Real-Time SDR-Based ISM-Multiantenna Receiver for DoA-Applications

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*—Spectral efficiency is one of the critical issues, which has to be considered during setup of large wireless sensor networks in the Internet of Things (IoT). This paper presents a real-time hardware/software demonstrator based on a flexible Software Defined Radio (SDR) and a low-cost multiantenna-array. The main purpose of this demonstrator is to evaluate cost-benefit parameters (i.e., required processing power, logic resources vs. performance of the multiantenna algorithm) 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 brief theoretical background of multiantenna receivers. 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 in a multithread-based manner. 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.**

***Keywords*–Multiantenna Systems; Wireless Sensor Networks; Spectral Efficiency; Software-defined-radio; IoT.**

I. Introduction

During the next years the number of IoT-nodes will increase rapidly [1], [2]. Simultaneously, the IoT-node complexity is widely spread starting with simple sensor nodes, used for temperature or humidity measurements, to completely integrated embedded systems which are able to control processes and act autonomously. Figure 1 shows the exponential increase of IoT-nodes starting from 1992 and the forecast of the number of devices in 2025 [2]. Additionally, the world population is given for the same years and it can be seen that from the year 2011 on the number of IoT-devices per person will be greater than one. The dominant drivers of this evolution are miniaturization, cost reduction and increased power efficiency of semiconductor and sensor devices. 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 critical role during IoT-transceiver development. Moreover, communication security and resistance against harmfully interfering signals will be further design objectives, as they are already today in nearly all other wireless systems [3].

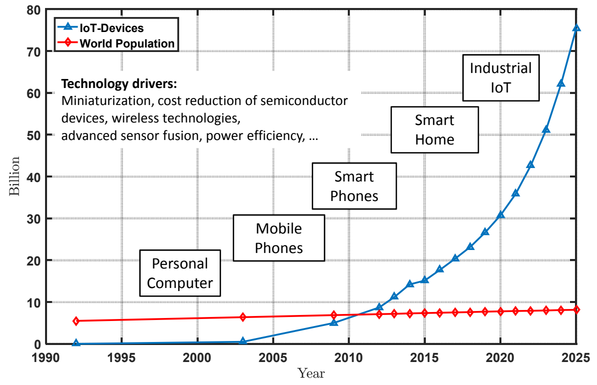


Figure 1. IoT-Roadmap (based on: [1], [2], [4])

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. 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 [5], [6]. Figure 2 shows a simple stack of a wireless sensor node, featuring data sink/source, sensor data preprocessing, and analog and digital multiantenna processing.

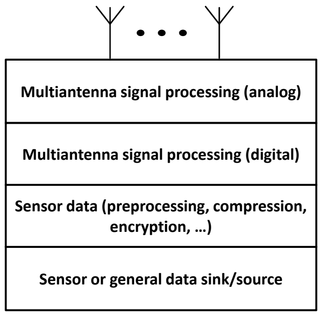

Multiantenna signal processing (analog)
Multiantenna signal processing (digital)
Sensor data (preprocessing, compression, encryption, ...)
Sensor or general data sink/source

Figure 2. 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 re-usability of the code play a central role, when realizing the digital signal processing part 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. Therefore, this paper describes the used SDR and digital calibration techniques to allow for these algorithms to be realized. Finally, the antenna array drives size and costs of the receiver and therefore is the key for user acceptance and suitable application domains. This paper focuses on the two upper layers (i.e., digital and analog multiantenna signal processing) shown Figure 2 and the antenna array. Since, during the design and evaluation process, flexibility is the key challenge, a flexible SDR-approach is adopted to implement these layers of the sensor node. The SDR has been programmed in a very modular way. Thus, the proposed system is very flexible and can be easily adapted to other receiver standards and frequency bands (e.g., DECT, GPS). Finally, a generic antenna design can be used to test receivers in various frequency bands and for different applications. For all examples in this paper a receiver setup for the 2.4 GHz ISM-band is assumed. 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 to gain information about the current environment. For a mathematical description, Section III defines the signal model and presents the simulation and receiver test environment. Afterwards, Section IV 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 V. The final part of Section IV presents some details of the low-cost antenna design and setup. Section V 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 VI, the used measurement setup and measurement results are described to show the potential of the overall receiver hardware/software-concept.

