A Hardware/Software Framework for Multiantenna Receivers

Janos Buttgereit, Erik Volpert, Horst Hartmann,
Dirk Fischer, Götz C. Kappen
NTLab
University of Applied Science Münster
Münster, Germany
Email: goetz.kappen@fh-muenster.de

Tobias Gemmeke
IDS
RWTH Aachen University
Aachen, Germany
Email: gemmeke@ids.rwth-aachen.de

***Abstract*—Multiantenna receivers play a significant role if security and spectral efficiency are critical issues for given applications. This could be during the setup of large wireless sensor networks in the Internet of Things (IoT) as well as for applications which suffer from interfering signals. This paper presents a framework to setup and evaluate real-time hardware/software demonstrators, based on flexible Software Defined Radio (SDR) hardware, low-cost multiantenna-arrays and a modular software architecture. The main purpose of this framework is to evaluate cost-benefit parameters (i.e., required processing power, logic resources vs. performance of the multiantenna algorithms) of the overall multiantenna receiver (i.e., antenna, analog and digital signal processing). Therefore, size and power consumption as well as miniaturization of the demonstrator are not considered at this time. To motivate software functions and high-level software architecture, this paper gives a theoretical background of multiantenna receivers and associated algorithms. A highly adaptable and modular C++-based framework has been developed that realizes all relevant low level and high level signal processing tasks (e.g., ADC-data transfer, online system calibration, Direction of Arrival (DoA) estimation and interferer suppression), as well as graphical visualization of the spatial spectrum. The multithread-based realization of the demonstrator ensures high performance and a convenient user experience. First measurements of the whole system (i.e., low-cost antenna, C++-based high level and low level signal processing, as well as graphical visualization using a host PC) in a real-world environment, proof functional correctness while demonstrating real-time capability of the overall system. This paper gives a qualitative overview of the required effort to change the center frequency or the type of modulation. Also, the paper shows the requirements to perform a change in the application domain, e.g., switching from DoA-estimation or interferer suppression.**

***Index Terms*—Multiantenna Systems; Wireless Sensor Networks; Spectral Efficiency; Software-defined-radio; IoT; GNSS; DECT.**

I. Introduction

Wireless systems have gained a fundamental importance in our everyday life. Since the number of transmitters increases rapidly, spectral efficiency and tolerance of interfering signals will be main issues for wireless systems in the upcoming years. In the following paragraphs three widely used systems (i.e., IoT, Global Navigation Satellite Systems (GNSSs), and mobile phones based on the Digital Enhanced Cordless Telecommunication (DECT) standard), are described to show their importance and how these systems can benefit from multiantenna technology. Multiantenna receivers are able to significantly improve spectral efficiency by using digital beamforming techniques. Interferer suppression can be realized by nulling techniques in the spatial domain [1]. Finally, the DoA of signals and interferers can be estimated, which can be used to increase received signal strengths and improve the security of the communication channel by digital post-processing in the spatial domain [2], [3].

Figure 1 shows a simple stack of a wireless sensor node, featuring sensor or general data sink/source, preprocessing, and analog and digital multiantenna processing. For the IoT-case the multiantenna approach is used to communicate within the sensor network, transmit sensor data or receive configuration data. Thus, the lowest layer is the data source in most of the cases. For a GNSS receiver the multiantenna receives the signals from the satellites and generates position, navigation and timing (PNT) information, which can be used by another entity. Hence the lowest layer in Figure 1 will be described by a data sink or a memory unit.

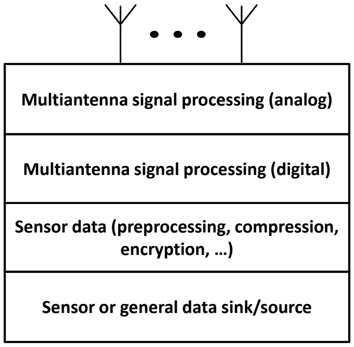

Multiantenna signal processing (analog)
Multiantenna signal processing (digital)
Sensor data (preprocessing, compression, encryption, …)
Sensor or general data sink/source

Fig. 1. Simple model of an IoT-sensor node.

The major drawbacks of multiantenna transceivers are the increased amount of required digital signal processing, as well as the complexity of algorithms and software-code. Therefore, a clear code structure, as well as efficiency, flexibility and reusability of the code play a central role, when realizing the digital signal processing layers of the receiver. Also, for the sensor node, special care must be taken during the realization of the analog part and the data transfer to the digital domain. Especially, interferer suppression and DoA-estimation rely on coherent signal reception and processing. This paper focuses on the two upper layers (i.e., digital and analog multiantenna signal processing) shown Figure 1 and the antenna array.

Since, during the design and evaluation process, flexibility is the key challenge, a flexible and modular SDR-approach is adopted to implement receivers for various systems. A generic antenna design has been used to complete the signal processing chain. Thus, the proposed overall system features a high degree of flexibility and can be easily adapted to different receiver standards and frequency bands (e.g., IoT, DECT, GPS) described in the following subsections.

A. *IoT*

During the next years the number of IoT-nodes will continue to increase rapidly [4] [5]. At the same time, the complexity of IoT-nodes extends over a wide range starting with simple sensor nodes, used for temperature or humidity measurements, to fully integrated embedded systems which are able to control processes and act autonomously. Figure 2 shows the exponential increase of IoT-nodes starting from 1992 and gives a forecast of the number of devices in 2025 [5]. The figure also shows typical achievements and wireless applications over time.

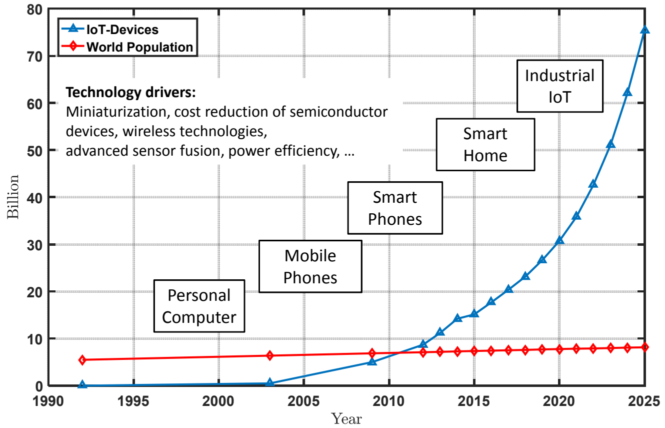


Fig. 2. IoT-Roadmap (based on: [4]–[6]).

Additionally, the world population is given for the same years and it can be seen that with the year 2011, the number of IoT-devices per person is greater than one. The dominant drivers of this evolution are miniaturization, cost reduction and increased power efficiency of semiconductor and sensor devices - and the overarching digitalization of our daily lives. Most IoT-based sensor nodes exchange data adopting wireless standards, suitable for required short or long-range communication. Thus, since the spectrum is a limited resource, spectral efficiency will play a crucial role during IoT-transceiver development. Moreover, communication security and resistance against interfering signals will be further design objectives, as they are already today in nearly all other wireless systems [7]. For all examples in this paper, dealing with IoT, a receiver setup for the 2.4 GHz ISM-band is assumed.

