Implementation of an IEEE 802.15.4 Transceiver with a Software-defined Radio setup

Stefan Knauth
Lucerne University of Applied Sciences
CEESAR – Center of Excellence for Embedded Systems Applied Research
Technikumstr. 21
CH 6048 Horw / Switzerland
stefan.knauth@gmx.ch
www.ceesar.ch

Abstract

An implementation of an IEEE802.15.4 transceiver using software defined radio technology on a standard PC has been developed. The setup is based on the USRP, a commercially available software-defined radio hardware and on the GnuRadio software framework. The software runs on the PC to which the USRP is connected via USB2.0. The setup is ideally suited for education, monitoring, and for a deep understanding of the IEEE802.15.4 physical layer, for example for engineering students. The implemented RX software includes synchronisation, gain adjustment, a differential phase decoder, a precise symbol correlator and a data frame interface for upper software layers. The system may run on two channels synchronously. The software implementation is described with focus on the frame detection and synchronisation, the differential IQ demodulator and the correlator. It operates with an acquisition rate of 4 MSamples, each sample consisting of two IQ values.

Introduction

Wireless functionality is becoming more and more a standard feature of all kinds of electronic embedded systems. As a designer, one typically integrates an OEM wireless module into the own application or one uses a highly integrated transceiver according to a manufacturer-given reference design. By these approaches, one may deploy wireless technologies without detailed knowledge of the underlying radio and communication principles. But If one is interested in the latter, today reading textbooks is no longer the only way of studying such systems. By employing commercially available peripheral electronics, an ordinary PC may be turned into an "any-standard" radio transceiver. The signal processing is performed by free software, which can be studied and modified. Besides the experimental study of the technology, one gets a handy tool to, for example, check the functionality of a radio system and to locate error sources.

Software defined radio overview

"Software Defined Radio" (SDR) is a radio architecture where most signal processing tasks are performed by software instead of analogue electronic circuits. Typical tasks, which have been negotiated by discrete electronic circuity in classic radio setups include channel filtering, mixing, fine tuning, baseband filtering, and demodulation. In a SDR approach, depending on the performance of the available computing platform, all this functionality may be implemented in software. Typical computing platforms are personal computers (PC), microcontrollers, signal processors or FPGAs. Except the PC version, the digital processors may be integrated on a chip together with the analog radio hardware and AD converters,

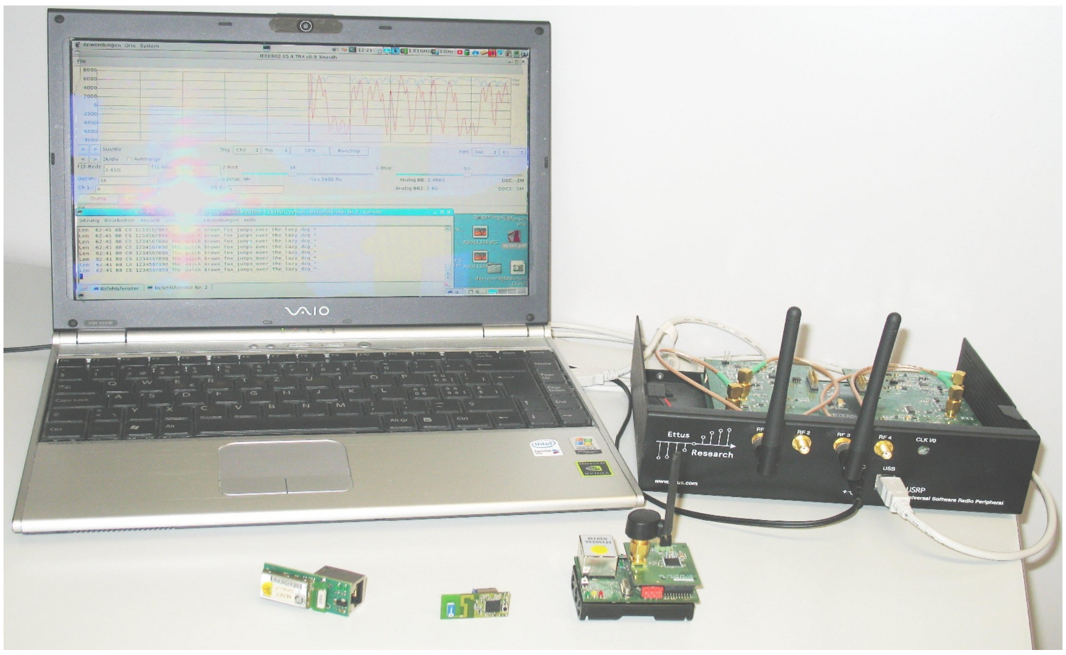


Fig. 1: Software defined radio setup: Notebook (left), SDR hardware (USRP [1], right) and some IEEE802.15.4 / ZigBee modules [2] (foreground)

forming highly integrated SDR solutions. Today, as an outcome of the rapid technology push in the mobile wireless industry SDR has practically ousted classical radio approaches in most mass market wireless applications.