Section VII summarizes the paper and shows the intended optimization steps of the receiver hardware/software (i.e., miniaturization, introduction of new algorithms, introduction of new applications).

II. Problem Definition

IoT-nodes and IoT-node networks suffer from the operation of a large amount of nodes in close vicinity and indoor operation. As discussed in the introductory part this leads to:

- Interference 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 [7]. Therefore, the critical task is to perform a cost-benefit-analysis (e.g., minimal power consumption vs. meeting application defined DoA-estimation accuracy as well as interferer suppression) in short time.

To quickly develop and evaluate IoT-nodes that fulfill the required user demands, performance needs to be observed i.e., the quality of several different DoA-algorithms and 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 arrays under various circumstances and trigger critical estimation edge cases to ultimately develop better or cheaper algorithms and arrays. This research approach is followed by a design and realization phase of the low-cost and low-power sensor, analog and digital signal processing hardware (cf. Figure 2).

III. Signal model and simulation

This section describes the signal model and the simulation setup, as well as simulations results. Furthermore, the main algorithms for DoA-estimation (e.g., Capon-Beamformer and Multiple Signal Classification (MUSIC) algorithm [8]) are introduced and the respective equations are given.

A. Signal Model

In this section the signal model, based on the theory described in [6], [5] and [8], is briefly summarized while the description is restricted to one received signal. We assume that we are in the far field of the sending antenna, the narrow band assumption holds and that the antenna has a flat frequency response. Then the vector $\mathbf{u}$, which might be used to describe signal and interferer, can be defined. Figure 3 shows an arbitrary antenna array with N antenna elements and the vector $\mathbf{u}$.

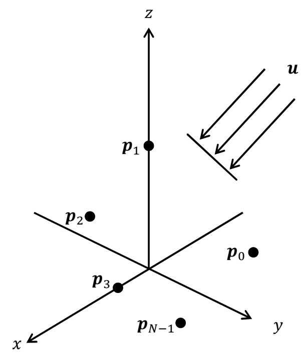


Figure 3. Multiantenna Model

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

$$\mathbf{u}(\phi,\theta) = \begin{pmatrix} -\sin\theta\,\cos\phi \\ -\sin\theta\,\sin\phi \\ -\cos\theta \end{pmatrix} \tag{1}$$

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

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

In the following, the angles ϕ and θ are omitted. If it is now assumed that an N-element antenna (cf. Figure 3) receives this signal from a defined DoA the resulting equation, which describes the time-dependent output vector, is:

$$\mathbf{x}(t) = \exp(-\mathbf{j}\mathbf{p}\mathbf{k})s(t) + \mathbf{n}(t) = \mathbf{a}s(t) + \mathbf{n}(t) \tag{3}$$

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

$$\begin{aligned}\mathbf{R} &= E\{\mathbf{x}(t)\mathbf{x}^H(t)\} \\ &= \mathbf{a}E\{\mathbf{s}(t)\mathbf{s}^H(t)\}\mathbf{a}^H + E\{\mathbf{n}(t)\mathbf{n}^H(t)\} \\ &= \mathbf{a}\mathbf{P}\mathbf{a}^H + \sigma^2\mathbf{I}\end{aligned} \tag{4}$$

Equation (4) can be written using a unitary matrix $\mathbf{U}$ and a matrix of the Eigenvalues $\Lambda = \mathrm{diag}\{\Lambda_0, ..., \Lambda_{N-1}\}$ [9].

$$\begin{aligned}\mathbf{R} &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H \\ &= \mathbf{U}_s\mathbf{\Lambda}_s\mathbf{U}_s^H + \mathbf{U}_n\mathbf{\Lambda}_n\mathbf{U}_n^H\end{aligned} \tag{5}$$

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 [8] this matrix is called $\hat{\mathbf{R}}$.