B. *Satellite Navigation Receivers*

While the first subsection deals with IoT receivers, the focus in this section lies on GNSS receivers. The most popular GNSS is the American Navstar GPS operational since 1995 [8]. Today, there are other systems available as the Russian GLONASS [9], the Chinese Beidou [10] or the European Galileo [11] based on the same idea as Navstar GPS [12]. All these systems assume that the satellite positions are known, based on the transmitted orbital data and the time of transmission. During the last decades the importance and dependency of the every day life on GNSSs has become clearly visible. Today, the power grid systems, access and industrial control, banking operations and communication systems rely heavily on GPS to provide worldwide position, navigation, and timing information. At the same time, the number of applications and sold GNSS receivers increases rapidly [13]. Also, GNSSs have become part of the critical infrastructure (see [14] and [15]), which confirms the importance of this technology.

Nevertheless, GNSSs are very vulnerable to interfering signals [16]. These signals might be transmitted unintentionally (e.g., spurious frequencies from digital television (DVBT) or from distance measurement equipment (DME) used at airports to guide incoming airplanes). The reason for the high susceptability is the very low signal power of GNSS signals received at the earth's surface. Figure 3 shows the power spectral density (PSD) for the GPS signal, the noise floor and an interfering signal. The GPS signal level is about 20 dB below the noise floor since these signals (as the signals of all other existing GNSSs) are transmitted as spread-spectrum signals. That means they are composed of a carrier signal, a Pseudo Random Noise (PRN) sequence and a data signal. To separate signals from different satellites a Code Division Multiple Access (CDMA) based scheme is adopted.

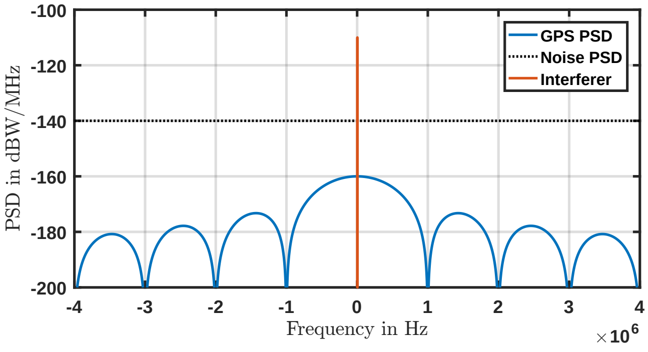


Fig. 3. Comparison of Signal and Noise PSD.

The PRN sequence is multiplied with the navigation data (which for GPS has a data rate of only 50 Hz) and therefore the overall power of the signal is distributed over a large frequency range with a main lobe bandwidth of about 2 MHz (see Figure 3). Thus, a standard GPS receiver placed near the earth's surface receives the incoming signal with only ≈ -158 dBW. To recover the navigation message, which includes the orbital data, the receiver correlates the incoming signal with the known and satellite specific PRN code and thus de-spreads the signal. This correlation process compresses the signal power in a frequency band of about 100 Hz and yields about 40 dB of correlation gain. After this correlation process the 50 Hz navigation message can be recovered [17]. Afterwards, the time of transmission and orbital data can be decoded. With this data the receiver calculates the satellite position at time of transmission. Finally, the receiver determines its current position, time and velocity based on the distance measurements to at least four satellites.

While the dependency on satellite navigation increases, the disruption of GNSS signals has also become more and more obvious during the past years. Simple GNSS jammers transmit a signal with a significantly higher power in the GNSS frequency band and therefore prevent successful signal reception [17]. Figure 3 shows the PSD of GPS, Noise and the interfering signal before the despreading process. In the time domain only the interfering signal and noise is visible. After the despreading process the GPS data signal has a bandwith of about 50 Hz and a PSD maximum value of -120 dBW/MHz. At the same time the power of the CW-interferer is spread to a bandwith of about 2 MHz and the maximum value of the PSD is about 40 dB lower. It can be seen that standard GNSS receivers therefore have a certain resistance to interfering signals as long as the interferer-to-noise ratio (INR) is lower than $\approx$ 40 dB [17]. Nevertheless, interfering signals at the earth's surface easily exceed this budget because of the very weak GNSS signal. More complex architectures called GNSS-spoofers counterfeit the GNSS signal so that the user position can be faked [18]. Both types of interfering signals can be identified and suppressed based on spatial signal processing techniques using SDR-based receivers and the multiantenna design approach.

C. DECT Receivers

Mobile phones based on the DECT-standard are very widespread. In Europe DECT operates in the frequency range from 1881 MHz to about 1897 MHz and therefore directly below a frequency range assigned to the Global System for Mobile Communications (i.e., GSM). Nevertheless, since a large number of handsets and base stations are equipped with low cost oscillators, the proposed center frequency is changing over the time and thus can act as an interferer for GSM communication systems.

DoA-estimators, built based on the approach presented in this paper, significantly simplify the detection of DECT transceivers with this malfunction. The system presented is a 3D-DoA estimator and therefore replaces the standard approach using directional antennas to find interfering DECT base stations. Since the transmitted data is irrelevant for the detection of DECT transceivers, only the lowest layer of the DECT protocol should be considered for DoA-estimation. This layer is responsible for the realization of transmission channels over the radio medium.

The DECT protocol uses a combination of frequency division multiple access (FDMA), time division duplexing (TDD) and space division multiple access (SDMA) to realize several connections at the same time [19]. The DECT center frequencies for the FDMA realization can be calculated using:

$$f_c = 1897.34 \text{ MHz} - a \cdot 1.728 \text{ MHz with } a = 0, 1, ..., 9. \quad (1)$$

Furthermore, DECT defines 24 time slots, which together form a frame, shown in Figure 4. The first 12 time slots realize the downlink, the last 12 the Uplink. Each time slot includes up to 480 bits. However, part of the time available for transmission is used as guard time. Most bits are spread up to a so-called synchronization field S or data field D. In addition, a Z-field can be appended to the data field. This Z repeats the last 4 bits from to detect collisions between two channels. Figure 4 shows a full slot, which is one of the possible frame structures.

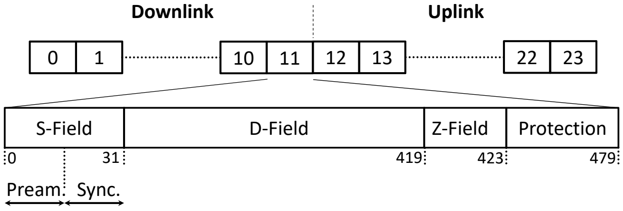


Fig. 4. DECT protocol full slot.

This lower layer allows a 3D DoA-estimation based on a spatial covariance matrix and the algorithms described in Section IV.