The simplest SDR setup consists just of an antenna connected to the input of an AD converter, the latter being connected to a computer. AD converters with rates in the range of 100 MHz and resolutions of 12 bit are < 10 EUR mass marked products today. In order to capture a radio wave by the converter, the wave must be sampled once per half-wave (Nyquist-Theorem). Accordingly, the simple setup does already work as a simple receiver for the radio bands up to 50 MHz, including the long wave, so-called "AM" and short wave radio bands. Also, the receiver may demodulate and for example record all stations simultaneously. But when regarding the simultaneous reception of different transmitters, a limitation of the SDR principle becomes obvious: a linear 12 bit AD converter provides a dynamic range of 36 dB. Since the signal strengths of all transmitters which shall be received have to lie in this 36 dB range, only the strongest transmitters can be detected. Therefore in practice even a "very digital" receiver design will incorporate a channel filter or at least a band filter. For common UHF and SHF frequencies, that is the range of some 100 MHz to above 10 Ghz, typically also an analog mixing stage is employed for reducing the radio frequency to a frequency which can be sampled by the converter.

A software defined radio setup for use with standard PC's

In order to convert a standard PC into a SDR setup, one needs an AD converter and a HF frontend circuity, as mentioned earlier. Also suitable signal processing software is needed. A commercial example of such a PC hardware is the "Universal Software Radio Peripheral" (USRB) by Ettus Research [1]. Fig. 1 shows the setup consisting of a Notebook (left), the USRP (right) and some IEEE802.15.4 transceiver modules (foreground) manufactured at our Lab [2]. In figure 2 a simplified schema of the USRP is displayed: when operating in receive mode, the radio signal propagates from the antenna (ANT) via the antenna switch (TX/RX) to a low noise preamplifier (LNA). Here the signal is band-filtered and amplified. A PC-manageable reference oscillator creates a quadrature signal, that is two signals of the same frequency, but with a phase shift of 90°. The frequency of the signals shall be the desired receiver frequency. The two reference signals are then multiplied with

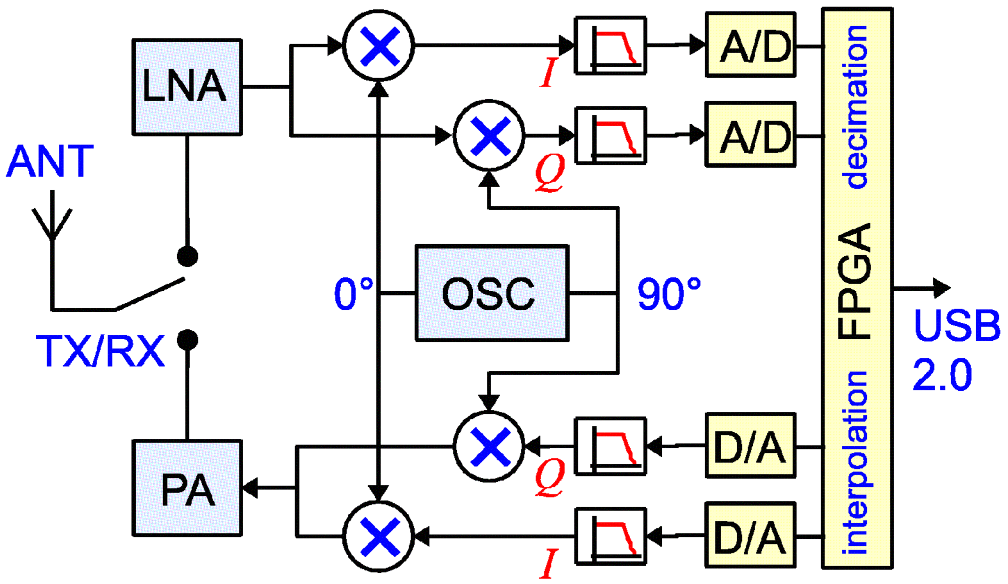


Fig. 2: A simplified schematic of a software defined radio hardware

the amplified radio signal and the output of each multiplier is low-pass filtered and aliasing-filtered before it is digitized with an AD converter. The setup with multiplying of the received signal with its own center frequency leads to a resulting baseband frequency of 0 Hz and is a typical SDR receiver technology which is also referred to as "direct downconversion receiver". The demodulation with two signal path with a 90° offset is called quadrature- or IQ demodulation. By perceiving of the I- and the Q channel all kinds of demodulation can be performed, for example amplitude shift keying (ASK), frequency shift keying (FSK) or phase shift keying (PSK).