In this work two DoA-estimation algorithms are considered. First the Capon and second the MUSIC algorithm [8]. Both algorithms generate a spatial spectrum, where the maximum gives an estimate of the DoA of the incoming signal.

For the Capon beamformer, the spatial spectrum is defined as:

$$P_{\mathrm{CAP}} = \frac{1}{\mathbf{a}^H(\phi,\theta)\hat{\mathbf{R}}^{-1}\mathbf{a}(\phi,\theta)} \tag{6}$$

The MUSIC spectrum is defined as:

$$P_{\mathrm{M}} = \frac{\mathbf{a}^H(\phi,\theta)\mathbf{a}(\phi,\theta)}{\mathbf{a}^H(\phi,\theta)\hat{\mathbf{U}}\hat{\mathbf{U}}^H\mathbf{a}(\phi,\theta)} \tag{7}$$

For interferer suppression a simplified version of the Applebaum [5] array will be used.

B. 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 V) to validate the correctness of the real-time C++-based receiver results.

Therefore, 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 (3)). These signals were used as input signals for the C++-based and MATLAB based offline processing, by using a simple file format developed for this purpose.

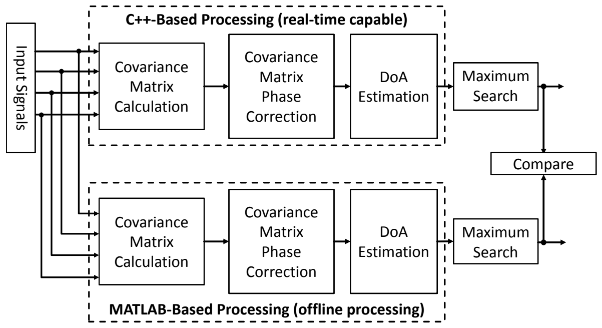

C++-Based Processing (real-time capable)
Input Signals
Covariance Matrix Calculation
Covariance Matrix Phase Correction
DoA Estimation
Maximum Search
Compare
Covariance Matrix Calculation
Covariance Matrix Phase Correction
DoA Estimation
Maximum Search
MATLAB-Based Processing (offline processing)

Figure 4. MATLAB and Real-Time C++ Comparison

Both processing paths in Figure 4 estimate covariance matrix, spatial spectrum, as well as the azimuth and elevation angles using floating-point precision (see Section V 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. The results show that while the spatial spectrum of the Capon Beamformer is slightly degraded the MUSIC-spectrum is identical with a resolution of 1° (see Section V).

IV. 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 task, while the software is described in detail in Section V. Mainly an 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 5). 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 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.

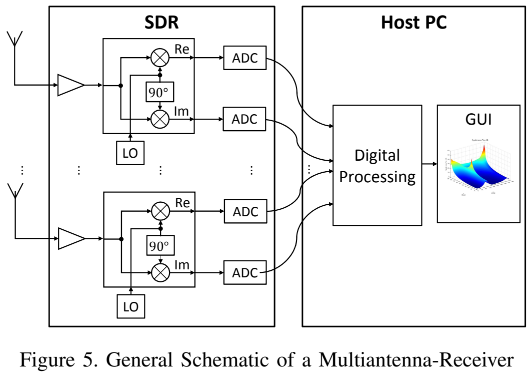


Figure 5. General Schematic of a Multiantenna-Receiver

A. Receiver Hardware-Setup

A general approach of low cost multiantenna receivers for Global Navigation Satellite System (GNSS) receivers has been described in [3]. 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 [3] with a commercially available SDR [10]. This architecture features substantially more flexibility, which comes at significantly higher costs, which are 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 [10]. Each daughterboard has a frequency range from 400 MHz to 4.4 GHz, allows duplex operation, 40 MHz bandwidth and 16-bit ADC resolution. X300 device can be equipped with two daughterboards, therefore a 4 channel SDR-receiver requires four SBX daughterboards and two X300. Each X300 is connected to a host PC using a 10 GBit/s connection.