D. Organization of the paper

The rest of the paper is organized as follows. Section II gives a general discussion of the problem from the application's and user's point of view. It can be seen that the DoA-estimation is a crucial part during beamforming and interferer suppression, as well as during the process assessing information about the current environment. For a mathematical description, Section III defines the signal model and presents the simulation and receiver test environment. Section IV briefly introduces the spatial processing algorithms for DoA estimation and interferer suppression. Afterwards, Section V gives an in-depth description of the hardware used throughout the paper. The central part of the presented receiver is the SDR, which allows to select various frequency bands and to define sampling frequency and receiver bandwidth. Additionally, this section provides a high level overview of C++-based receiver software (low-level and high-level Digital Signal Processing (DSP)) and Graphical User Interface (GUI) programming, as well as a description of the various external and internal interfaces of the system, while details of the receiver software are discussed in Section VI. The final part of Section V presents some details of the low-cost antenna design and setup. Section VI is devoted to the software-realization of the receiver and gives implementation details of the main blocks of the receiver software (e.g., recording of the incoming frontend samples, calculation of the covariance matrix, DoA-estimation and visualization of the time plot and the DoA-spectrum. Special emphasis lies on the thread-based realization to ensure real-time performance, portability and flexibility). Therefore, this section deals with three central points:

- Parallel realization of the receiver software tasks.
- Object oriented programming to ensure flexibility and cope with large code-complexity.
- Cross-platform realization of the software-code.

In Section VII-A and Section VII-B, the used measurement setup and measurement results are presented to show the potential of the overall receiver hardware/software-concept. Section VIII shows the steps required to change the application domain or frequency band. Finally, Section IX summarizes the paper and lists the intended optimization steps of the receiver hardware/software (i.e., miniaturization, introduction of new algorithms, introduction of new applications).

II. Problem Definition

Spatial signal processing and a flexible hardware/software architecture for real-time implementation of this processing are mandatory to solve the problems discussed earlier.

IoT-node networks suffer from the operation of a large amount of nodes in close vicinity and indoor environment. Interference and spoofing are the main problems for GNSS receivers. As discussed, for the DECT-application the search for transmitters using a wrong transmission frequency poses the relevant task. Therefore, as discussed in the introductory part this leads to the following problems:

- Interference,
- Spoofing and
- Multipath (especially in an indoor environment).

While multiantenna concepts are able to mitigate these problems, hardware and software development is time-consuming, and power consumption of the sensor node is always a critical issue [20]. Therefore, the neuralgic task is to perform a cost-benefit-analysis (e.g., minimal power consumption vs. meeting application defined development time and performance) in short time.

To quickly develop and evaluate a multiantenna systems (e.g., IoT-nodes, multi-antenna GNSS receivers, and DECT localization devices) its performance has to be assessed efficiently validating that user requirements are met. This includes the quantitative assessment of different DoA estimation or interferer suppression algorithms as well as low cost antenna setups for various real-world signal situations under real-time conditions.

Thus, the first step is to develop a modular PC-application that uses SDR-hardware as input source, runs various estimation algorithms and visualizes their results in real-time using a GUI. This application acts as a proof-of-concept demonstrator and shall help to judge performance of the algorithms and antenna arrays under various circumstances and trigger critical corner cases to ultimately develop better or cost-effective algorithms and arrays. This research focused exploration phase is followed by a design and realization phase of the low-cost and low-power sensor, analog and digital signal processing hardware (cf. Figure 1).

III. Signal model and simulation

This section introduces the signal model and the algorithms used for DoA-estimation (Capon-Beamformer and Multiple Signal Classification (MUSIC) algorithm [21]) and interferer suppression (MVDR [2] and subspaced based). Additionally, the simulation setup, as well as simulation results are presented.

A. Signal Model

In this section the signal model, based on the theory described in [2], [3] and [21], is briefly summarized while the description is restricted to one received signal. It is assumed that the receiver is in the far field of the transmitter, the narrow band assumption holds and that the antenna has a flat frequency response. Then the vector $\mathbf{u}$, which is used to describe signal and interferer, can be defined. Figure 5 shows an arbitrary antenna array with N randomly distributed antenna elements and the vector $\mathbf{u}$.

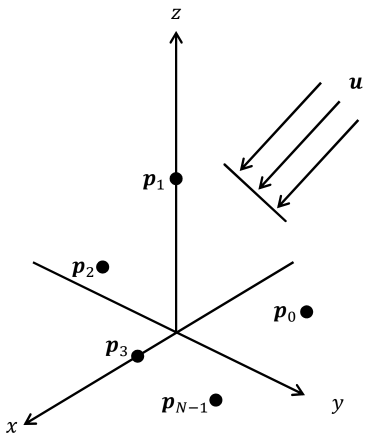


Fig. 5. Multiantenna Model.

Then, $\mathbf{u}$ can be written, depending on ϕ and θ as

$$\mathbf{u}(\phi,\theta) = \begin{pmatrix} -\sin\theta\,\cos\phi \\ -\sin\theta\,\sin\phi \\ -\cos\theta \end{pmatrix} = -\mathbf{a}(\phi,\theta) \tag{2}$$

and the wave number $\mathbf{k}$, relative to the origin of the given coordinate system can be calculated as

$$\mathbf{k}(\phi,\theta) = \frac{2\pi}{\lambda}\cdot\mathbf{u}(\phi,\theta). \tag{3}$$

In the following, the angles ϕ and θ are omitted. Furthermore, it is assumed that an N-element antenna (cf. Figure 5) receives this signal from a defined DoA. Then, the time-dependent output vector is described by the following equation:

$$\mathbf{x}(t) = \exp\left(-\mathrm{j}\mathbf{p}\mathbf{k}\right)s(t) + \mathbf{n}(t) = \mathbf{a}s(t) + \mathbf{n}(t) \tag{4}$$

Next, the so called spatial covariance matrix $\mathbf{R}$ can be estimated using the estimation operator $E\{\cdot\}$ as

$$\begin{aligned}\mathbf{R} &= E\{\mathbf{x}(t)\mathbf{x}^H(t)\} \\ &= \mathbf{a}E\{\mathbf{s}(t)\mathbf{s}^H(t)\}\mathbf{a}^H + E\{\mathbf{n}(t)\mathbf{n}^H(t)\} \\ &= \mathbf{a}\mathbf{P}\mathbf{a}^H + \sigma^2\mathbf{I}\end{aligned} \tag{5}$$

Equation (5) can be re-written using a unitary matrix $\mathbf{U}$ and a matrix of the N Eigenvalues $\Lambda = \text{diag}\{\Lambda_0, ..., \Lambda_{N-1}\}$ [22].

$$\begin{aligned}\mathbf{R} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H \\ &= \mathbf{U}_s\mathbf{\Lambda}_s\mathbf{U}_s^H + \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^H\end{aligned} \tag{6}$$

The Eigenvalues of noise (index n) and signal (index s) have been separated. For a real-word implementation only a limited number of samples can be recorded and used to estimate the spatial covariance matrix. Following the conventions of [21] this matrix is called $\hat{\mathbf{R}}$.

The above discussion assumes that the DoA of a useful signal should be estimated. While this use case is described in more detailed in the following section, subsection IV-B discusses the case of interferer suppression, i.e., removing interfering signals.

IV. Spatial Algorithms

A. Direction of Arrival Estimation (DoA)

In this work two DoA-estimation algorithms are considered. The Capon beamformer generates a so called spatial spectrum by using a beamsteering approach. This simple approach is limited, especially if two signal sources impinge from closely separated elevation and azimuth angles. A great benefit is the very low computational complexity and a smooth spatial spectrum. The MUSIC algorithm [21] uses a subspace based approach, which yields very high DoA-estimation accuracy at the cost of increased computational complexity. Both algorithms generate a spatial spectrum, where the maximum gives an estimate of the DoA of the incoming signal.

