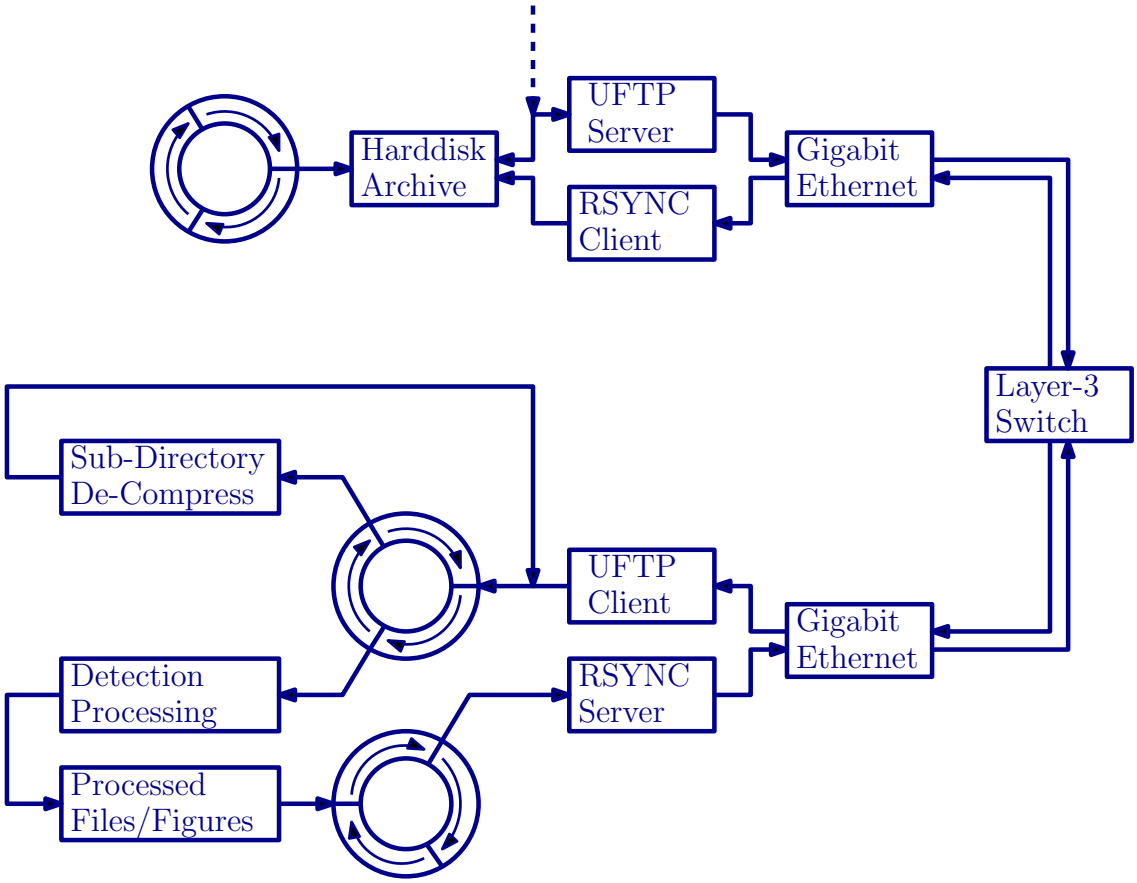


Figure 40: Raw Data Analyzing Block Diagram

4 Implementation

This section describes the concepts and part of implementation details of the software engineering ([16]) and radar-system process management. A combination of currently available software tools and own development ideas is in focus and gives at the end an overview of the whole workflow of the previous explained designs.

4.1 Client-Server Model

SSH Communication At this point the possibility to work in a LAN configuration with multiple nodes in a computer cluster is realized. But an important thing is the inter-communication between the nodes in the cluster. To keep it as intuitive and secure as possible the communication is implemented by using SSH. With SSH on Linux operating systems it is possible to execute server scripts and also return informations back from remote terminal printed informations into subprocess pipes ([29]).

Permissions Since SSH is a secure transfer mechanism, a password is required to authenticate the communication between two nodes. When running scripts in a cluster this results in a lot of password input of the radar administrator. Fortunately, there is way out of this dilemma by using *authorized keys* ([55]), based on a *Diffie-Hellman Key-Exchange*. Whenever a cluster is build up, the password-less communication between the client-node and the server-nodes is forced.

Master-Slaves Cluster The previous requirements of the inter-communication in the cluster implies an easy way to define the communication bi-directional model as *Master-Slave* implementation. The master-node acts as a client and the slave-nodes acting as servers and the pronouncing in this special type of cluster application is *Master-Client* and *Slave-Servers* ([61],[26]). This abstraction of the fundamental basement for cluster computing is illustrated in Fig.41:

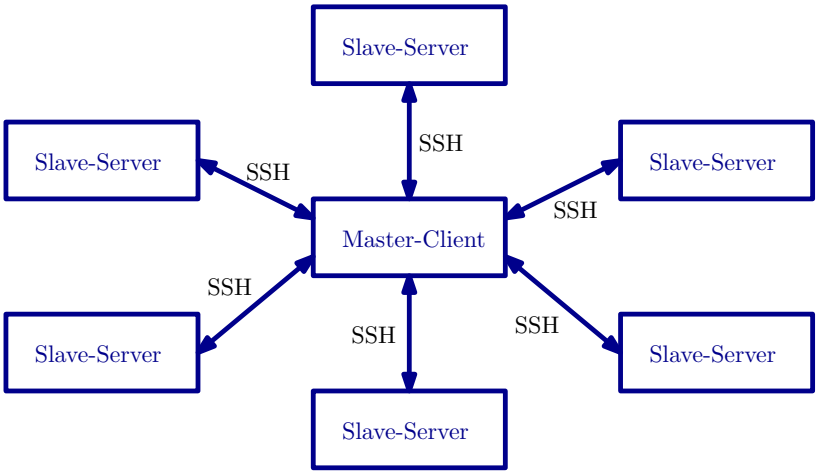


Figure 41: Master-Client Slave-Server Concept

4.2 Client Scripts

4.2.1 Configuration Hierarchy

The configuration of the radar system is implemented with java-script-object-notation (JSON) ([9]) configuration files in the project management. All hardware components are connected with a specific node in the cluster and do a specific task by communicating over interfaces. This is also a good point to define a hierarchy for the configuration parameters. The top level of the hierarchy is the server-node internet-protocol (IP) address in the LAN, so the client script maps the server script and specific parameters to the node where the server script needs to be executed. By communicating with the hardware over interfaces, this is the next level in the configuration hierarchy. All the underlying configuration parameters depending on the parameters of the specific client and server scripts.

After reading the configuration files in the client scripts, these configuration variables are separated for the server scripts to point the right parameters for the correct execution. For some server scripts the parameters are large and it is not recommended to put all the parameters over argument parsing. To avoid this, the separated JSON parameters are stored in a unique sub-configuration JSON file and will be send to the specific server-node. Only by giving the absolute path to the configuration file on the server node, the parameters will be read from the JSON file by the server script. This intuitive configuration hierarchy is shown in Fig.42:

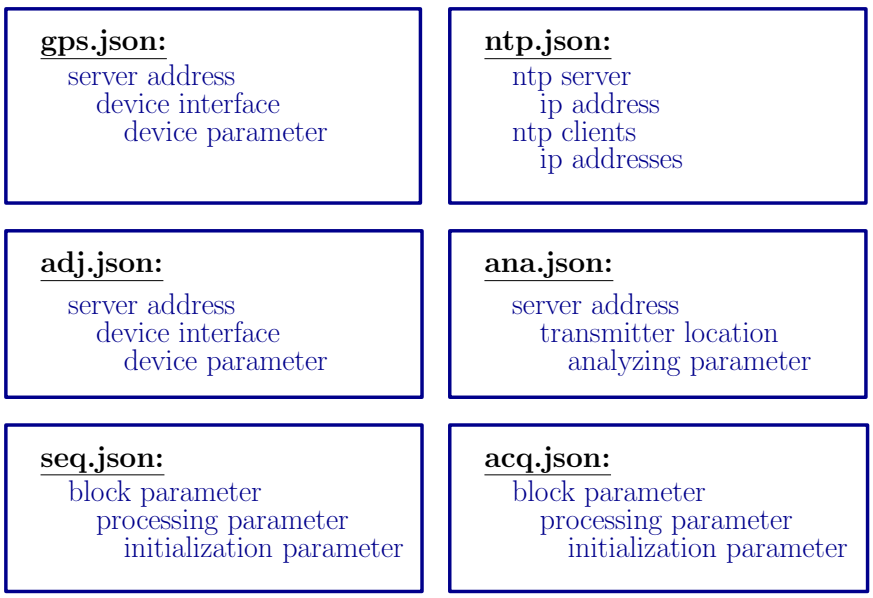


Figure 42: JSON Cofiguration File Hierarchy

4.2.2 Logging Hierarchy

For each started server script exists a logging file in the project management. These logging file registers the state and the errors of the radar system operation. Whenever an error occurs, the radar administrator is able to know where the error happens and what was the origin of the error ([31]). That implies an easy identification and allows a quick repair/exchange of the damaged hardware. The logging files are one of the parts to transfer to a central file server, because these files are not large in size of bytes. All logging prints were ordered in a structured hierarchy specified by a timestamp, server application and process state information. This hierarchy is illustrated in Fig.43:

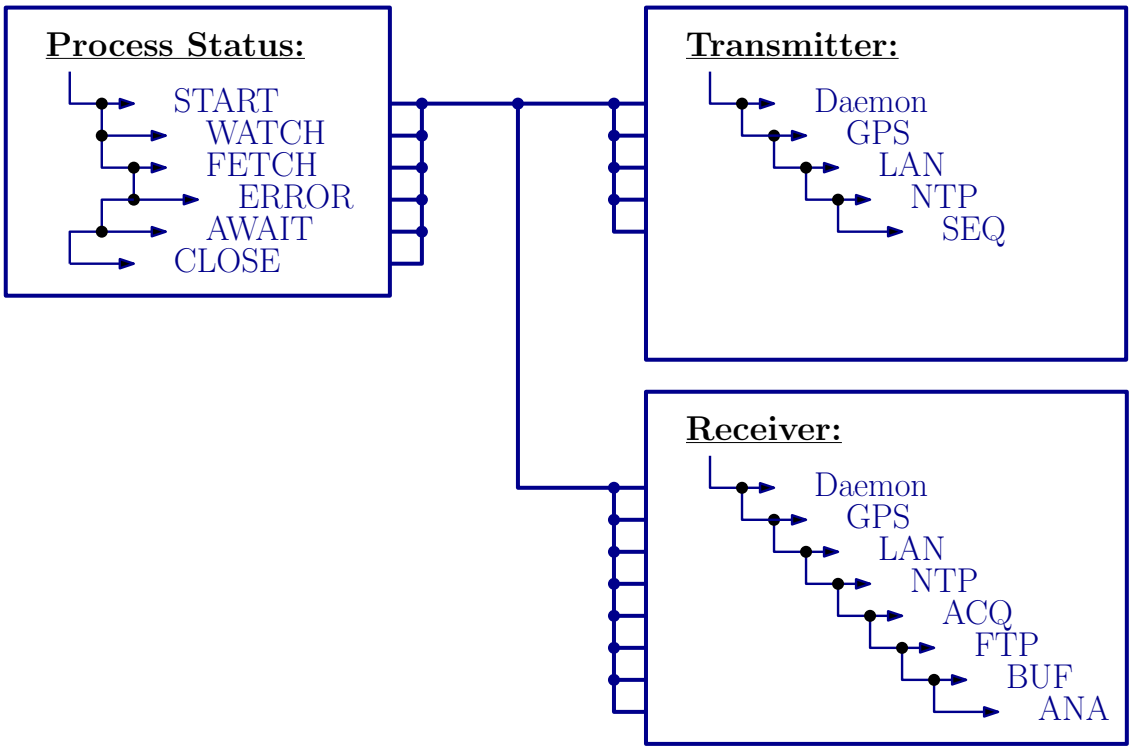


Figure 43: Logging Information File Hierarchy

4.2.3 Systemd

Service Scripts For radar system operation it is important to be able to *start*, *stop* and *restart* the client scripts. These are the basic command implementation for Linux operating system daemons and this concept is adaptable for our radar system management. In Linux the core implementations are so-called *init.d* scripts ([45]), but trying it out to execute python scripts, a lot of problems arised. Fortunately, another system daemon project is implemented in Linux in the last years, called *systemd* ([57]). This is a very easy to configure and understandable software tool, to realize an automated

and programming-language independent script execution. The basic synthax is almost the same as with the *init.d* implementation.

- `$ sudo systemctl start` <SERVICE-TO-EXECUTE>
- `$ sudo systemctl restart` <SERVICE-TO-EXECUTE>
- `$ sudo systemctl stop` <SERVICE-TO-EXECUTE>
- `$ sudo systemctl status` <SERVICE-TO-EXECUTE>
- `$ sudo systemctl enable` <SERVICE-TO-EXECUTE>
- `$ sudo systemctl daemon-reload`

Handler Scripts With such handler scripts, which works stringent together with the client scripts, it is possible to do a script execution with systemd before and after the main client script. This is very comfortable, because the project management will be cleaned anyway by all kind of system failures or total radar system collapse due to power off. This realizes a system self-survey without interaction of the radar administrator.

Daemon-Workflow When the scripts are ready, the system daemon services and handlers are ready to be executed through systemd. In Fig.44, the workflow is illustrated. The core concept is the service script for systemd. This holds each necessary parameter for the daemon process execution. For the service configuration files, templates are created in the software project to be able to quickly find the modification steps and the meaning behind the parameters.