Figure 5 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, totally unsynchronized LOs will generate totally useless input signals. If the phase offset between the individual LOs is known, they can be easily canceled out by correcting the unwanted phase shift in software. To overcome 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. Measuring results showed that switching over to the synchronization signal each 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 6. 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.

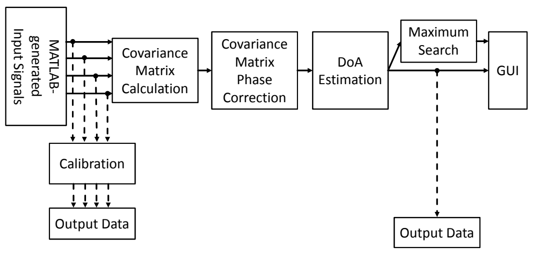

MATLAB-generated Input Signals
Covariance Matrix Calculation
Covariance Matrix Phase Correction
DoA Estimation
Maximum Search
GUI
Calibration
Output Data
Output Data

Figure 6. 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 (c.f. Figure 10). 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 tests in the ISM-band an array with four ground plane antennas has been designed. This type of antenna is low cost and easy to build and allows simplified antenna tuning [11]. Figure 7 shows the VSWR-plot of a single antenna, which shows a minimum at the desired frequency $f = 2.45$ GHz.

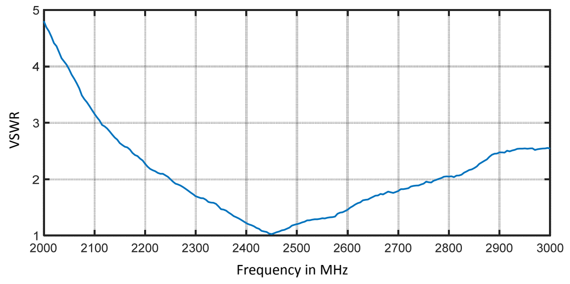


Figure 7. VSWR-Measurement used for Antenna Tuning

The driven element and the four radials do have a mechanical length of around $l = 3$ cm, which is approximately $\lambda/4$ for the selected center frequency of 2.45 GHz. As can be seen in Figure 8 the rectangular array features an inter-element spacing of $\lambda/2 \approx 6.1$ cm. 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 $\phi = 0°$. While this is a prefect setup for ground based signals and interferers it will lead to problems if the desired signals have larger elevation angles.

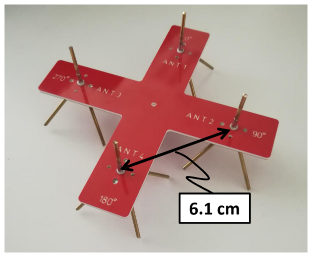


Figure 8. Low-Cost Multiantenna Realization

V. Receiver Software Realization

This section discusses details of the signal processing block realization in Figure 6. 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.

- Optimum real-time DSP performance without any sample-drop combined with an optimal GUI-operation.

Furthermore, the phase-synchronization described in Section IV has to be implemented. To achieve these goals, 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 [12]. Besides the ability of displaying the current 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 using software like MATLAB afterwards (see Section III).

A. *Concurrent Data Processing*

To make use of modern multicore-CPUs and meet the throughput requirements, the work is 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.