1) Capon Beamfomer: For the Capon beamformer, the spatial spectrum is defined as:

$$P_{\text{CAP}} = \frac{1}{\mathbf{a}^H(\phi,\theta)\hat{\mathbf{R}}^{-1}\mathbf{a}(\phi,\theta)} \tag{7}$$

It can be seen that the Capon beamformer algorithm is based on the inverse spatial covariance matrix $\hat{\mathbf{R}}^{-1}$. During the search phase the steering vector $\mathbf{a}(\phi,\theta)$ is generated for all ϕ and θ, and the values for the spatial spectrum $P_{\text{CAP}}(\phi,\theta)$ are calculated and stored. The final step of the algorithm is to find the maximum in the two-dimensional search space.

2) MUSIC Algorithm: The MUSIC spectrum is defined as:

$$P_{\text{M}} = \frac{\mathbf{a}^H(\phi,\theta)\mathbf{a}(\phi,\theta)}{\mathbf{a}^H(\phi,\theta)\hat{\mathbf{U}}\hat{\mathbf{U}}^H\mathbf{a}(\phi,\theta)} \tag{8}$$

After the spatial spectrum has been estimated a search algorithm estimates the absolute maximum.

B. Interferer Suppression

For interferer suppression a simplified version of the Applebaum array [2] is used. Again, the estimated spatial covariance matrix $\hat{\mathbf{R}}$ is used to calculated the weights $\mathbf{W}$.

$$\mathbf{W} = \mu\hat{\mathbf{R}}^{-1}\mathbf{U}_d^* \tag{9}$$

In equation (9), μ is an arbitrary scalar constant which can be used to scale the weights $\mathbf{W}$. Again the inverse of the covariance matrix $\hat{\mathbf{R}}$ is used. Finally, $\mathbf{U}_d$ in equation (9) allows to steer the beam into a desired direction. For the GNSS case it is assumed that the received signal is composed of the useful signal, noise and the dominant interfering signal. Applying the weights $\mathbf{W}$ to the input signal, received at the N antennas, the dominant signals are removed and, in the GNSS case, the output signal has a noise like characteristic. Since the GNSS signal is about 20 dB below the noise floor the interferer suppression algorithm does not affect the useful signal. Additionally, if the satellite position is already known based on another information source (e.g., GNSS assistance data) $\mathbf{U}_d$ can be used to perform beamsteering and increase signal receive power. Again, the estimated spatial covariance matrix $\hat{\mathbf{R}}$ and the decomposition in equation (6) can be used to efficiently calculate the inverse of the spatial covariance matrix.

$$\hat{\mathbf{R}}^{-1} = \hat{\mathbf{U}}\hat{\mathbf{\Lambda}}^{-1}\hat{\mathbf{U}}^H \tag{10}$$

As can be seen in equation (10), the inverse of $\hat{\mathbf{R}}$ can be calculated based on the unitary matrix $\hat{\mathbf{U}}$ and the matrix $\hat{\mathbf{\Lambda}}^{-1}$ which has the reciprocal Eigenvalues of $\hat{\mathbf{R}}$ on the main diagonal.

C. Real-time Receiver Tests

The whole receiver signal processing chain has been developed and simulated in MATLAB. This Golden Reference model has been used during the receiver design process (see Section VI) to validate the correctness of the real-time C++-based receiver results. Figure 6 shows the signal processing path for the case of DoA estimation.

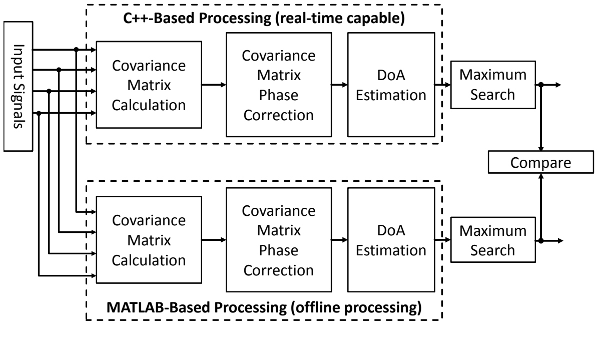

C++-Based Processing (real-time capable)
Input Signals
Covariance Matrix Calculation
Covariance Matrix Phase Correction
DoA Estimation
Maximum Search
Compare
Covariance Matrix Calculation
Covariance Matrix Phase Correction
DoA Estimation
Maximum Search
MATLAB-Based Processing (offline processing)

Fig. 6. MATLAB and Real-Time C++ Comparison.

The modulated carrier signals with random elevation and azimuth angles were generated in MATLAB for each sensor element and for various array geometries (i.e., circular, rectangular and uniform linear). Additionally, additive white Gaussian noise has been added to the signals (cf. equation (4)). These signals were used as input signals during C++-based and MATLAB based offline processing.

Both processing paths in Figure 6 estimate the covariance matrix, spatial spectrum, as well as the azimuth and elevation angles using floating-point precision (see Section VI for implementation details). Since input signal and data precision are identical, the results could be directly compared, which eases the debugging of the real-time capable C++-receiver.

Additionally, the effect of a reduced precision (e.g., single precision calculations) can easily be investigated. In case of a DoA application the results show for example that the spatial spectrum of the Capon Beamformer is slightly degraded while the MUSIC spectrum is identical based on a resolution of 1° (see Section VI).

V. SDR-Based Receiver Overview

The following subsections give an overview of the hardware (i.e., SDR, host computer and low cost multiantenna) used to realize the DoA estimation, while the software is described in detail in Section VI. Mainly Ettus SDRs, equipped with daughterboards and connected to a host computer using 10 GBit/s connections are used for analog preprocessing, analog-to-digital conversion and realization of the signal processing algorithms (cf. Figure 7). On the host computer the Ettus API is used to establish the connection, control data transfer and configure the Ettus daughterboards. Moreover, the DoA-estimation, interferer suppression and calibration algorithms, as well as the GUI are implemented on the host computer. For maximal flexibility (i.e., center frequency, antenna dimensions and geometry, as well as number of antenna elements) and minimal costs, the receiver antenna array is manufactured in-house based on simple dipole antennas.

A. Receiver Hardware-Setup

A general approach of low cost multiantenna receivers for GNSS receivers has been described in [7]. Since the hardware should be used to evaluate various DoA and interferer suppression algorithms, the concept presented in this work replaces the FPGA development board used in [7] with a commercially available SDR [23]. This architecture features substantially more flexibility, which comes at significantly higher costs. A reasonable trade-off, which is acceptable during this early phase of the receiver design. The proposed receiver hardware is based on an Ettus SDR USRP X300 equipped with two SBX daughterboards [23]. Each daughterboard has a frequency range from 400 MHz to 4.4 GHz, allows duplex operation, 40 MHz bandwidth and 16-bit ADC resolution. Each X300 device can be equipped with two daughterboards, therefore a 4 channel SDR-receiver requires four SBX daughterboards and two X300.