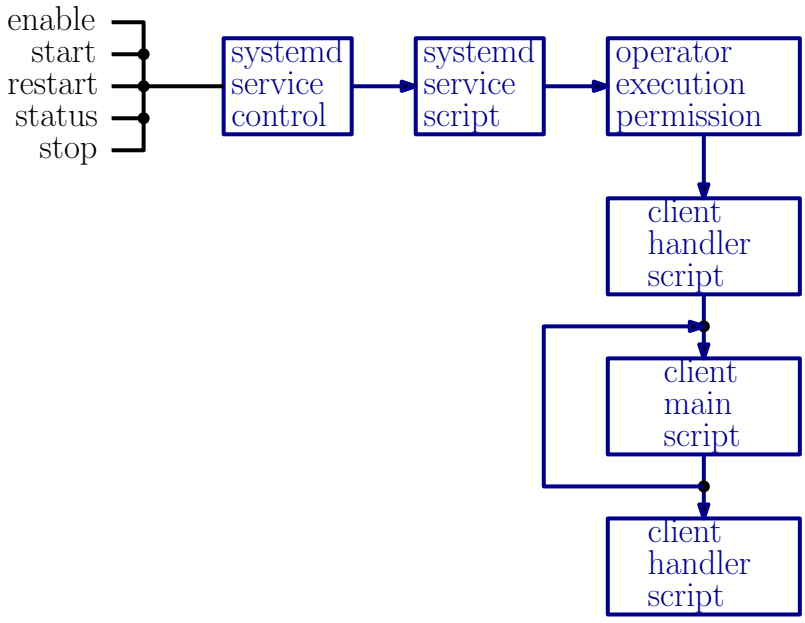


Figure 44: Systemd Service Workflow

4.2.4 Synchronization

GPS Receiver Interfaces Without a synchronized radar system it is not recommended to operate a radar system. This clearly implies that the synchronization for each station is set with the highest priority in the management hierarchy. Since the *Trimble Thunderbolt-E* GPS receiver is used, the handling and hardware observation during runtime is in the focus of the implementation. The device itself provides the folllowing interface ([58]):

- GPS receiver antenna
- 1PPS
- 10MHz reference signal
- RS-232 serial interface

GPS Receiver Information All the information of the GPS receiver state is transferred over the serial interface based on the trimble-serial-interface-protocol (TSIP) information exchange ([58]). For this special protocol, developed by Trimble themselve, exists a Python module called *Python-TSIP* ([19]). With this software part it is possible to get the specific information from the receiver that is needed. The GPS receiver documentation showed that there are a lot of information parameters the GPS receiver can deliver if needed. But not all information is necessary to get the synchronization status. There is one special protocol information report (0x8F-0xAC) with bit-coded information of the following states:

- receiver mode
- disciplining mode
- holdover duration
- minor alarms
- gps decoding
- disciplining activity

The listed parameters are the most critical to take care during runtime and have a general overview of the GPS receiver synchronization and interface state. These parameters can be read out in a loop with a program and a decision of the synchronization state can be made during runtime. The main output of the program is the receiver information of synchronization or un-synchronization, which provides the information for the next hierarchy level in the radar system.

Datetime Update But this is not the only thing needed to synchronize one node in the computer cluster. When the synchronization state is reached, the client script starts to read out another parameter. This parameter is the GPS receiver time given by the *GPS-Time-Format*, which is a special time format. After reading this information, a conversion to the *UTC-Time-Format* is calculated and this timestamp updates the computer *date time*. This is the basic implementation of the synchronization script.

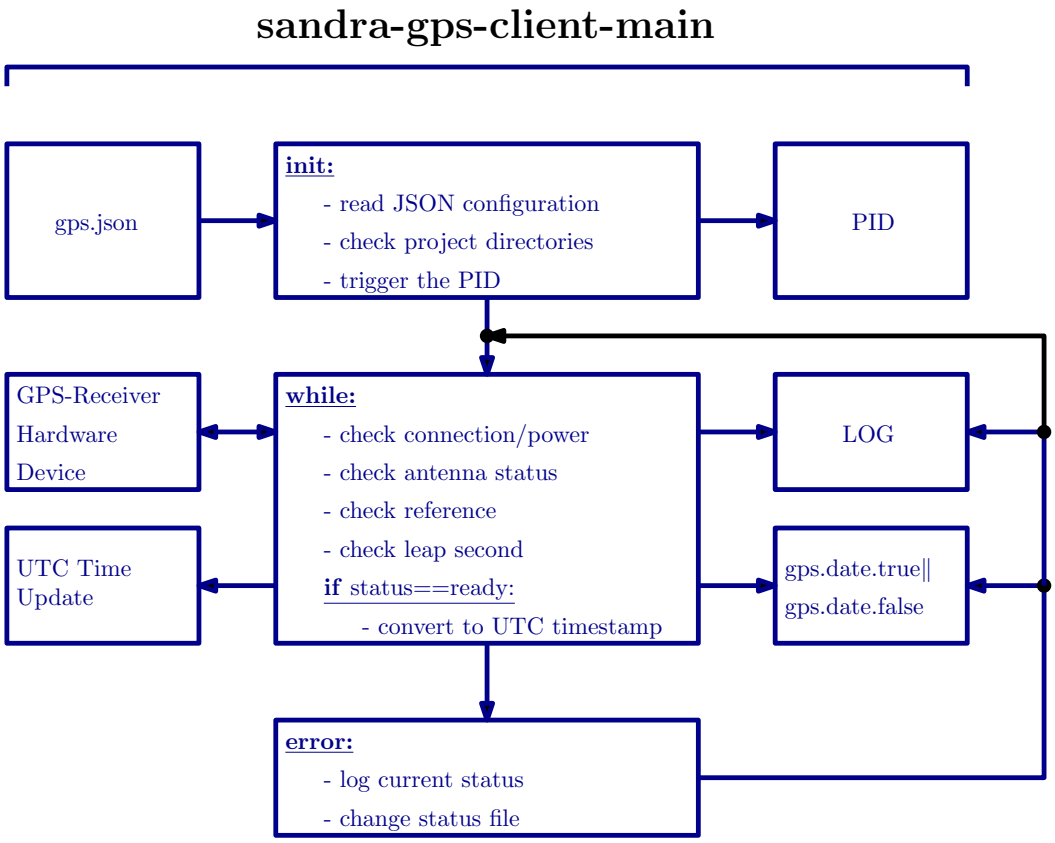


Figure 45: GPS Client Script Workflow

4.2.5 Network Time Protocol

Local NTP Server After the synchronization state is reached, the other computers in the cluster needs to be synchronized. This is very rudiment work for NTP. The computer which interfaces with the GPS receiver updates the date time and therefore this computer acts as the NTP server. This server is a configured *local NTP server* and broadcasts the timing information within the LAN ([14]). All the other nodes in the LAN are configured as *NTP clients* and updating their own date time.

NTP-Daemon This sounds very easy, but an important fact needs to take into account. The NTP that synchronizes a LAN is not synchronizing ad-hoc. The intercommunication between NTP server and clients is based on exchanging a lot of information and is doing an iterative statistically convergence to the true time of the server and this procedure takes a bit of time. This information of the NTP synchronization state of the server and the clients is provided by the NTP-daemon (NTPD) ([49]). The client script to manage the NTP synchronization provides a state information file of all the nodes in the computer cluster and if all of them are synchronized the transmitter and receiver client scripts are able to run and do the right job.

LAN Availability Another additive information of the NTP synchronization state check routine is the possibility to take care of the inter-connection of all nodes in the computer cluster. This directly gives information of the availability for all computers in the LAN configuration. The status of computer availability is also put in an information file.

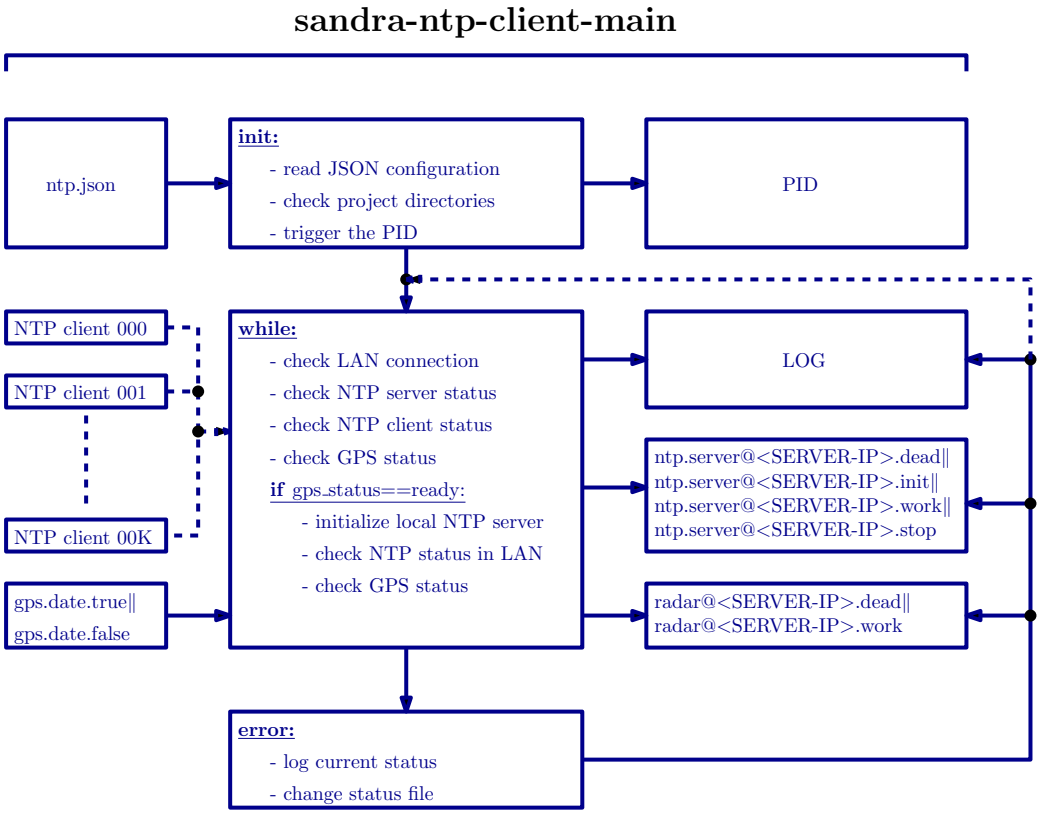


Figure 46: NTP Client Script Workflow

4.2.6 Transmitter/Receiver

The client scripts are the core of the whole radar system. These scripts are working as main execution script and take control of the specific system requirements like GPS and NTP synchronization or transmission and reception. Part of the scripts are a mapping of radar administrator configuration parameters to the specific started server scripts on a specific computer within the cluster. In Fig.47 the main structure of the client scripts is shown.

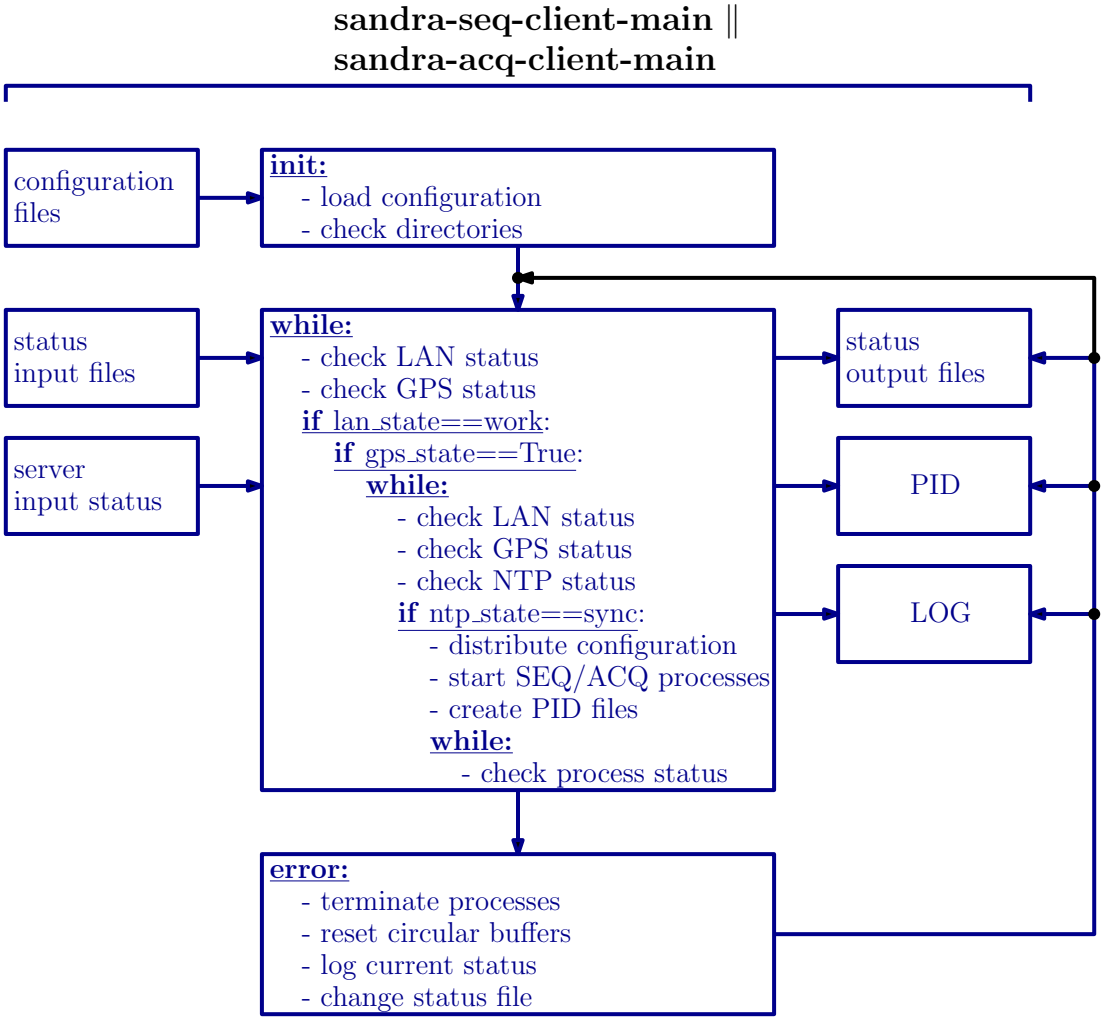


Figure 47: Transmitter/Receiver Client Script Workflow

4.2.7 Server Script Start

When the configuration parameters are separated and distributed, the server scripts will be executed remotely over SSH and the script will be started with the so-called no-hangup (NOHUP) signaling ([50]), which lets the process run in the background. The process-identifier (PID) of the started process is also send back from the execution server to the client server process and these PID's are logged in a state information file.

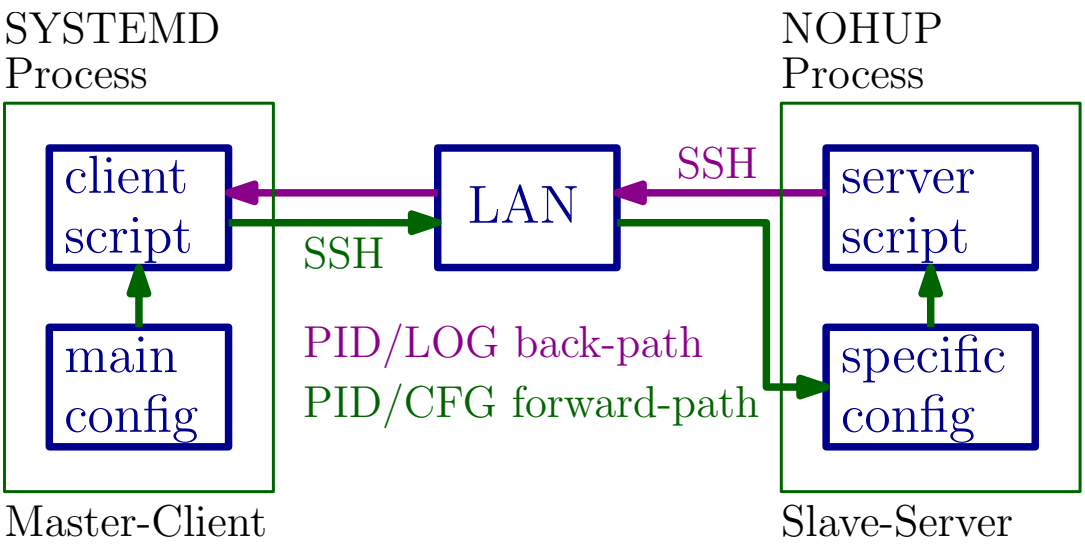


Figure 48: Remote Server Script Execution Workflow

After starting the server scripts from the clients, a mechanism to observe the PID's of the remote started server script is necessary. This is very easy by checking the PID is alive over the linux implemented *ps* command ([51]). If some process is terminated due to an error or from the operating system itself, the client triggers the missing process execution and starts to terminate/kill the other spawned server processes in the computer cluster. The PID control is not the only indicator for a failure operation of the radar system. While GPS and NTP servers are programmed to run continuously and report information of the states of the hardware they observe, it is necessary to trigger the states. When a failure state occurs, the client script is also able to terminate the spawned PID's.