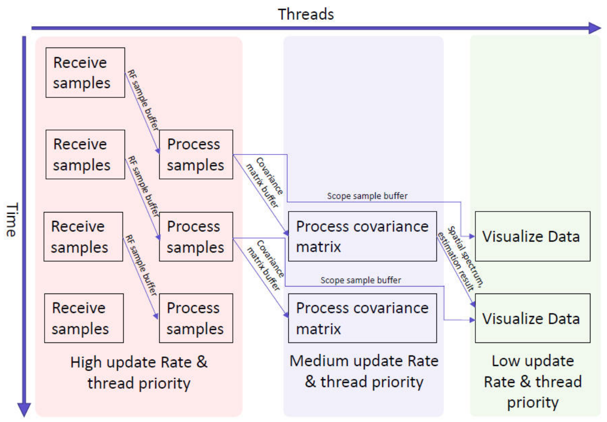


Figure 9. Multithreaded Software Pipeline

Figure 9 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 narrows down 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 passed for processing.

Samples are received by blocking calls to the Ettus UHD API [13], 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 the acquisition of the following sample block in parallel. The sample processing thread fills a buffer for the scope 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 finished, the phase correction factors are applied to the matrix, which leads to much smaller computational overhead, 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 current 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 will be passed.

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 won't 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, e.g.,

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

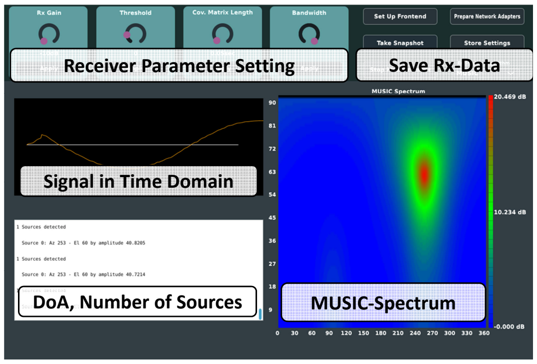
An important feature of C++ is the ability to describe (fully virtual) interface classes. This feature has been to describe a generic DoA-algorithm class, consuming a covariance matrix and generating a spatial spectrum, as well as a pair of estimation vectors that can be overridden by an actual implementation. 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 powerful 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.

VI. Measurement Setup and Results

The described SDR-based demonstrator featuring the low-cost multiantenna array has been used to perform some indoor measurements in the ISM-band at 2.45 GHz. Since multipath and interfering signals are expected in the utilized frequency band the environment close to the real application and the measurement quality will be degraded. Nevertheless, first qualitative results show an accuracy of the DoA-estimation around 5° for the azimuth θ and approximately 10° for the elevation angle ϕ. Moreover, the real-time GUI (cf. Figure 10) shows a correct dynamic behavior. 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 measurement results, and ease software debugging.



```
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
```

Figure 10. Graphical User Interface of the Multiantenna-Receiver

Besides the qualitative test a first profiling has be conducted to evaluate the computational requirements of the three threads shown in Figure 9. The profiling shows that about 53% of the overall processing time is consumed by the GUI and the user interaction (i.e., the green block in Figure 9) while 45% is required for the covariance matrix calculations and the DoA algorithm (i.e., blue block in Figure 9). The high priority thread (i.e., red block in Figure 9) 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 algorithm.

VII. Conclusion and Further Development

Spectral efficiency, robustness and security are critical design parameters of wireless IoT-sensor nodes. 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.

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.

References

[1] Cisco Internet Business Solutions Group (IBSG), “The internet of things,” 2011.

[2] “Number of IoT Devices,” 2018, URL: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/ [accessed: 2018-07-20].

[3] G. 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.

[4] “World Population,” 2018, URL: https://www.populationpyramid.net/world/2025/ [accessed: 2018-07-20].

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

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

[7] 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, 2018, pp. 1–1.

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

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

[10] “Ettus Homepage,” 2014, URL: http://www.ettus.com/ [accessed: 2018-07-20].

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

[12] “Juce Homepage,” 2018, URL: https://juce.com/ [2018-07-20].

[13] “Ettus API,” 2014, URL: http://files.ettus.com/manual/page_coding.html/ [accessed: 2018-07-20].