The USRP does not actually operate on a 0 Hz baseband frequency on the analog side, but the software programs the OSC such that it will be some MHz shifted away from the desired center frequency, and later applies a second multiplying to correct the baseband signal on the digital side in the FPGA. This enables automatic offset elimination in- and gain equalization between the two RX paths, and also allows the PLL of the OSC to have a high bandwidth and have a high spacing between lockable frequencies.

The transmission path of the USRP is reverse to the RX path: Also for transmission, the signal may be composed of a 0° and a 90° fraction (I and Q Channel), thus allowing easy implementation of PSK and FSK. The signal of each D/A converter passes an aliasing filter and gets multiplied with its respective OSC channel (I respective Q). The output signals of the multipliers are added, filtered and passed to the power amplifier (PA), from which the radio signal reaches the antenna via the TX/RX switch.

The FPGA, among others, may perform a data rate reduction on the receive side, with respective filtering and bandwidth reduction, and a data rate increasal on the sender side. The maximum data rate across the USB2.0 interface is 16 millions IQ sample pairs, when in 8 bit mode. This corresponds to a RF bandwidth of 16 MHz (+/- 8 MHz).

The USRP is supported by the software collection and framework of the GnuRadio project [3]. GnuRadio is a community project licensed under the GPL and is integrated into mayor GNU/Linux operating system distributions, for example Debian and Ubuntu, but also runs on Windows and MAC OS. With the help of the sample applications included in the software distribution, the USRP can be quickly converted into an Oscilloscope, a spectral analyser or an FM radio. Numerous further examples exist.

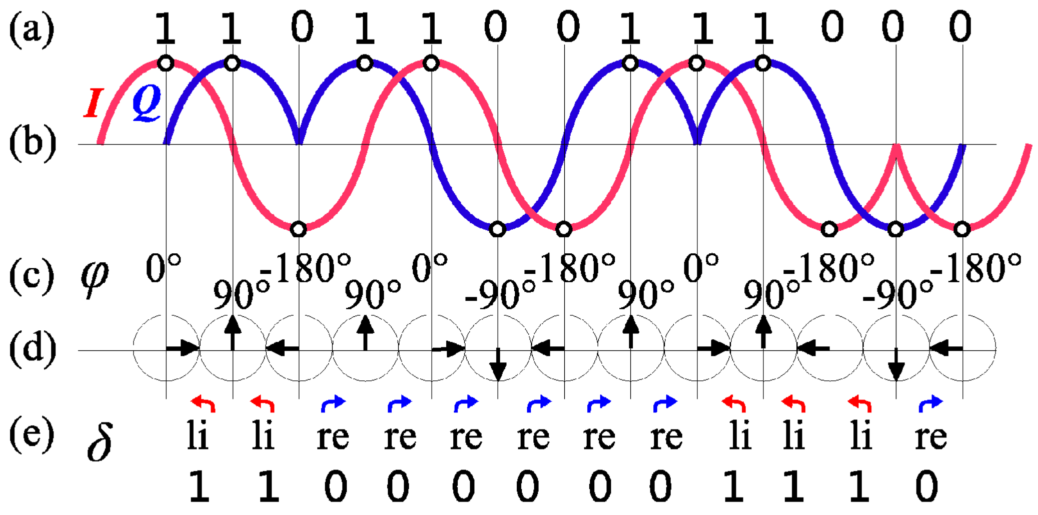


Fig. 3: IEEE802.15.4 O-QPSK modulation scheme. The beginning of the symbol "0" is shown. Being also the synchronisation symbol, it leads every RF packet .