Figure 7 shows the setup based on multiple, independent receiver units, each generating their own LO (Local Oscillator) signal. As the phase relation of the received signals is a key factor for most DoA and interferer suppression algorithms, and the LO-phase will be added to the input signal phase, independent LOs will generate useless input signals. If the phase offset between the individual LOs is known, they can be canceled out by correcting the unwanted phase shift in software. To address this issue, the SDR-receivers, used in the presented setup, have two separate inputs, one connected to the antenna and one connected to a synchronization signal that is distributed to all receivers from a central signal source.

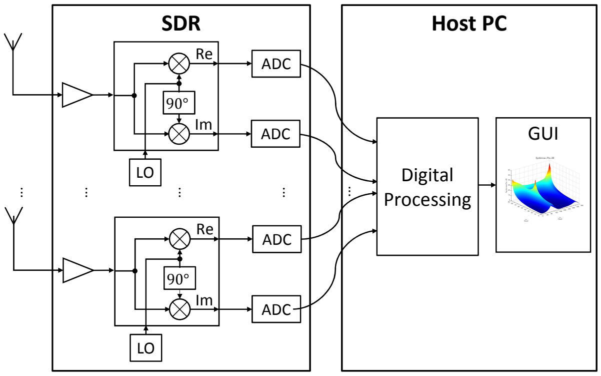


Fig. 7. General Schematic of a Multiantenna-Receiver.

Measuring results show that switching over to the synchronization signal every five seconds to re-calibrate the LO-phase error correction values is sufficient to get an overall stable measurement situation. Additionally, a time-invariant phase error is introduced by slightly different cable lengths (i.e., connections between antenna array and receiver). This error was measured once and is added as a time-invariant complex correction factor to the dynamically measured correction factors.

B. Software Overview and GUI

A high level schematic of the demonstrator software is shown in Figure 8. On the left hand side the four 16-bit digital input streams enter the signal processing stage and the spatial covariance matrix is calculated. The subsequent block performs the calibration of the spatial covariance matrix by applying time-varying and time-invariant complex correction factors.

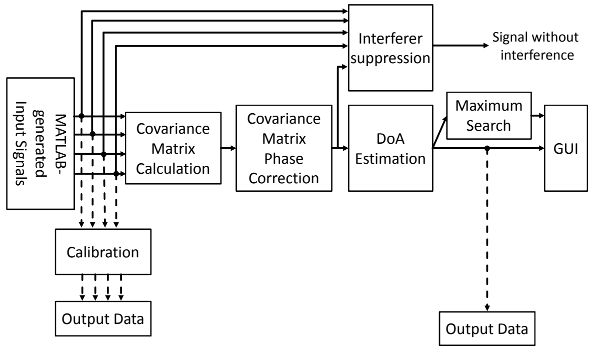

MATLAB-generated Input Signals
Covariance Matrix Calculation
Covariance Matrix Phase Correction
Interferer suppression
Signal without interference
DoA Estimation
Maximum Search
GUI
Calibration
Output Data
Output Data

Fig. 8. Real-Time Demonstrator Schematic.

Based on the corrected covariance matrix, the estimation algorithm (e.g., MUSIC, Capon) generates the spatial spectrum, which is displayed in the GUI. A parallel task searches for the maximum in the spatial spectrum. Its numerical results (i.e., elevation and azimuth) are also displayed in the GUI (cf. Figure 17). For debugging purposes the software allows reading out the four channel input data, as well the output of the estimation algorithm. The data files can be used to compare the results of the C++-based processing of the real-time demonstrator and the MATLAB-based Golden Reference model (see Section III).

C. Antenna Setup

For the design of the 2x2 multiantenna array a two step approach has been taken [1]. First, a single ground plane antenna with four radials has been designed and manufactured (cf. Figure 9).

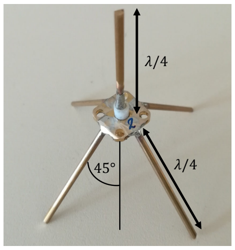


Fig. 9. Single dipole realization.

This type of antenna is low cost, easy to build and allows simplified antenna tuning [24] as well as adaptation to other frequency bands. The driven element and the four radials do have a mechanical length of approximately $\lambda/4$ for the selected center frequency f_c (cf. Figure 10). The antenna impedance is tuned to 50 Ω and the Voltage Standing Wave Ratio (VSWR) has been measured as a quality metric. Based on the approach described in [1] two more antennas (GPS and DECT) have been designed and manufactured (cf. Figure 10). For GPS this type of antenna is not the optimal choice and will be replaced by patch antennas in future designs.

System	Center Frequency	Wavelength λ	Antenna Spacing	Radial Length
ISM	2.45 GHz	12.2 cm	6.1 cm	3.1 cm
GPS	1.575 GHz	19.0 cm	9.5 cm	4.8 cm
DECT	1.89 GHz	15.9 cm	7.9 cm	4.0 cm

Fig. 10. Center Frequencies.

Figure 11 shows the resulting VSWR-plot of a single antenna for ISM-, GPS-, and DECT-center frequencies. It can be seen that all antennas achieve an VSWR ≈ 1 for the required center frequency (cf. Figure 10).

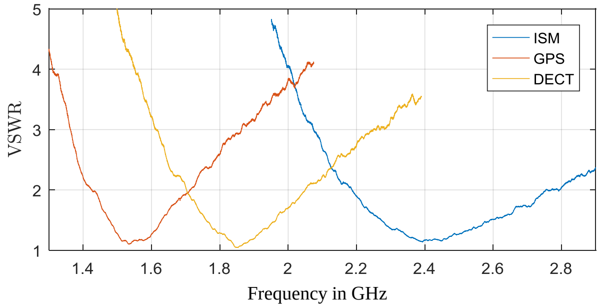


Fig. 11. VSWR-Measurement used for Antenna Tuning.

Additionally, it can be observed that the bandwidth increases for higher center frequencies. This behavior is attributed to relationship of conductor diameter and center frequency. Since at this time bandwidth is not a critical issue this effect will be ignored. As expected, things change when the single elements are combined in 2x2 array as shown in Figure 12 for the case of the ISM-, DECT- and GNSS-frequency band 2x2 multiantenna. In the construction shown, the electronic beam pattern is omni-directional for the azimuth angle, while there is no radiated energy at an elevation angle of $\phi = 0°$. This is a prefect setup for ground based signals and interferer detection systems. It will lead to problems if the desired incoming signals have larger elevation angles.

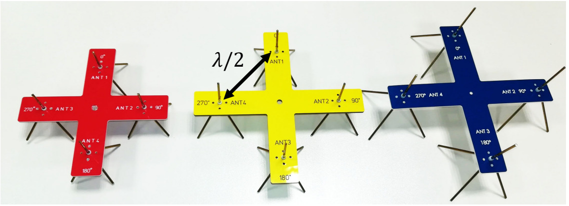


Fig. 12. Low-Cost Antennas for ISM (red), DECT (yellow) and GNSS (blue).

Again the VSWR is used to assess the quality of the manufacturing and tuning process. For the ISM antenna the results are shown in Figure 13. The figure includes all four antennas of the array.