4.3 Server Scripts

Over the last 3 years it figured out that a lot of software and hardware project where developed in a hierarchy structure. With this development concept, a clear interface between different software parts is available and also modularity of the software is given. Since hardware and software is working stringently together, the first thing is to write programs that are only focusing on the hardware that is used. This results in an easy way to extend the radar system with additional hardware without big problems. All the servers have the same implementation concept:

- parse configuration and system parameters as terminal arguments
- initialize and run the hardware
- if an error occurs, then terminate/kill the specific hardware process
- write the error ad-hoc to a file

This makes it easy to write software for other hardware and implement it within the whole project structure.

4.3.1 Transmitter

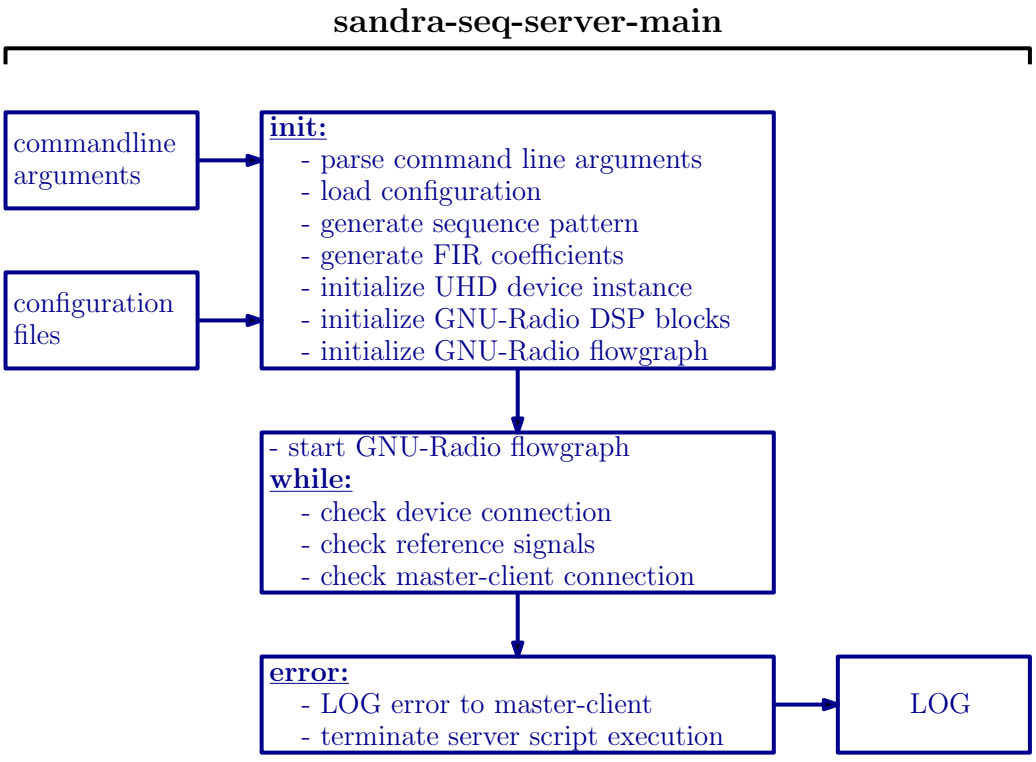


Figure 49: Transmitter Server Script Workflow

Error Handling The USRP-N200 is not only transferring the complex samples over the gigabit ethernet interface. The UHD provides also the status of the USRP-N200 and this status reports are readable by the UHD API functions included in the GNU-Radio libraries. Also a good error indicator is the object instantiation through the API itself. If for example the connection or a power error of the USRP-N200 occurs the runtime environment throws a runtime error and this implies whether the device is cut off from power-supply or the ethernet connection failures. While referencing the USRP-N200 with a 1PPS and a 10MHz reference signal the API provides information of the status from the device itself. This is very useful, if for example the reference is lost from the reference signal splitter or maybe a cable is not working properly. Then the script is ready to detect such errors and will terminate the execution and delivers an information state of the hardware.

4.3.2 Receiver

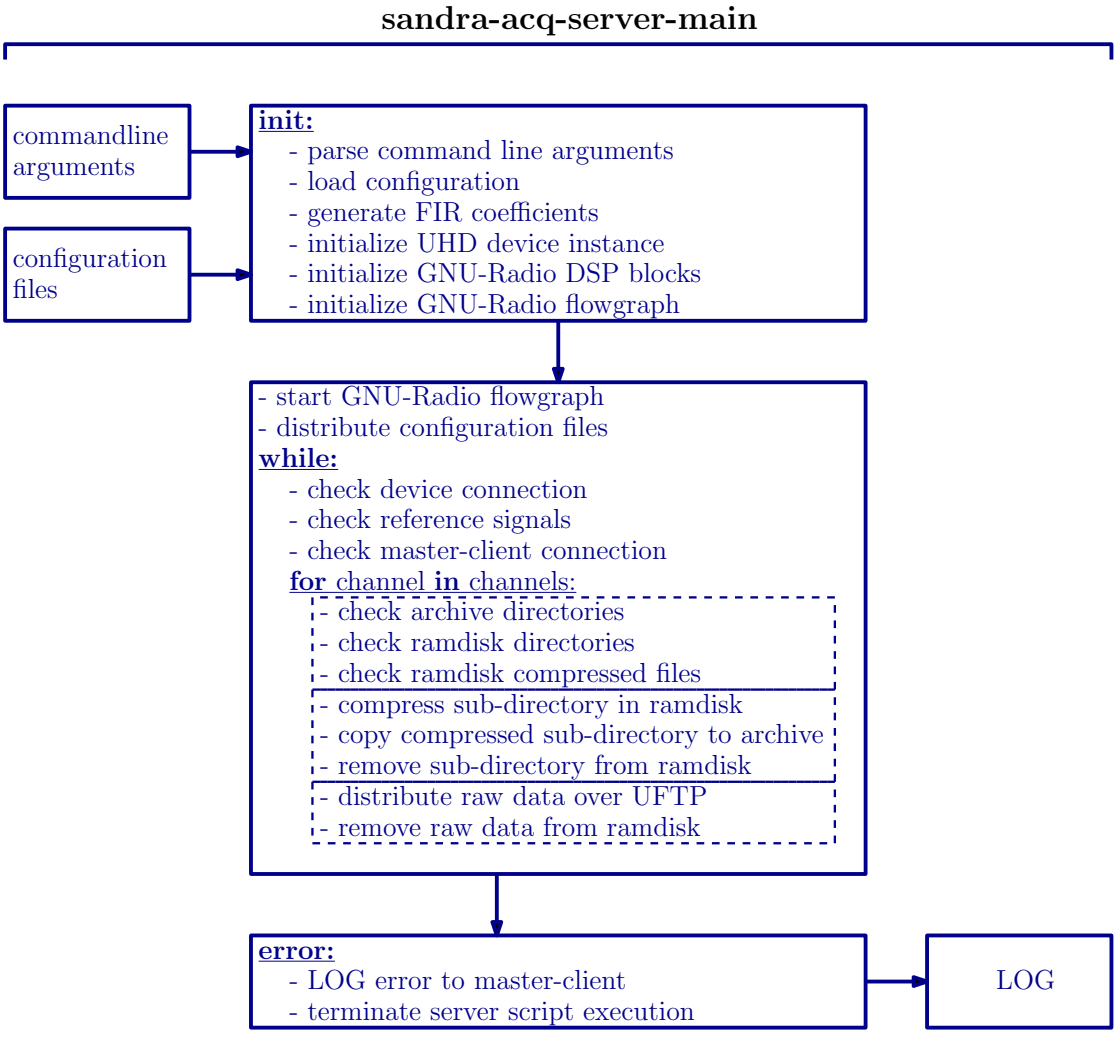


Figure 50: Receiver Server Script Workflow

Error Handling While using also the USRP-N200 on reception, the same error handling procedures as for transmitters are take into account. But one additional error is included related to the acquisition of the raw data with the Digital-RF library. This throws an error if a packet is lost through the GNU-Radio signal processing stream. This is also extremely important, because by imagine a packet is lost within the stream, a leackage of bytes is the result and this ends up in incoherence of the complex baseband signal raw data. When ever such errors occur, that impacts on kind of a synchronization distortion, a nearly ad-hoc reaction on the errors is realized. This is very helpful for a reduced effort of radar system monitoring and the system is able to fetch such states and the error handling is implemented in pure software, so the radar administrator is not forced anymore to search for possible errors on the whole receiver system.

4.3.3 Buffer

The buffer script is very easy in execution, since only using operating system implemented commands. There are no special needs for an error handling, since the script is only working as a state machine to observe the directories and clean up the circular buffer. After doing the cleaning, the script simply waits, until the buffer triggers back into the next rotation cycle. In Fig.51, the buffer workflow is illustrated.

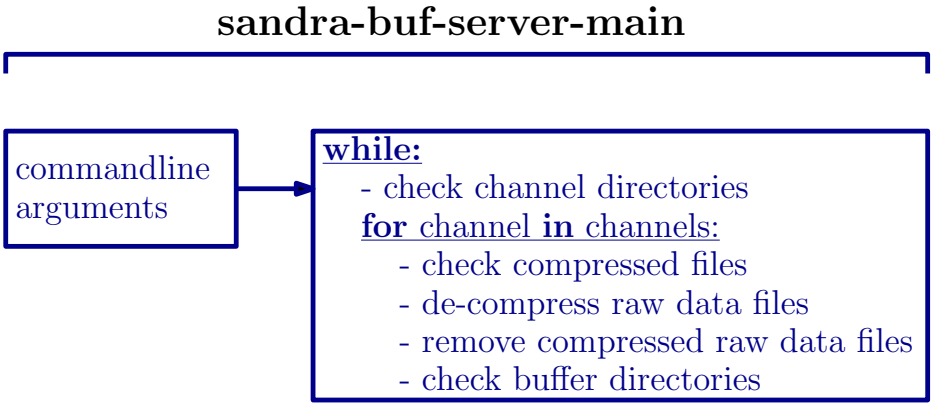


Figure 51: Buffer Server Script Workflow

4.3.4 Detection-Software Decoding-Extension

Script Template The core implementation of the detection and identification is already in operational mode. But the implementation has no decoding of pseudo-random sequences included. This is added during the working period and this was a non-trivial implementation since the decoding is a deconvolution of the acquired raw data. The mathematics are already described and the assumptions are also derived by [59]. But the main problem is the speed and the resources used by this delivered deconvolution script.

Since the assumption for the raw data deconvolution results in a *matrix multiplication*, the mathematics become very easy on a sheet of paper. But the real problem is the multi-dimensional array structure ([23]) of the radar raw data to cover range, time and channel as the main dimensions of the raw data array.

Source Code Bottlenecks The basic implementation and the recipe was given by a script from Juha Vierinen. The programming language used to do the heavy mathematics is Python with the modules *NumPy* ([23]) and *SciPy* ([38]). The bottleneck of the execution speed of the script was *for-looping* over the dimensions of the raw data array, this is not using the advantage of *NumPY*. NumPy is a *High-Level API* that is based on the *Low-Level* languages *Fortran* or *C++*. If using the NumPy-API right, a really optimzed and very powerful execution of array-manipulation can be achieved. The best practice is given by static multi-dimensional array manipulation ([25],[8]), which allows to operate *axis-based* on the raw data array.

Runtime Improvements The problem should now be reduced to the main deconvolution. The whole array is allocated as a 3-dimensional complex array with dimensions in *range*, *time* and channel. This is the generated output from the HDF5 reading routine of the detection routine. The first thing is to clean the raw data from radio interference and unwanted voltage spikes to reduce leakage effects of the deconvolution. When the raw data is clean, the deconvolution matrix multiplication is only done over the whole *range* axis of the array for each time-bin and each channel-bin. To make this operation very fast, the NumPy routine *numpy.einsum(...)* ([23]) is used. This routine is very well implemented in sense of doing *tensor-algebraic* calculations over multiple dimensions with full optimzed usage of resources. This improvement brings the deconvolution to a reasonable usage of one CPU for the parameters we'll use in the MMARIA-CW project. If the parameters changed to values where the task could not be done anymore by one CPU, the decoding provides a parallelization of the problem ([61],[28]) over the channel dimension and points the channel separated raw data array to different cores of the computer. This is known as single-instruction-multiple-data (SIMD) method in parallel computing.

5 Evaluation and Verification

5.1 Experimental Purpose

For the evaluation/verification of the developed software and hardware it is necessary to define an experiment to point out the new system features. The following list of points should explain in short, for what the system should be evaluated/verified and also the necessary steps to prepare the system for a real world application.

- cluster hardware setup
- cluster synchronization
- mobile network accessability
- transmitter/receiver configuration
- decoding techniques
- system calibration

In Fig.52, a testbed for the wanted purpose is illustrated as a block diagram and Fig.53 shows the real world testbed setup. The testbed includes a cluster of 3 computers. All computers are connected together over a LAN configuration, realized with an 8-port Layer-3 gigabit ethernet non-blocking switch. The master-client node in the cluster is the computer to configure the radar testbed setup. The computers in use provides two gigabit ethernet interfaces. One interface of the master-client is connected to the LAN network and the other interface is connected with another 5-port gigabit ethernet switch. The other two remaining computers are the slave-server computers in the LAN.

The last device in the LAN is the mobile router. This device is used to verify the system accessability through the internet. Therefore the router is configured with a static IP address and the router acts as the internet gateway for the domain-name-system (DNS) configuration. Only the master-client computer is first of all configured to be reachable through the internet.

The master-client is used to configure and execute a 5 channel receiver system and one channel transmitter independent from each other. Another main task of the master-client is the synchronization of the cluster by interfacing and communicating with the GPS receiver. The last point is the acquisition of the data. All the 5 channels are received, processed and stored on ramdisk as compressed data.

To verify also the cluster capability to process the data at a receiver station with different links directly, 2 slave-servers are configured to aquire the received data over UFTP and do the further detection signal processing. In this case there is nothing to detect, since using a testbed configuration. But the detection software is able to run in non-detection mode and be able to provide also figures of the processed range-time-intensity (RTI), power-spectral-density (PSD) and noise levels, including long-time overview plots to verify the decoding under realtime conditions. The processing is evaluated with 4 independent decoding processes realized by one pulsed compression sequence decoding and 3 pseudo-random deconvolution decodings. This defenetly shows the advantage of the system to parallel process the data at a remote station.

This experiment is also a good point to calibrate the radar system under real world conditions. Therefore a range-offset and phase-difference calibration is performed. For the range calibration an additional oscilloscope is included in the testbed. The reason is the measurement of the transmitter delay in comparison to a 1PPS signal but this is explained later in this section.

Figure 52: Real World Testbed Setup

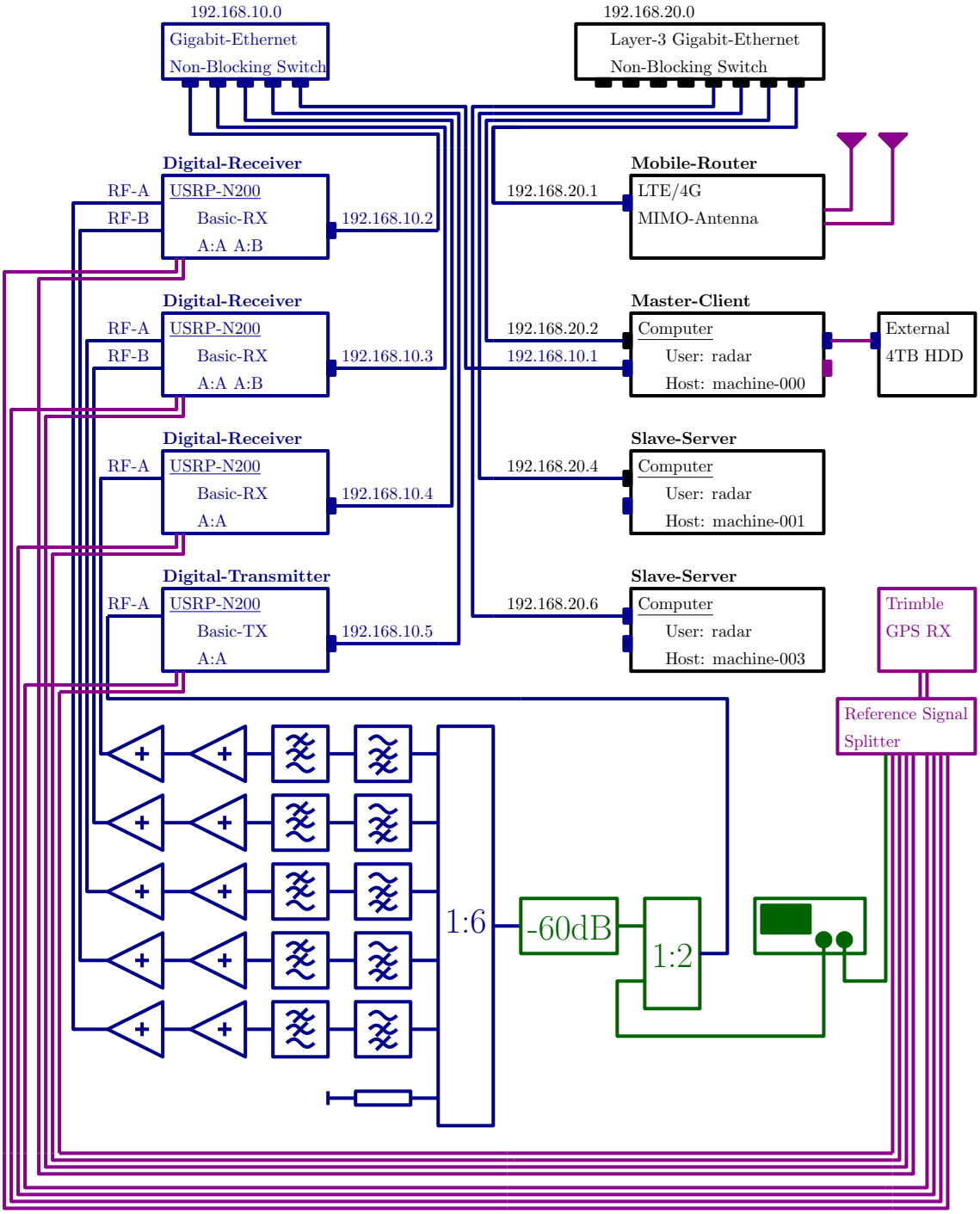


Figure 53: Testbed Setup Block Diagram

5.2 Experimental Testbed

The radar is now ready to use in a verification application. This application should be as easy as possible and should provide as much information and functionality proofs. Therefore a special customized test signal is designed to provide the required verification results.

5.2.1 Test Signal Design

Test Signal Description This signal is designed to simulate a real world multistatic radar application and should cover the information of different radar transmitter stations operated with different codes. Under this conditions some special signal modifications are taken into account. To cover the right decoding and signal separation, the complex baseband signal is admitted with a *range delay*, a *timing gap*, a *doppler-shift* and a complex additive-white-gaussian-noise (AWGN).

Test Signal Parameter After the distributed rawdata is processed in the separated detection processes, the ability to verify the correct processing is given. These specific parameters are listed in Fig.54. These parameters are *unique* and there is no possibility to do some misinterpretation which is the most important part for the verification.

Type	Range / km	Delay / km	Timegap / s	Doppler / Hz
Barker-7	600	0	0...5	-15
Pseudo-471	3000	150	5...10	-5
Pseudo-329	3000	300	10...15	+5
Pseudo-100	3000	450	15...20	+15

Figure 54: Table of Testbed Parameters

Test Signal Synthesis A block diagram on how the test signal is generated is illustrated in Fig.55 and the sequence is stored as a binary file. The block diagram also includes the *verification* processing chain of the test signal correctness. This is important to more or less playing to find the right parameters for the signal generation because all the signal information is present in the same sequence at the same time. This is the *worst-case* scenario of the radar multistatic application.

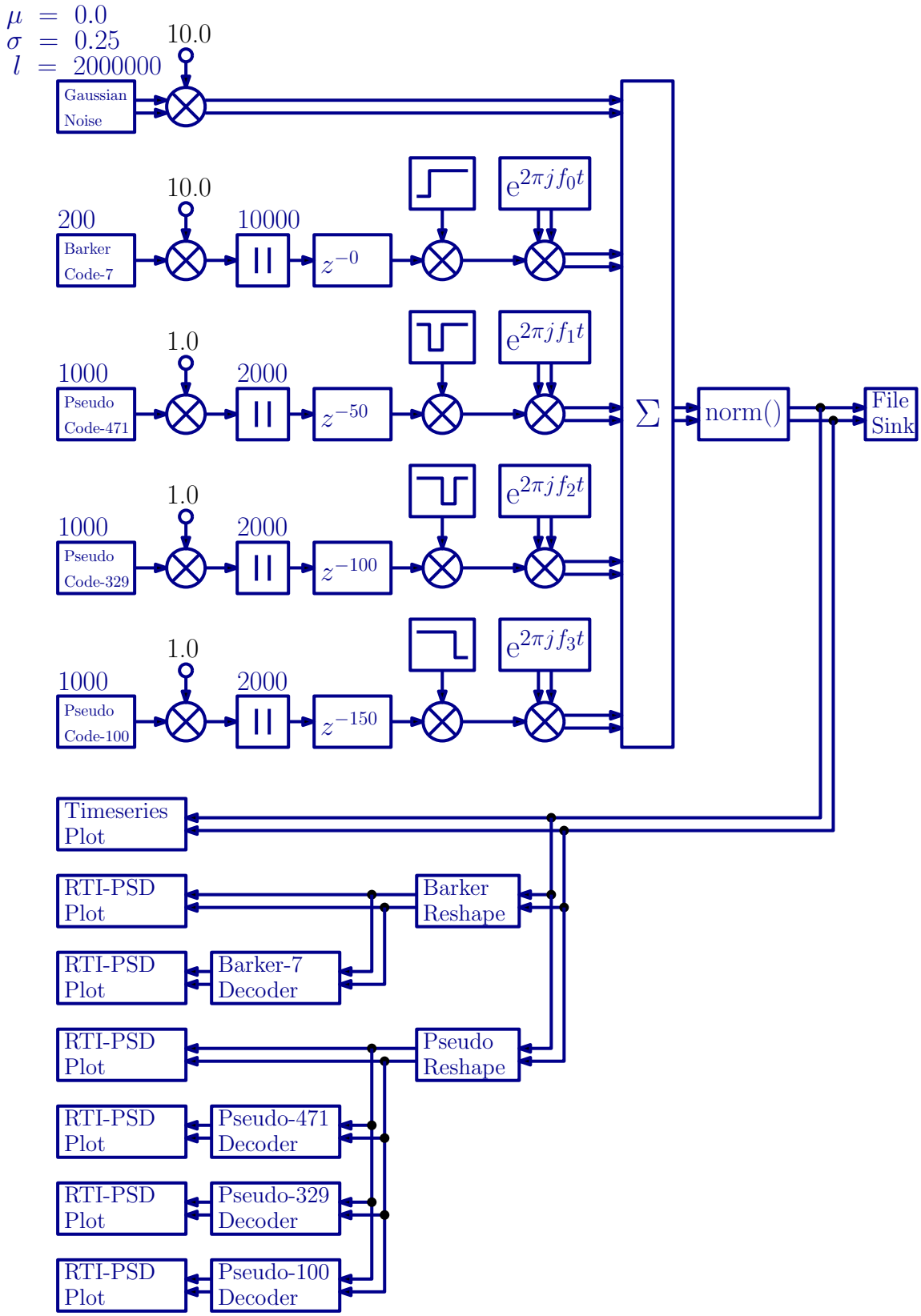


Figure 55: Testbed Signal Generator Block Diagram

Test Signal Baseband After program execution the complex baseband test signal is illustrated in Fig.56 in *real* and *imaginary* part. As seen in the figure some spikes occur in the time domain, which implies that the signals are not totally covered in the noise. These spikes are related to the doppler shifted barker-7 code and the reason of the higher amplitude is the decoding coverage of the signal, since the cross-correlation of all the signal mixed together is not perfect and orthogonal at all.

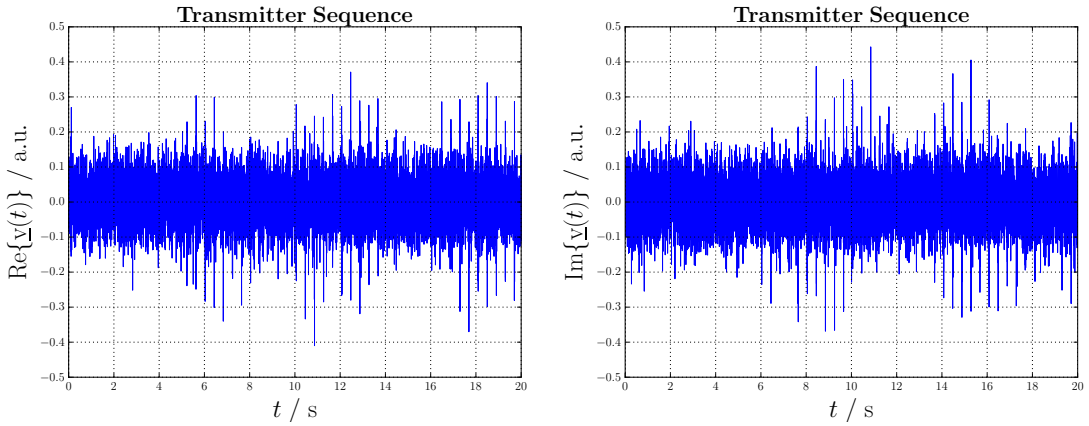


Figure 56: Complex Baseband Testbed Signal

5.2.2 Transmitter Signal Verification

Barker-7 Code To proof the correct test signal design, a first verification test is done, to check the parameters and the decoding properties. In Fig.57, the undecoded reshaped raw data time series for the barker-7 sequence is illustrated as a RTI. The undecoded barker sequence is seen at the bottom of the RTI plot with the required time gap and the PSD shows also the spectral components of the test signal. Notice the occurance of the spectral properties of the pseudo random sequences covered in the noise.

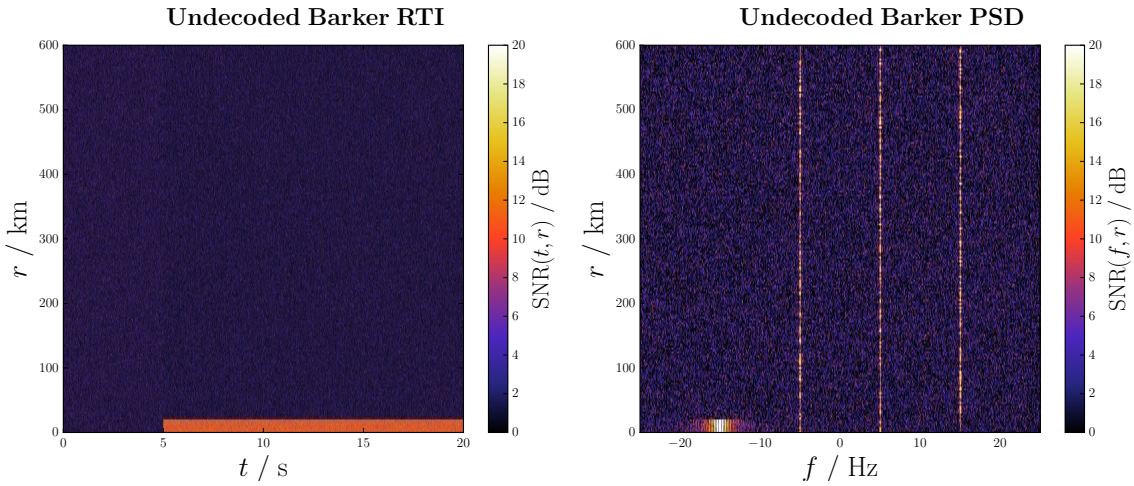


Figure 57: Undecoded Barker-7 RTI

After correlation decoding the barker-7 sequence, the RTI and PSD shows the expected *peak-to-sidelobe* sequence and the spectral properties are also present in the PSD. The timegap also provides a good visibility of the decoded raw data. The expected decoding and radar properties are marked within the red boxes, to extract the wanted information out of the RTI and PSD pictures. The comparison of the undecoded test signal with the decoded test signal RTI shows that the decoded peak occurs with a delay, related to barker-7 code length. This is shown in Fig.58 for verification.