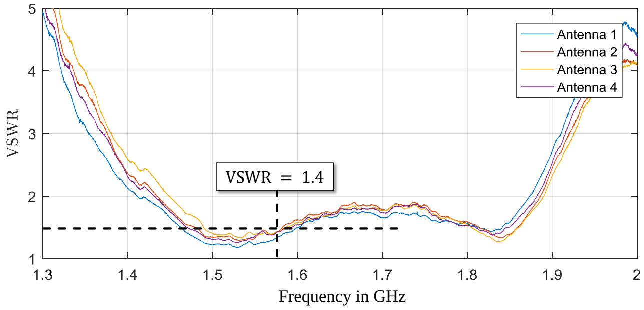


Fig. 13. 2x2 ISM Antenna.

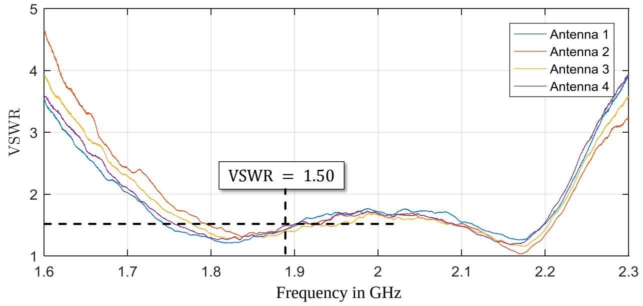


Fig. 14. 2x2 DECT Antenna.

Figures 13-15 show the results of the multiantenna realization for ISM, DECT and GPS receivers. Firstly, the figures highlight the consistency of the achieved antenna VSWR despite of the rather simple manufacturing process. Secondly, as expected, the VSWRplot has a significantly different characteristic compared to the single antenna. As shown, the array arrangement features a wider bandwidth.

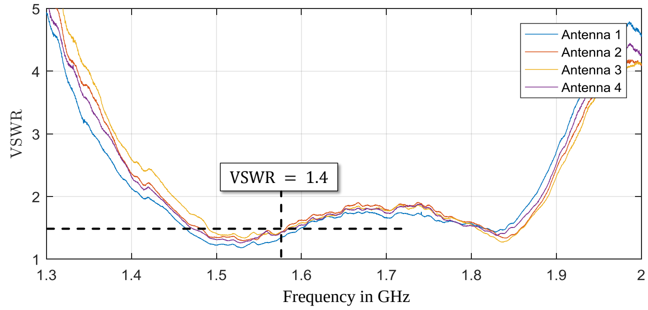


Fig. 15. 2x2 GPS Antenna.

VI. Receiver Software Realization

This section discusses details of the signal processing block realization shown in Figure 8. As mentioned in Section II the software should meet the following key constraints:

- Modular software architecture, e.g., implementing a new estimator or interferer suppression algorithm should be as easy as programming the algorithm itself.
- Modular hardware architecture, e.g., changing antenna array dimensions should be as easy as changing the description of the antenna positions, changing the center frequency should just be a change of a single variable.
- Real-time DSP performance without any sample-drop combined with an optimal GUI-operation.

Furthermore, the phase-synchronization described in Section V has to be implemented. To achieve these goals, a state-of-the-art DSP-software design flow is employed. The software is solely written in C++, using a cross-platform capable framework, originally developed for professional audio DSP-applications [25]. Besides the ability of displaying live measurement snapshots of the input signals in the time domain, the resulting spatial spectrum can be captured at any moment in time and stored to data files. This allows to analyze all parameters for various signal situations in post-processing using software like MATLAB (see Section III).

A. Concurrent Data Processing

To make use of modern multicore-CPUs and meet the throughput requirements, the computations are spread over multiple threads running in parallel, arranged in a software-pipeline structure, where each thread is a consumer of the previous thread's data and a producer for the following thread. Passing data from one thread to another is done by simply swapping buffers.

Figure 16 shows the data flow. All data exchange buffers are allocated twice at start-up. As memory allocation is a system call with unpredictable execution time on general purpose operating systems, avoiding memory allocation on the high and medium priority threads turns the operations invoked on these threads to function calls with fully predictable execution time. This guarantees that the thread's job will be predictably finished before the next data buffer is available for processing.

Samples are received by blocking calls to the Ettus UHD API [26], which invokes the 10 GBit Ethernet interface and returns as soon as a whole block of samples has been received from the hardware units and filled into the buffer passed to the API call. This buffer is forwarded to the sample processing thread afterwards, which returns the buffer it processed in the previous run to the receive thread to be filled again. This enables the new sample block to be processed, while another thread handles in parallel the acquisition of the following sample block.

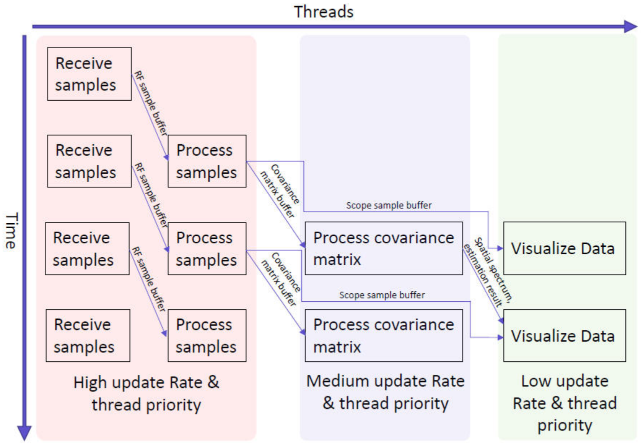


Fig. 16. Multithreaded Software Pipeline.

The sample processing thread fills a buffer if needed and then accumulates samples into the covariance matrix. Computation of this matrix is done by extensive use of SIMD-instructions on sub-vectors that exactly fit one cache-line of the CPU and uses an additional thread, not shown in the figure, to parallelize the matrix computation even further.

After a covariance matrix calculation is finished, the phase correction factors are applied to the matrix, which leads to much smaller computational load, compared to correction on a sample-basis. Depending on the covariance matrix accumulation length, which can be modified using the GUI at runtime, the accumulation process is done over several sample blocks. Thus, in general it takes several runs of the sample processing thread until a covariance matrix is handed over to the covariance matrix processing thread, which realizes the described estimator algorithm. This is why the update rate of the covariance matrix thread is slightly lower. However, the DoA-algorithms invoked on this thread, usually do some computational heavy tasks like Eigenvalue-decomposition and matrix inversion, so the broader time-slot for this thread gives it the ability to finalize computations, before the next covariance matrix is ready.

The estimation algorithms in general are expected to generate a spatial spectrum in the form of a 90x360 matrix (in case of a usual angular resolution of 1° - other values are possible) and two vectors with azimuth and elevation angles of the estimated source positions. Those buffers are again handed over to the GUI-thread that visualizes the spatial spectrum and prints out the positions of sources detected in a given interval. As updating the GUI is scheduled by the operating system, frame drops are theoretically possible at this point. However, those drops will not interrupt the processing activity.

Practically, a GUI framedrop almost never happens, which leads to a smooth presentation of the spatial spectrum.

A special case is handled when the receiver switches over to the synchronization signal. In this case, the covariance matrix computation will be paused and the phase correction value table will be updated, depending on the measured input signal phase offsets.