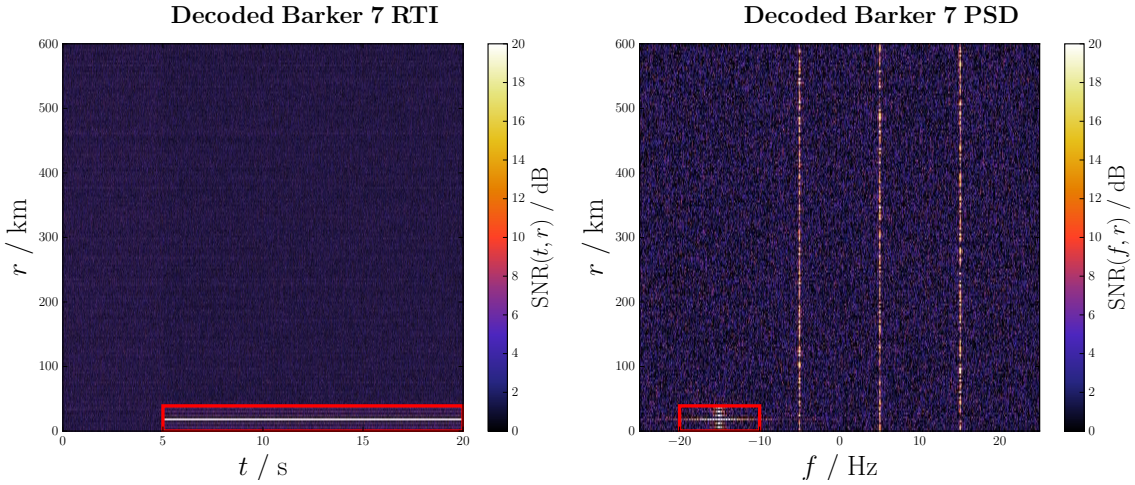


Figure 58: Decoded Barker-7 Testbed Signal RTI

Pseudo-Random Codes The next verification test is to deconvolve the covered pseudo random sequences within the noise. Before doing deconvolution, the raw data time series is reshaped in another order, to do the decoding of the sequences. Since the barker-7 code is covered 5 times within one pseudo random sequence, the undecoded RTI and PSD shows also 5 times the occurance of the barker-7 code. In Fig.59, these plots are shown.

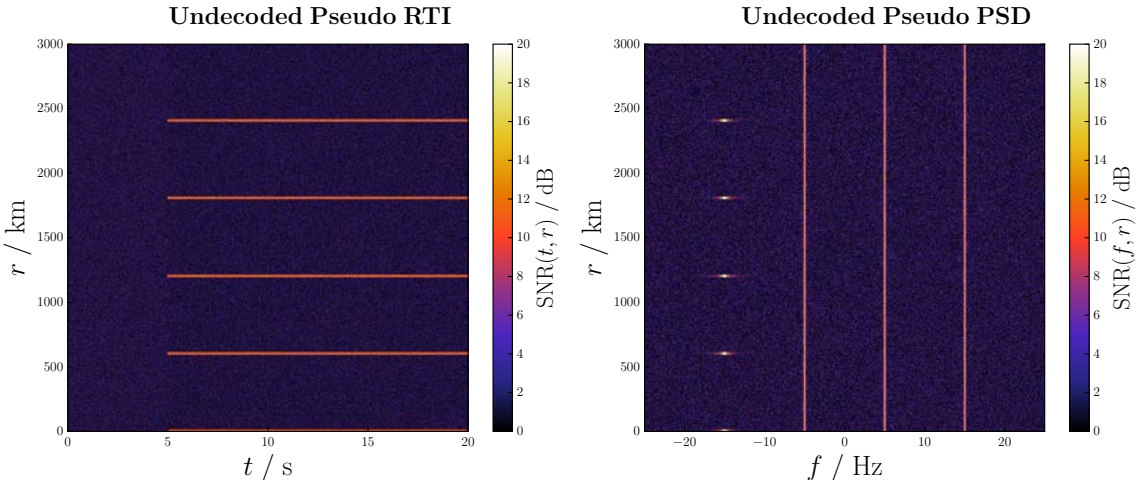


Figure 59: Undecoded Pseudo Random Testbed Signal RTI

Now the deconvolution on the same raw data is done for the 3 different pseudo random seuqences and the original range bins are reduced from 1000 bins to 200 bins. All the decoded pseudo random sequences showing the expected parameter configurations in the RTI and PSD. These are also marked as red boxes in the RTI and PSD plots and Fig.60 illustrates the verified parameters after deconvolution.

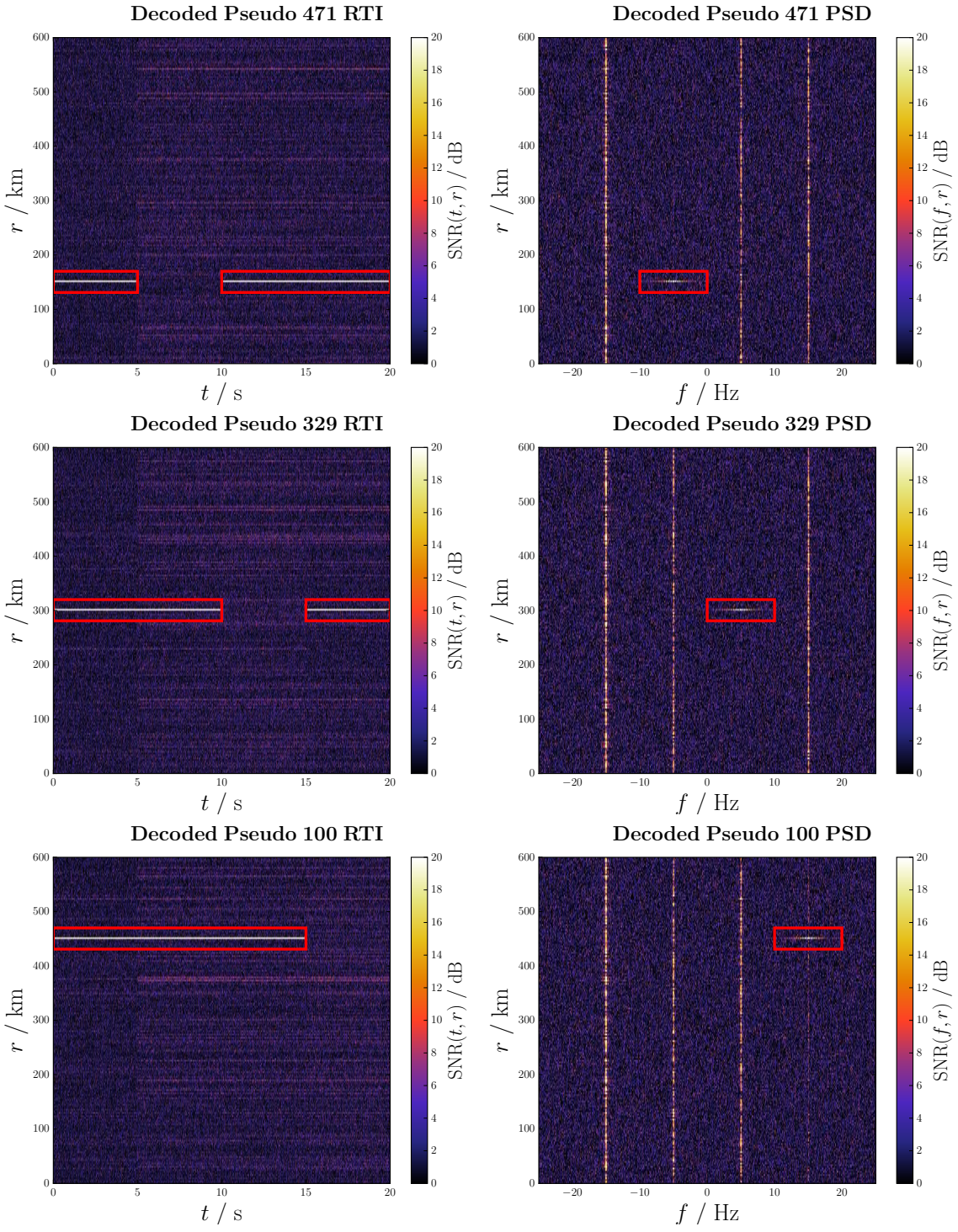


Figure 60: Decoded Pseudo Random Testbed Signal RTI

5.2.3 Receiver Signal Verification

The test signal is stored within a binary file and the *custom-sequence* feature of the SanDRA transmitter is used to convert the test signal from baseband to 32.55MHz and the signal is attenuated and split into the 5 receiver channels. Since the transmitter and receiver are synchronized with the same GPS reference and operating on the same computer cluster, the decoded signals should appear in the RTI and PSD generated by the distributed executed detection software. The graphical results from the detection software are illustrated in the following listed figures.

Barker-7 Code For the barker-7 code, the received signal after decoding is illustrated in Fig. 61 The parameters from the test signal are as expected and occurs in the decoded receiver raw data the same. The only difference is the delay in the range axis, but this is explained later in the calibration section.

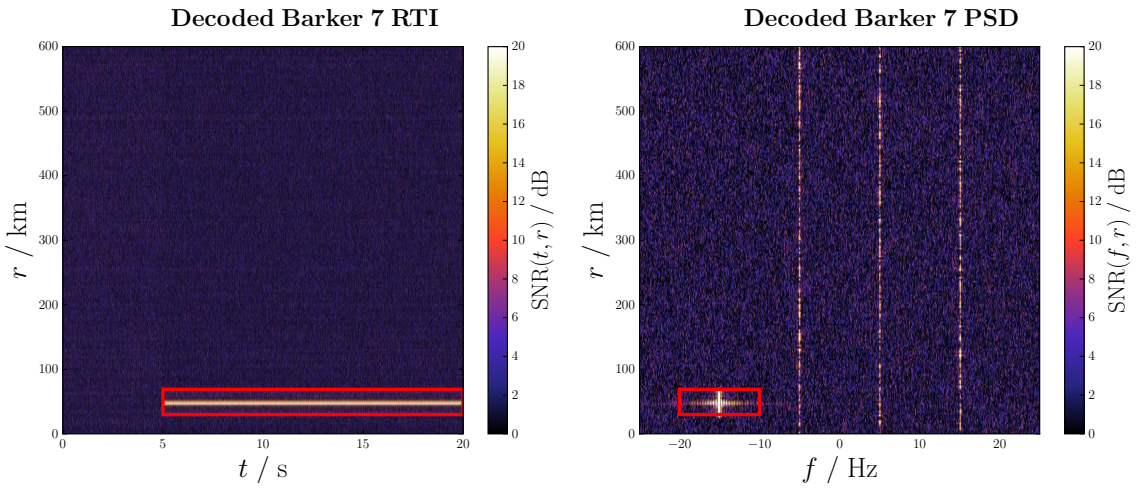


Figure 61: Decoded Barker-7 Receiver Signal RTI

Pseudo-Random Codes The pseudo random sequences are also as expected and also a range delay is seen in the decoded raw data. Since the detection software works with integration over all channels, more sensitivity is gained and the PSD shows a weak spectral line for the specific doppler frequency in the range gates because the doppler frequency offset is applied as a continuous oscillation in the test signal. Fig.62 shows the results from the detection sofware.

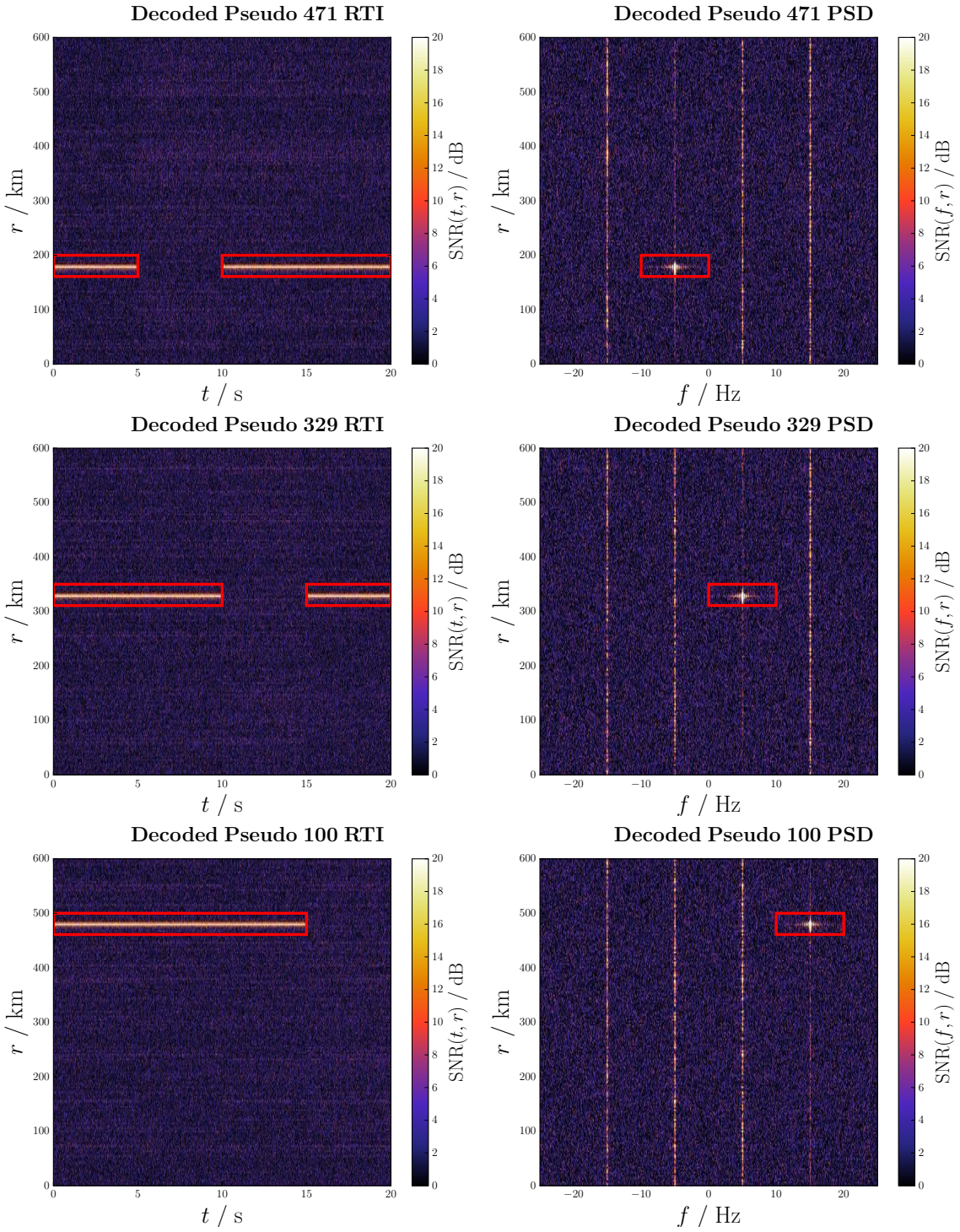


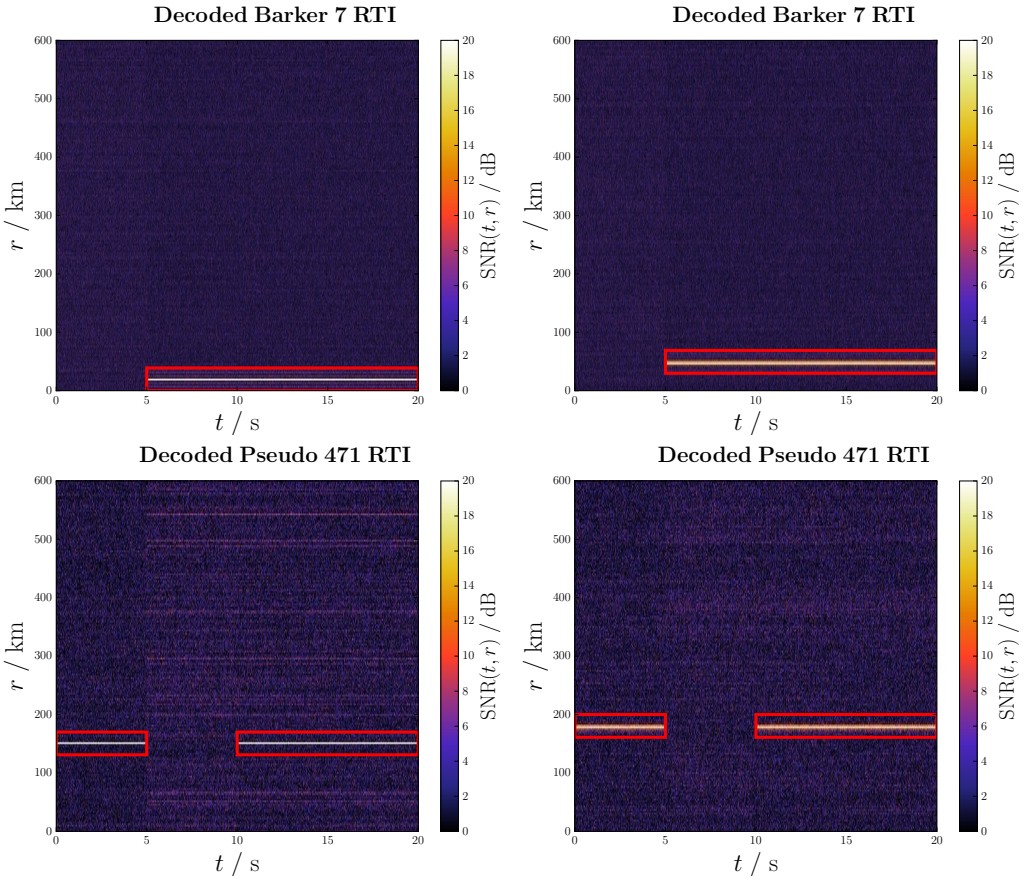
Figure 62: Decoded Pseudo Random Receiver Signal RTI

5.2.4 Calibration

Transmitter Delay During the execution of the test signal verification, an oscilloscope is used to trigger a 1PPS signal and the synthesized test signal, to measure the time delay implied from the transmitter signal processing chain time delays. This value is important and needs to be converted into an integer sample value. The conversion to the integer sample delay is done by the following formula:

$$\begin{aligned}\Delta N_{\mathrm{TX}} &= \lfloor \Delta\tau_{\mathrm{TX}}\ f_{\mathrm{S,RX}} \rfloor \\ \Delta N_{\mathrm{TX}} &= \lfloor 65\mu\mathrm{s}\ 100\mathrm{kHz} \rfloor \\ \Delta N_{\mathrm{TX}} &= 6\end{aligned}$$

Total Delay The decoded raw data in the RTI and PSD from the detection software, seen in Fig.63, shows also a range delay. This delay is a composition of the *transmitter* and *receiver* time delay. In a more general case a third value is included in the radar time delay, seen in the receiver. This value is the nearly direct ground wave propagation between transmitter and receiver. Since feeding the transmitter signal into the receiver directly and assuming the operation of the analog hardware devices are in the linear parameter domain, where no group delay implies an occuring wave propagation of the test signal, this third time delay value is cancelled in the calculation for the receiver time delay.



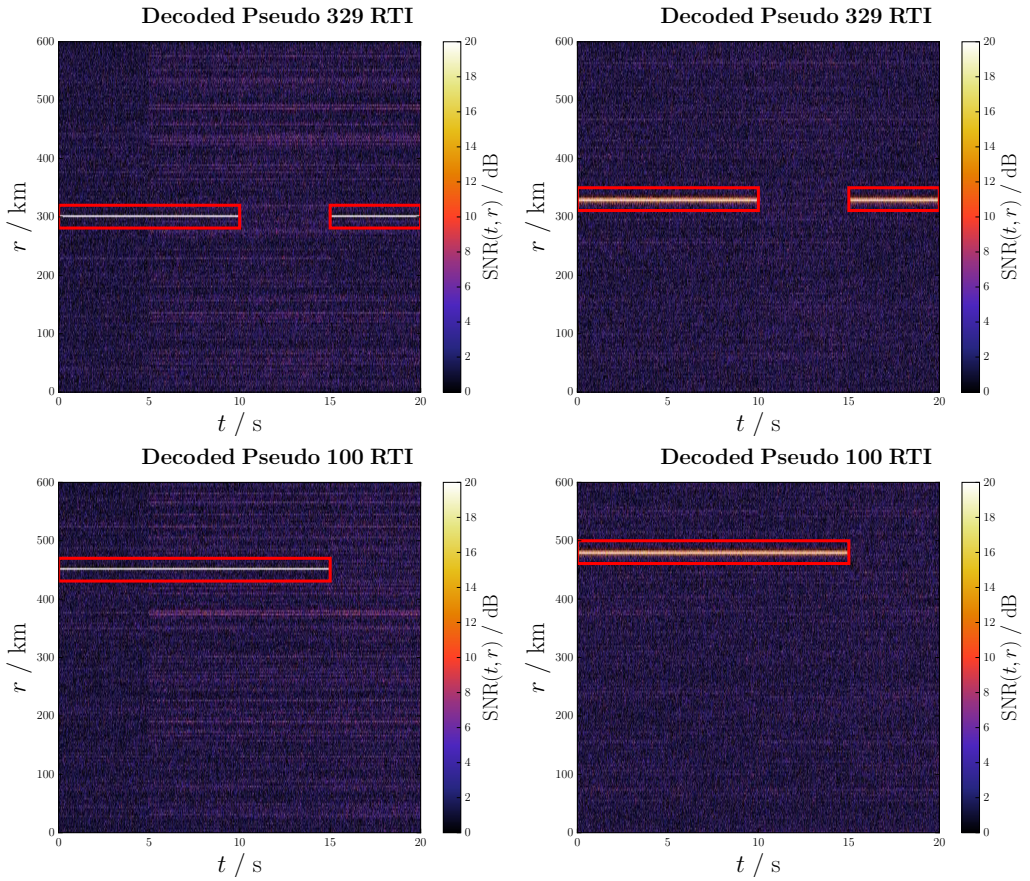


Figure 63: Decoded Testbed and Receiver RTI Comparison

In Fig.63, one additional feature of the extended detection software is the implementation of the pre-cleaning of the raw data before decoding with the pseudo random deconvolution matrix. The left column shows the uncleaned decoded test signal raw data of the receiver and the right column shows the cleaned decoded raw data from the detection software.

Receiver Delay The starting point is the extraction of the integer time delay from the detection software produced RTI. This is simply the integer value from the array, where the maximum of the decoded raw data arise. The formula to calculate the receiver time delay is:

$$\Delta N_{\mathrm{RTI}} = \Delta N_{\mathrm{TX}} + \cancelto{0}{\Delta N_{\mathrm{GW}}} + \Delta N_{\mathrm{RX}}$$
$$\Delta N_{\mathrm{RX}} = \Delta N_{\mathrm{RTI}} - \Delta N_{\mathrm{TX}}$$

$$\Delta \tau_{\mathrm{RX}} = \Delta N_{\mathrm{RX}}\ T_{\mathrm{S,RX}}$$
$$\Delta \tau_{\mathrm{RX}} = (\Delta N_{\mathrm{RTI}} - \Delta N_{\mathrm{TX}})\ T_{\mathrm{S,RX}}$$

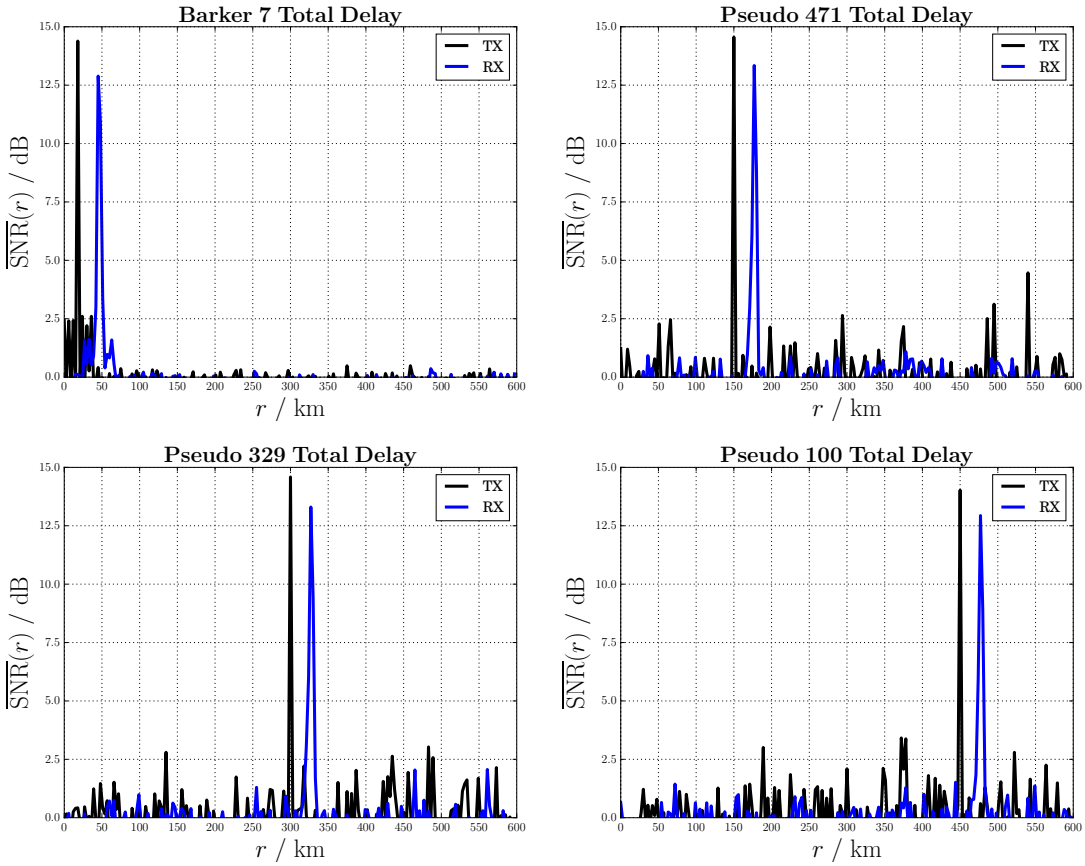


Figure 64: Total Range Delay of Transmitter and Receiver Signal

Fig.64 shows the mean range dependency of the SNR for all time bins of the channel mean RTI. The black curve is the original transmitter decoded signal and the blue curve is the receiver decoded signal. The difference ΔN_{RTI} of both peaks is the total delay between transmitter and receiver. Now all necessary parameters for the range calibration are known and the values could be determined.

Range Calibration The results of the range calibration are shown in Fig.65 The results are for all the differetn sequences the same, because the same parameters for the bits and center frequency are used.

$\Delta\tau_{\mathrm{TX}}$ / μs	ΔN_{TX}	$\mathbf{\Delta r}_{\mathrm{TX}}$ / km	ΔN_{RTI}	ΔN_{RX}	$\Delta\tau_{\mathrm{RX}}$ / μs	$\mathbf{\Delta r}_{\mathrm{RX}}$ / km
65	6	**18**	9	3	30	**9**

Figure 65: Table of Range Delay Results

Phase Calibration For phase calibration, the maximum peak for each time bin of the decoded raw data is of interest. Since the radar is coherent in time, the array index for the maximum peak stays the same for all time bins and the array could be sliced for all channels. Now we need to convert the frequency shift to the zero baseband frequency by applying a simple multiplication with a complex harmonic of inverted frequency.

Then the angle of the complex array is calculated for each time bin over all channels and the mean value over the time bins gives the phase value for all channels. The last step is to convert the radiant angle to degrees and subtract the last channel angle from all angle values, to convert to the wanted phase differences. This is known as *relative* phase calibration. The results are shown in Fig.66, and the values are quasi equal for all the different codes. This is clear, since using the same frequency and same hardware for raw data acquisition. The last row in the table shows the mean phase differences of the calibration. For the last channel all values are equal to zero and not shown in the table.

	$(\varphi_1 - \varphi_5)$ / °	$(\varphi_2 - \varphi_5)$ / °	$(\varphi_3 - \varphi_5)$ / °	$(\varphi_4 - \varphi_5)$ / °
Barker 7	−0.573	−9.616	+1.030	−5.253
Pseudo 471	−0.567	−9.744	+1.067	−5.218
Pseudo 329	−0.546	−9.778	+1.025	−5.238
Pseudo 100	−0.554	−9.743	+1.047	−5.225
Mean Values	−0.561	−9.720	+1.042	−5.234

Figure 66: Table of Phase Difference Results

5.3 Experimental Field Campaign

In the previous section, the realtime decoding and data distribution of the same raw data is verified by a testbed experiment. Now a real world experiment is in focus to proof also the accessability through the internet and configuration. For this experiment current installed hardware is used. One problem is the fact, that the planned radar network for the MMARIA-CW project is not existing at all. Therefore a special campaign is designed to verify the functionality of the system.

5.3.1 Field Campaign Design

For the field campaign, a pulse transmitter in Juliusruh is used. This transmitter operates with a barker-7 sequence. The installed testbed receiver is further used in Kuehlungsborn and plugged into the antenna interface, to receive and acuire data from a *5 Yagi-Antenna Cross-Dipole Jones-Configuration* at 32.55MHz center frequency. To addtionally have the possibility to use pseudo random codes, the prototype pseudo random transmitter in the metal box is used and installed in Kröpelin, very close to Kühlungsborn. Therefore the pseudo random transmitter is operating at a very low power of only 20W. For verification of the arbitrary station access, the configuration and operation monitoring is done in Wismar University.