B. Object-Oriented Signal Processing

Object-oriented signal processing increases flexibility, as it allows a modular structure that directly models the signal flow block-diagram. Classes are used to encapsulate domains, e.g.,

- SDR-hardware
- Sample buffers
- Covariance matrix calculation
- Phase correction measurement and application
- DoA-algorithm
- Interferer suppression
- Spatial spectrum visualization

An important feature of C++ is the ability to describe (fully virtual) interface classes. This feature has been used to describe a generic DoA-algorithm class, consuming a covariance matrix and generating a spatial spectrum, as well as a pair of estimation vectors that can be overridden by an actual implementation. A Capon Beamformer, as well as a MUSIC estimator algorithm have been implemented, which can be chosen at runtime. As mentioned in the earlier sections, further algorithm development is one of the main goals. Thus, implementing new algorithms and switching from the one the other at runtime, while remaining within the same real-world signal situation, is a highly effective feature of the demonstrator. Another powerful options comes from the SDR-hardware abstraction layer, which is currently under development for its next iteration. This next generation will allow to use a completely different receiver hardware, abstracted by the same IO-interface class thus requiring minimal or no changes to the algorithm and visualization part of the software.

C. Cross-Platform Implementation

The abstraction approach described in the previous subsection allows for portability of the code to various processing platforms. In a first version, this allows to build software from the same codebase that runs on all three major operating systems (Microsoft Windows, Linux and Mac OS) without code changes. Therefore, various parts of the software can be implemented on different operating systems and could be seamlessly integrated. This approach significantly speeds up development time as team members could exactly use their development tools of choice. For the final application this results in the key benefit that the whole application or parts of it can be easily ported to an IoT-device. By design, an embedded Linux platform, as used for most IoT-devices, is a fully compatible target for the application, which radically enhances the code re-use factor for upcoming development. Furthermore, deployment to mobile platforms, like Android or iOS, are suitable options.

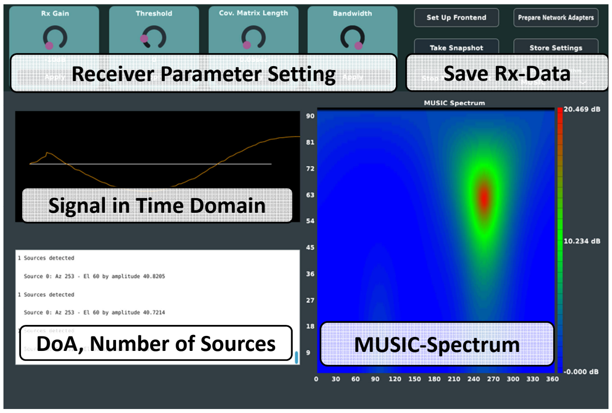
VII. Experimental Results

This section presents the real-time GUI developed. Additionally, compared to [1], quantitative results of DoA-estimation use cases are shown and prove the validity of the presented approach.

A. Measurement Setup

The described SDR-based demonstrator combined with the low-cost multiantenna array has been used to perform indoor measurements in the ISM-band at 2.45 GHz. Since multipath and interfering signals are expected in the utilized frequency band, the environment is close to real-world applications.

A real-time GUI (cf. Figure 17) is urgently required to setup measurement parameters, save debug data and control correct dynamic behavior during the measurements. The GUI features some additional options (e.g., taking a data snapshot, real-time modification of receiver parameters, selection of the DoA-Algorithm), which help to improve overall measurement results, and ease software debugging. The receiver raw data storage capability allows fast development of new algorithms.



```
1 Sources detected
  Source 0: Az 253 - El 60 by amplitude 40.8205
1 Sources detected
  Source 0: Az 253 - El 60 by amplitude 40.7214
```

Fig. 17. Graphical User Interface of the Multiantenna-Receiver.

Besides, a first profiling has be conducted to evaluate the computational requirements of the three threads shown in Figure 16. The profiling results show that about 53% of the overall processing time is consumed by the GUI and the user interaction (i.e., the green block in Figure 16) while 45% is required for the covariance matrix calculations and the DoA algorithm (blue block in Figure 16). The high priority thread (red block in Figure 16) only consumes about 1.5% of the overall processing time. These numbers are a good starting point for optimization and for comparison of various DoA-estimation algorithms.

B. Measurement Results

This section presents first quantitative measurement results for the multiantenna DoA-estimation in the ISM-frequency band. The measurement setup is composed of a single dipole transmit antenna and a multiantenna receiver as shown in Figure 18. The setup guarantees a constant distance between transmitter and multiantenna. Additionally, the setup ensures that the the receiver operates under far field conditions.

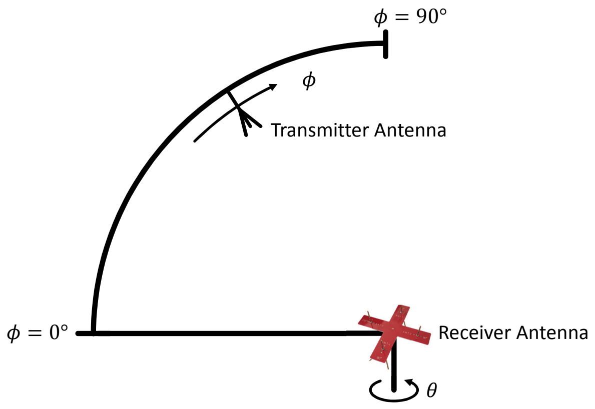


Fig. 18. Measurement setup.

Elevation angles are measured with a 10° spacing moving the transmit antenna from 10° to 90°. Azimuth angles are modified by rotating the multiantenna. Three different azimuth angles have been selected (90°, 135°, and 180°).

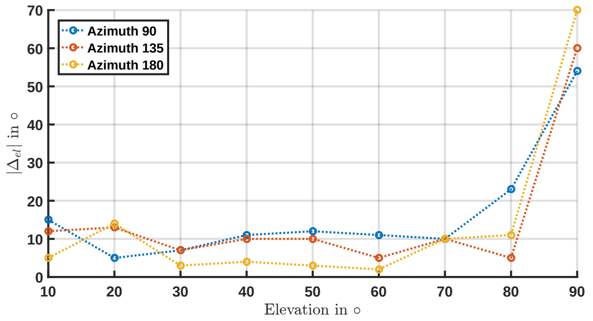


Fig. 19. Elevation error.

Figures 19 and 20 show the absolute estimation error for the elevation angle estimation $|\Delta_{el}|$ and the azimuth angle estimation $|\Delta_{az}|$.

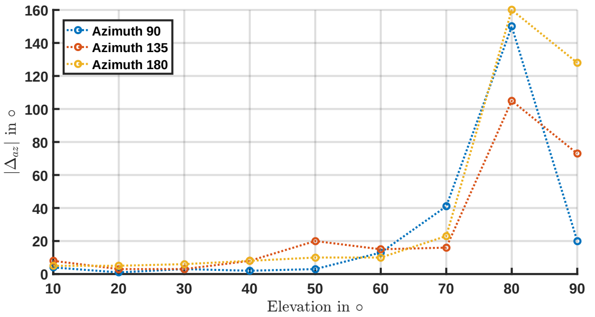


Fig. 20. Azimuth error.