Function	Location	Operation
Administrator	23966, Wismar	Configuration
Transmitter	18556, Juliusruh	Barker-7
Transmitter	18236, Kröpelin	Pseudo-471/329/100
Receiver	18225, Kühlungsborn	Acquire/Process

Figure 67: Field Campaign Setup Parameter

For the experiment a sweep of the pseudo random seeds is done in a time multiplexing session, because only one transmitter exists at this moment. The receiver is processing for all the configured sequences at the same time in parallel. The transmission sequence timeline is shown in Fig.68

Sequence	Start Timestamp	Stop Timestamp	System
Barker-7	< 2017-07-12 15:00:00 UTC	> 2017-07-12 16:00:00 UTC	Genesis
Pseudo-471	2017-07-12 15:00:00 UTC	2017-07-12 15:20:00 UTC	SanDRA
Pseudo-329	2017-07-12 15:20:00 UTC	2017-07-12 15:40:00 UTC	SanDRA
Pseudo-100	2017-07-12 15:40:00 UTC	2017-07-12 16:00:00 UTC	SanDRA

Figure 68: Field Campaign Timeline

5.3.2 Field Campaign Verification

In Fig.69, the block diagram for the field campaign is illustrated. Only the used computer and network hardware is shown to see the access flow to the stations and also the computers are marked with the executed scripts, to see which services running on the distributed computers. The results of the parallel processing are listed in the paragraphs 5.3.2, 5.3.2 and 5.3.2.

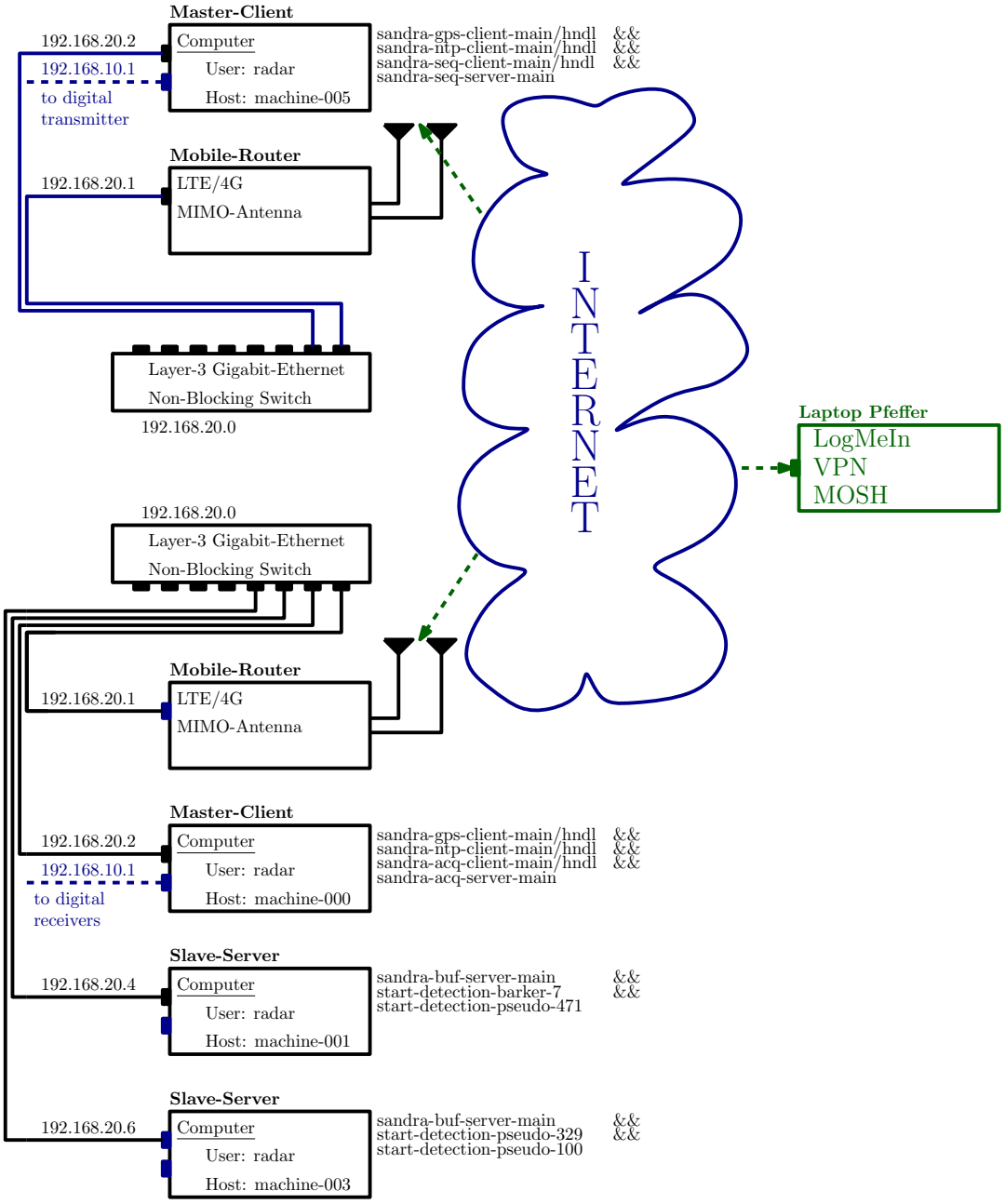


Figure 69: Field Campaign Setup Block Diagram

Pseudo Random Seed 471 Results

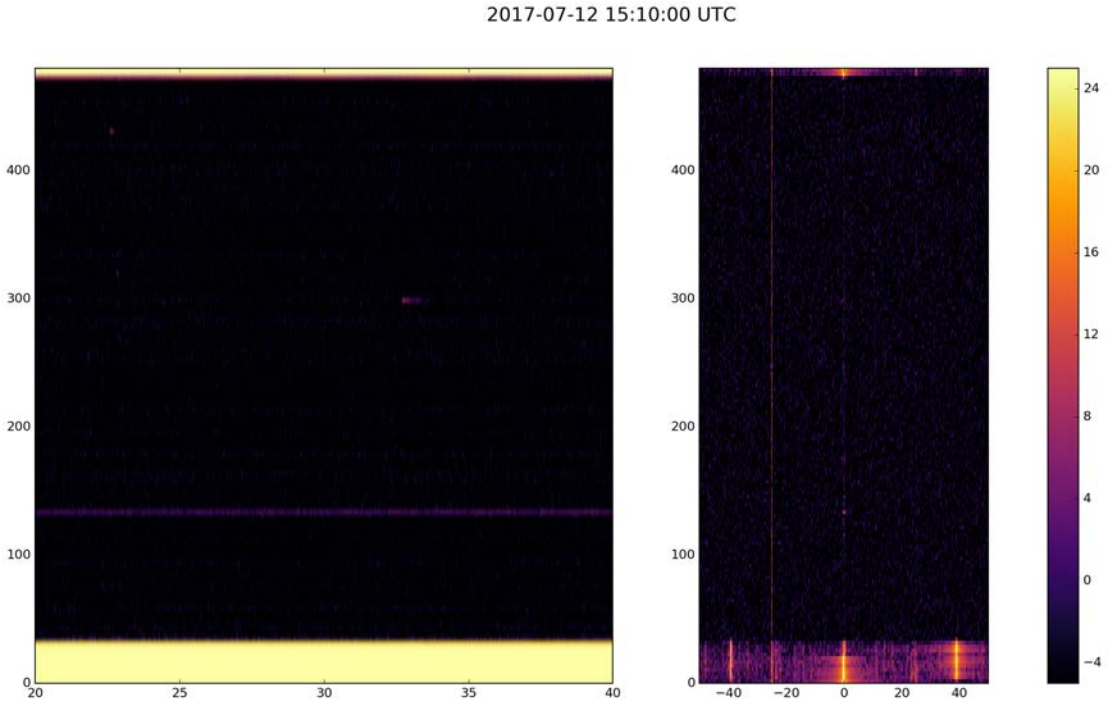


Figure 70: Barker-7 Decoded RTI and PSD for Seed-471

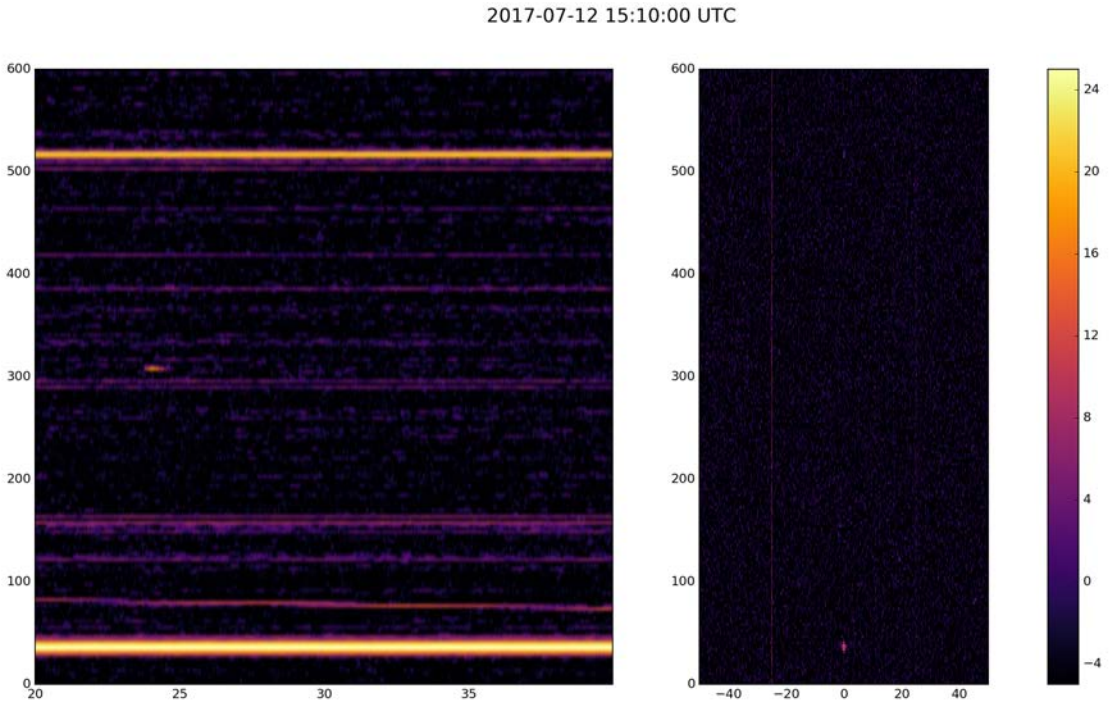


Figure 71: Pseudo-471 Decoded RTI and PSD for Seed-471

Pseudo Random Seed 329 Results

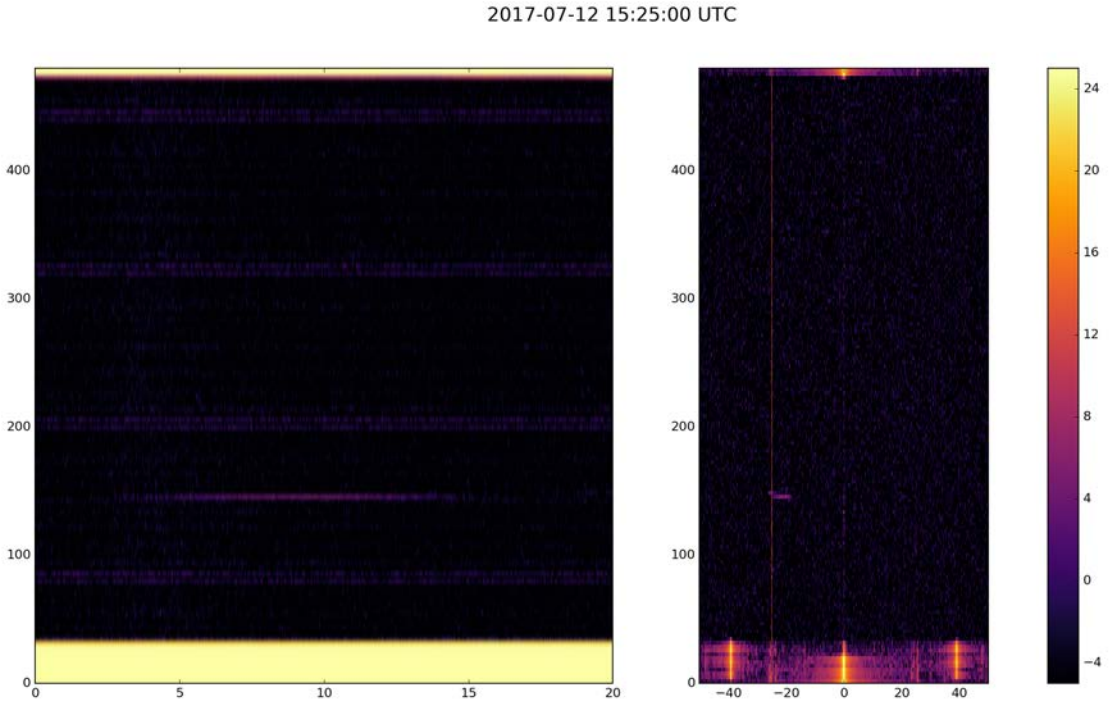


Figure 72: Barker-7 Decoded RTI and PSD for Seed-329

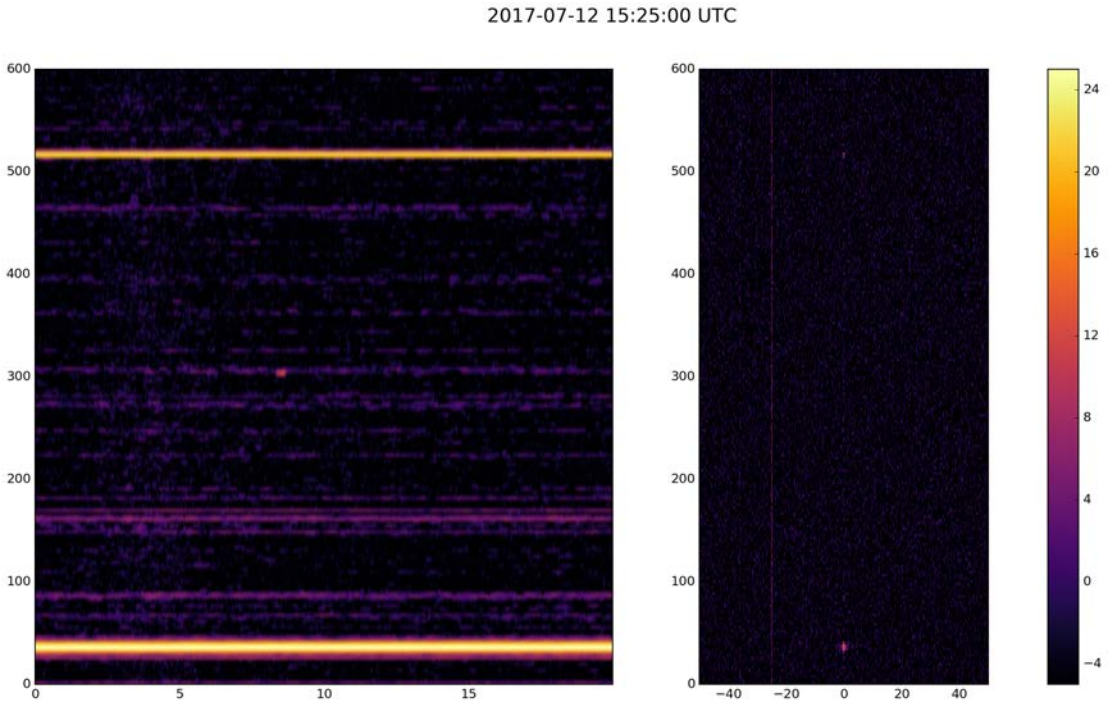


Figure 73: Pseudo-329 Decoded RTI and PSD for Seed-329

Pseudo Random Seed 100 Results

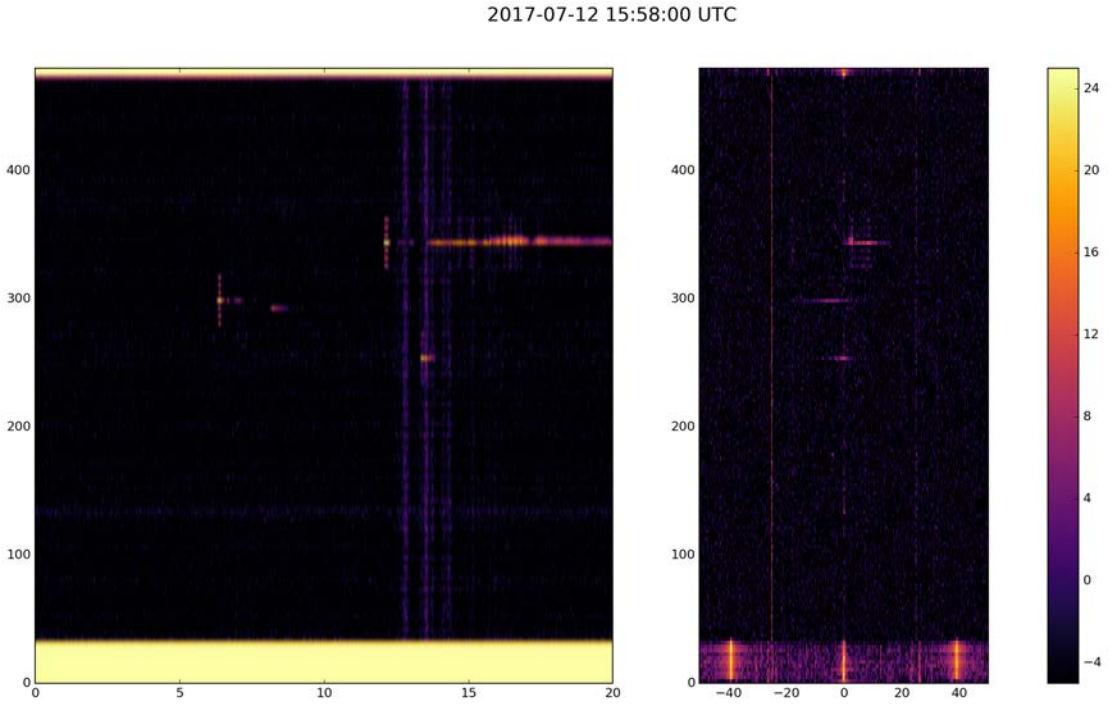


Figure 74: Barker-7 Decoded RTI and PSD for Seed-100

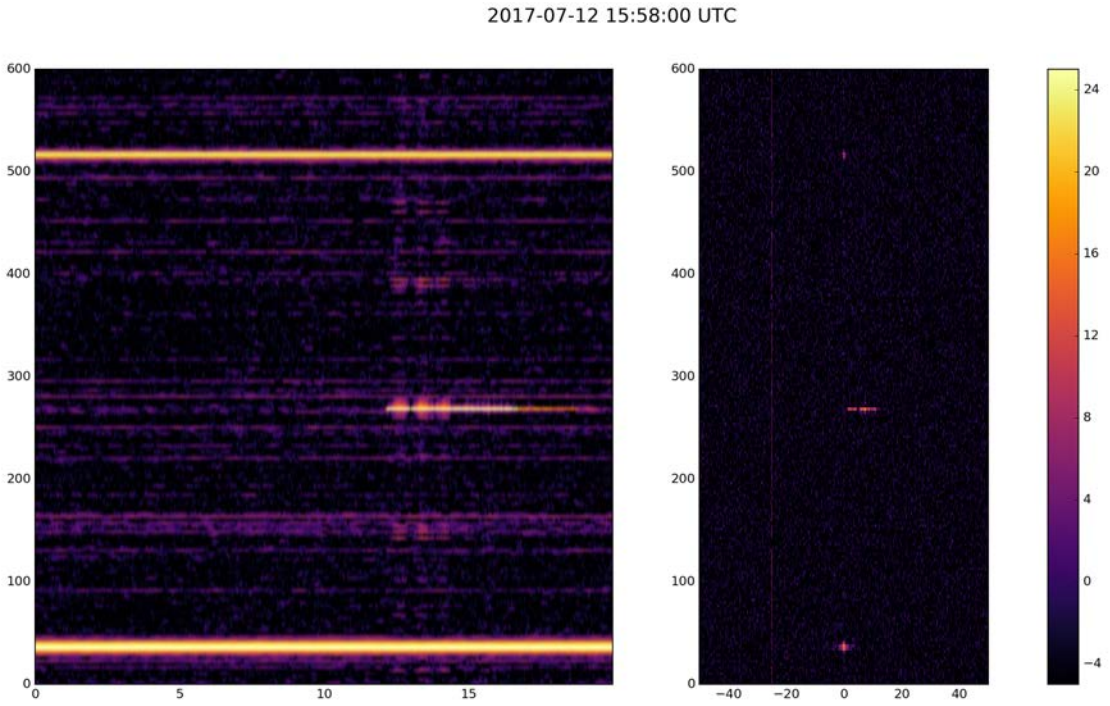


Figure 75: Pseudo-100 Decoded RTI and PSD for Seed-100

6 Conclusion

6.1 Summary

In this thesis, the main aspects of the developed software are verified. The data distribution over multi-casting is working as expected and the logging information of the current observed errors are triggered and forcing the system to stop the processes and restarting again. This works for the transmitter and receiver as well.

The connection to the remote computers with mobile networking is working well. Also the configuration of transmitter and receiver is working and with the usage of systemd, the process monitoring is as expected. The system also restarts after a power failure and is also reachable over internet after a reboot. These essential features allows to do further system observations and improvements in a multi-static radar network.

6.2 Future Work

Radar Network Installations During the working period of this thesis, the necessary steps for the radar network installations were done. This includes the visiting of all the 6 sites for transmission and proof the mobile connectivity. Also the statutory provisions are fullfilled and the frequency and location licenses are owned by IAP and certified by *Bundes-Netzagentur* to built up the stations in northern germany. Currently the progress is a Deutsche-Industrie-Norm (DIN) conform circuitry case prototype development for the transmitters. The further weeks will focus on the progress of the transmitter installations, to be able to start the radar network in an operational state.

System Validation When the radar network is in operation, the engineering of the system needs to be validated with the produced data. The processed data needs to be analyzed and bring some scientifical results. This process is also iterative, because the opinions and meanings of the scientists are input information to improve and modify some system specifications.

Digital Pre-Distortion As mentioned in this thesis, the used amplifiers are not perfect in sense of linearity. Therefore, digital-pre-distortion (DPD) is a technique to compensate the non-linear characterstic curve of the amplifiers, by manipulate the amplifier input signal with the current output signal information. To do DPD, a reasonable bandwidth is necessary, to cover the spectral non-linearities and avoid also aliasing effects for the DPD.

List of References

Book References

[6] A. Collette. *Python and HDF5*. O'Reilly, 2013.

[8] F. Doglio. *Mastering Python High Performance*. Packt Publishing, September 2015.

[10] H. Erlenkoetter. *C++ - Objektorientiertes Programmieren von Anfang an*. Rowohlt Taschenbuch Verlag, August 2010.

[11] H. Erlenkoetter. *C - Programmieren von Anfang an*. Rowohlt Taschenbuch Verlag, August 2010.

[16] C. Giridhar. *Learning Python Design Patterns, 2nd Edition*. Packt Publishing, Februar 2016.

[21] J. G. Proakis , D. K. Manolakis. *Digital Signal Processing*. Pearson, Februar 2014.

[25] M. Gorelick , I. Ozsvald. *High Performance Python*. O'Reilly, August 2014.

[26] J. Palach. *Parallel Programming with Python*. Pack Publishing, Juni 2014.

[28] F. Pierfederici. *Distributed Computing with Python*. Packt Publishing, April 2016.

[30] Luciano Ramalho. *Fluent Python*. O'Reilly, Juli 2015.

[31] D. Phillips , F. Romano , M. Czygan , R. Layton , S. Raschka. *Python: Real World Data Science*. O'Reilly, June 2016.

[35] J. G. Proakis , M. Salehi. *Digital Communications*. McGraw Hill, September 2010.

[36] P. Schnabel. *Kommunikationstechnik-Fibel*. Elektronik Kompendium, Oktober 2011.

[37] P. Schnabel. *Netzwerktechnik-Fibel*. Elektronik Kompendium, Oktober 2011.

[40] M. Skolnik. *Radar Handbook*. McGraw-Hill.

[41] K. W. Smith. *Cython*. O'Reilly, Januar 2015.

[61] G. Zaccone. *Python Parallel Programming Cookbook*. Packt Publishing, August 2015.

Internet References

[2] Bundesnetzagentur. *Homepage*. 2017. URL: https://www.bundesnetzagentur.de/cln_1431/DE/Home/home_node.html.

[3] D. Bush. *UFTP Multicasting*. 2017. URL: http://uftp-multicast.sourceforge.net/.

[7] D-Link. *DWR-921 Mobile Router*. 2017. URL: http://www.dlink.com/de/de/products/dwr-921-4g-lte-router.

[9] ECMA. *JSON*. 2017. URL: http://www.json.org/.

[12] Internet Engineering Task Force. *IANA and RFC*. 2017. URL: https://www.ietf.org/rfc.html.

[13] GNU-Radio Foundation. *Homepage*. 2017. URL: https://www.gnuradio.org/.

[14] Network Time Foundation. *Public NTP*. 2017. URL: http://www.ntp.org/.

[18] Hilberling. *Homepage*. 2017. URL: http://www.hilberling.de/.

[19] M. Juenemann. *Python TSIP*. 2017. URL: https://pypi.python.org/pypi/tsip/0.2.0.

[20] Canonical Ltd. *Xubuntu*. 2017. URL: https://xubuntu.org/.

[22] Netgear. *Webmanaged Layer-3 Switches*. 2017. URL: http://www.netgear.de/business/products/switches/web-managed/.

[23] NumPy. *Documentation*. 2017. URL: http://www.numpy.org/.

[24] MIT Haystack Observatory. *Open Radar Initiative*. 2017. URL: http://www.haystack.mit.edu/atm/open/radar/index.html.

[29] Python. *Documentation*. 2017. URL: https://www.python.org/doc/.

[32] Ettus Research. *Products*. 2017. URL: https://www.ettus.com/product.

[33] Ettus Research. *USRP Hardware Driver*. 2017. URL: https://files.ettus.com/manual/.

[34] Tomco RF. *Homepage*. 2017. URL: http://www.tomcorf.com/.

[38] SciPy. *Documentation*. 2017. URL: http://www.scipy.org/.

[43] Ubuntu Documentation Team. *GZIP*. 2017. URL: https://wiki.ubuntuusers.de/gzip/.

[44] Ubuntu Documentation Team. *Hamachi LogMeIn*. 2017. URL: https://wiki.ubuntuusers.de/LogMeIn_Hamachi/.

[45] Ubuntu Documentation Team. *INITD*. 2017. URL: https://wiki.ubuntuusers.de/Dienste/.

[46] Ubuntu Documentation Team. *Interfaces*. 2017. URL: https://wiki.ubuntuusers.de/interfaces/.

[47] Ubuntu Documentation Team. *Mobile Shell*. 2017. URL: https://wiki.ubuntuusers.de/Mosh/.

[48] Ubuntu Documentation Team. *Network File System*. 2017. URL: https://wiki.ubuntuusers.de/NFS/.

[49] Ubuntu Documentation Team. *Network Time Protocol*. 2017. URL: https://wiki.ubuntuusers.de/Systemzeit/.

[50] Ubuntu Documentation Team. *NOHUP*. 2017. URL: https://wiki.ubuntuusers.de/nohup/.

[51] Ubuntu Documentation Team. *Processes*. 2017. URL: https://wiki.ubuntuusers.de/ps/.

[52] Ubuntu Documentation Team. *RAM-Disk*. 2017. URL: https://wiki.ubuntuusers.de/RAM-Disk_erstellen/.

[53] Ubuntu Documentation Team. *RSYNC*. 2017. URL: https://wiki.ubuntuusers.de/rsync/.

[54] Ubuntu Documentation Team. *Runlevel Script Execution*. 2017. URL: https://wiki.ubuntuusers.de/rc.local/.

[55] Ubuntu Documentation Team. *Secure Shell*. 2017. URL: https://wiki.ubuntuusers.de/SSH/.

[56] Ubuntu Documentation Team. *Secure Shell File System*. 2017. URL: https://wiki.ubuntuusers.de/FUSE/sshfs/.

[57] Ubuntu Documentation Team. *Systemd*. 2017. URL: https://wiki.ubuntuusers.de/systemd/.

[58] Trimble. *Thunderbolt E Starter Kit*. 2017. URL: http://www.trimble.com/timing/thunderbolt-e.aspx.

[62] ZOTAC. *Mini-PC's*. 2017. URL: https://www.zotac.com/de/product/mini_pcs/all.

Thesis References

[5] M. Clahsen. "Detektion und Identifikation von mit Hilfe des MMARIA Konzepts gewonnenen Specular-Meteor-Echos". Bachelor Thesis. Universitaet Rostock, 2015.

[15] S. Geese. "Entwicklung eines multistatischen kodierten CW-Radars mittels USRP zur Analyse der mittleren Atmosphäre". Master Thesis. Hochschule Wismar, 2016.

[27] N. Pfeffer. "Radar-Messsystem mit USRP-N200". Bachelor Thesis. Hochschule Wismar, 2015.

[59] J. Vierinen. "On statistical theory of radar measurements". Dissertation. Aalto University, 2012.

Publication References

[1] R. H. Barker. *Group Synchronizing of Binary Digital Sequences*. 1953.

[4] G. Stober, J. L. Chau. *Multistatic and Multifrequency novel approach for Specular Meteor Radars to improve Wind Measurements in the MLT Region*. doi: 10.1002/2014RS005591: Radio Science, 2015.

[17] M. J. E. Golay. *Complementary sequences*. doi: 10.1109/TIT.1961.1057620, 1961.

[39] Y. S. Shmaliy. *Noise Power Gain for Discrete-Time FIR Estimators*. 2010.

[42] J. Vierinen, J. L. Chau, N. Pfeffer, M. Clahsen, G. Stober. *Coded continuous wave meteor radar*. doi: 10.5194/amtd-8-7879-2015: Atmospheric Measurement Technique Discussions, 2015.

[60] J. Vierinen, M. S. Lehtinen, M. Orispaa, I. I. Virtanen. *Transmission code optimization method for incoherent scatter radar*. 2008.