For the elevation angle it can be seen in Figure 19 that the estimation error $|\Delta_{el}|$ is well below 20° for elevation angles smaller than 60°. Because of the dipole beampattern and ambiguities, the estimation error increases rapidly for elevation angles larger than 70°. A similar result is shown in Figure 20 for azimuth error $|\Delta_{az}|$. The overall results show errors about 10° for moderate elevation angles and again the error increases rapidly for elevation angles larger than 70°.

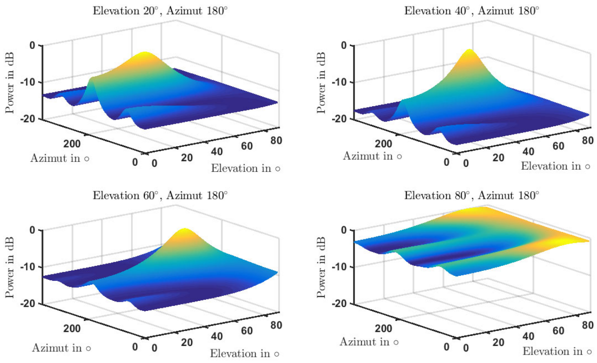


Fig. 21. 3D-DoA-estimation plot.

Figure 21 presents the DoA-estimation for four different elevation angles while a constant azimuth angle of 180° has been chosen. Again, it can be seen that the DoA is estimated correctly for elevation angles smaller than 60° and that the MUSIC spectrum degrades for larger elevation angles.

VIII. Adaptation to other Systems

As shown in [1] the proposed system is able to estimate the DoA of incoming RF-signals in the ISM-frequency band. In this section the simple adaptation of the hardware and software of the SDR-based receivers to other systems will be presented and a qualitative estimation of the required working time is given. Because of the high flexibility of the SDR-based approach and the low-cost antenna manufacturing process, switching to another system requires three main steps:

- Specification and manufacturing of the multiantenna
- Modification of constants in the SDR software code
- Measurements and calibration of the overall system

Thus, switching the system will only take a few hours, while the main time will be invested in measurements and calibration of the overall system. Modification of the SDR-code is mainly a modification of some variables. For the multiantenna manufacturing a computer numerical controlled (CNC) mill has been used to realize the antenna holding, while the single dipole antenna have been manufactured and tuned manually.

IX. Conclusion and Future Work

Spectral efficiency, robustness and security are critical design parameters of wireless IoT-sensor nodes and other wireless systems. Since costs (i.e., silicon area, power consumption) of multiantenna IoT-sensor nodes, compared to single antenna sensor nodes, are significantly higher, a detailed cost-benefit analysis has to be performed in a first step. This paper presents a modular and flexible hardware-/software-architecture, based on an SDR, which realizes the analog preprocessing and the AD-conversion. The modular C++-code realizes all digital signal processing parts, allows simple debugging and features easy extendability. The presented modular and generic approach supports porting the existing software to embedded platforms to reduce size and power consumption in a next step. Finally, a simple technique to realize low-cost antenna arrays supports the overall approach. Measurements and simulations validate functional correctness and the demonstrator shows real-time capability of the overall receiver. The presented concept and design framework has been used to realize multiantenna receivers operating in the ISM-, DECT- and GNSS-frequency band.

ACKNOWLEDGMENT

The authors would like to thank the team of the Central Area of Electrical Engineering and Computer Science (ZBE) for their support during the antenna manufacturing process. Also, we like to thank Kai Eßmann for the 3D-design of the multiantenna mounting.

REFERENCES

[1] J. Buttgereit, E. Volpert, H. Hartmann, D. Fischer, G. C. Kappen, and T. Gemmeke, "Real-Time SDR-Based ISM-Multiantenna Receiver for DoA-Applications," in *Proceedings of the Eleventh International Conference on Advances in Circuits, Electronics and Micro-electronics, CENICS*, 2018, pp. 5–10.

[2] R. T. Compton, *Adaptive Antennas, Concepts and Performance*. Prentice Hall, 1988.

[3] H. van Trees, *Optimum Array Processing*. John Wiley and Sons, Inc., 2002, ISBN: 9780471093909.

[4] Cisco Internet Business Solutions Group (IBSG), "The internet of things," 2011.

[5] "Number of IoT Devices," 2018, URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [accessed: [2019-02-26]].

[6] "World Population," 2018, URL: https://www.populationpyramid.net/world/2025/ [accessed: 2019-02-26].

[7] G. C. Kappen, C. Haettich, and M. Meurer, "Towards a robust multi-antenna mass market GNSS receiver," in *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, April 2012, pp. 291–300.

[8] E. Kaplan, C. J. Hegarty, *Understanding GPS: Principles and Applications*. Artech House, 2006.

[9] Russian Institute of Space Device Engineering, "Global Navigation Satellite System GLONASS Interface Control Document," 2008.

[10] China Satellite Navigation Office, "Beidou Interface Control Document, Open Service Signal (Version 2.0)," 2013.

[11] European Union, "European GNSS (GALILEO) Open Service, Signal-in-Space Interface Control Document," 2016.

[12] Global Positioning Systems Directorate System Engineering Integration, "Interface Specification IS-GPS-200," 2018.

[13] European GNSS Agency, "GNSS User Technology Report," 2018.

[14] C. Durkovich, "GPS and Critical Infrastructure," 2015.

[15] European Commission, "The European Programme for Cricital Infrastructure Protection (EPCIP)," 2016.

[16] R. H. Mitch, R. C. Dougherty, , M. L. Psiaki, S. P. Powell, B. W. O'Hanlon, J. A. Bhatti, T. E. Humphreys, "Signal Characteristics of Civil GPS Jammers," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, 2011, pp. 1907–1919.

[17] M. S. Braasch and A. J. van Dierendonck, "GPS receiver architectures and measurements," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, Jan 1999.

[18] M. L. Psiaki and T. E. Humphreys, "GNSS spoofing and detection," *Proceedings of the IEEE*, vol. 104, no. 6, pp. 1258–1270, June 2016.

[19] B. Walke, *Mobilfunknetze und ihre Protokolle, Mobile Communication Systems and their Protocols*. Teubner, 1988.

[20] J. Xue, S. Biswas, A. C. Cirik, H. Du, Y. Yang, T. Ratnarajah, and M. Sellathurai, "Transceiver Design of Optimum Wirelessly Powered Full-Duplex MIMO IoT Devices," *IEEE Transactions on Communications*, pp. 1955 – 1969, 2018.

[21] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 67–94, Jul 1996.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed. Baltimore: Johns Hopkins University Press, 1989.

[23] "Ettus Homepage," 2014, URL: http://www.ettus.com/ [accessed: 2019-02-26].

[24] T. A. Milligan, *Modern antenna design*. Macmillan, 1985. [Online]. Available: https://books.google.de/books?id=sxUoAQAAMAAJ

[25] "Juce Homepage," 2018, URL: https://juce.com/ [2019-02-26].

[26] "Ettus API," 2014, URL: http://files.ettus.com/manual/page_coding.html/ [accessed: 2019-02-26].