IEEE802.15.4 QPSK Modulation

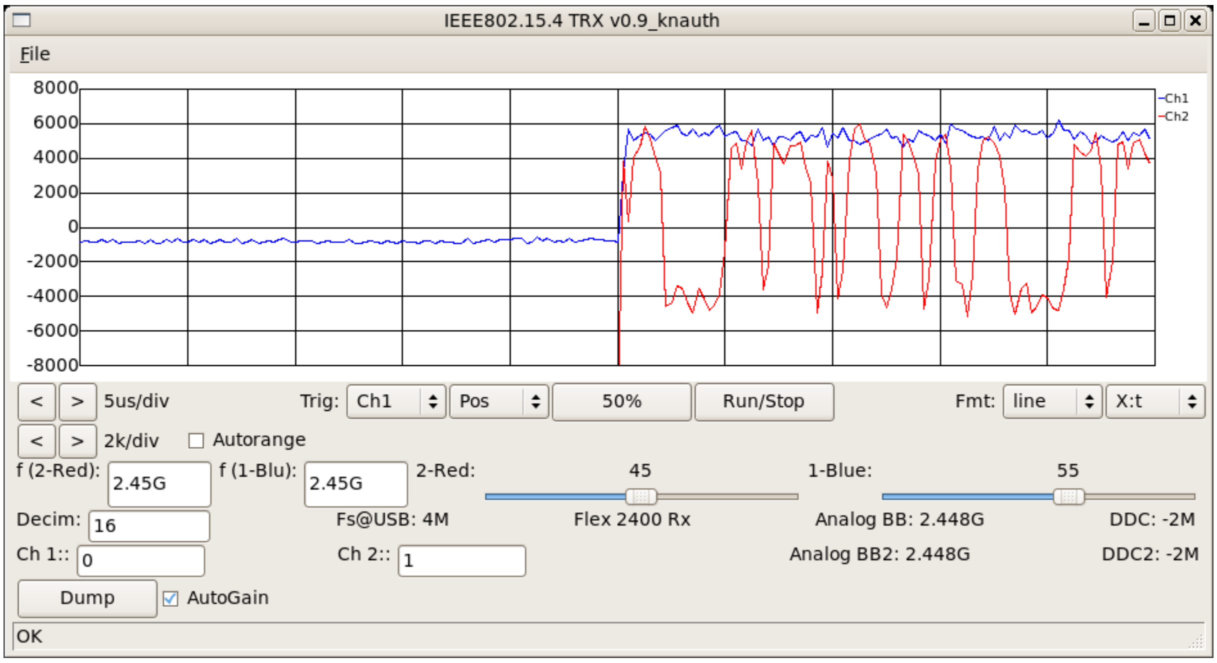
The IEEE802.15.4 standard [4] defines a low rate personal area network technology, which has become quite popular in the last years and also forms the lower layer of the ZigBee network protocol. There are already several system-on-chip solutions available which include a microcontroller and an 802.15.4 compliant radio transceiver [5]. From the modes described in the standard, the mostly employed one is the 2.4 GHz O-QPSK ("Orthogonal quadrature phase shift keying") modulation (see fig. 3). This phase modulation is achieved by alternately driving the I- and the Q component of the carrier frequency. A bit value is represented by the sign of the corresponding I- or Q component. "Quadrature" here means that 4 phase values are used for the signal transmission and "Orthogonal" means that the I- and Q bits are not transmitted at the same time but shifted against each other by half of the chip duration. This is visualised in figure 3: In the upper row (a) a bit sequence is displayed which shall be transmitted. Apparently this is the beginning of the "0" symbol, as defined in the standard. The bits at odd positions (1,3,5...) control the I-channel, and the even bits controll the Q channel, as sketched in fig. 3(b). A binary "1" is transformed into a positive $\sin^2$ excitation, a "0" into a negative $\sin^2$ excitation. During the maximum drive of each channel, the other channel is zero. Row (c) displays the resulting carrier phase, and row (d) shows it as angle of an arrow in the IQ-plane. The phase angle is generally given by the arc tan function, for the right quadrants this is

$$\varphi = arctan(\tfrac{Q}{I}), \text{ for } I > 0$$

The I-bits create phase values of 0° respectively -180°=+180°, and the Q-bits create phases of 90° respectively -90°. The phase change during each transmitted I- or Q half-chip (500 nanoseconds) is always either +90° or -90°, depending on the value of the current and the last bit. When displayed in the IQ-Plane, the phase change results in a 90°-rotation of the arrow in row (d), either clockwise or counter-clockwise, as shown in row (e).

Implementation

The software implementation of the IEEE802.15.4 transceiver in the GnuRadio framework adopts an existing program, the "OSCOPE" oscilloscope demo, and the "myblock" demo on writing own signal processing blocks. The GnuRadio signal processing is based on a "pipes



IEEE802.15.4 TRX v0.9_knauth

File

< > 5us/div Trig: Ch1 Pos 50% Run/Stop Fmt: line X:t

< > 2k/div Autorange

f (2-Red): 2.45G f (1-Blu): 2.45G 2-Red: 45 1-Blue: 55

Decim: 16 Fs@USB: 4M Flex 2400 Rx Analog BB: 2.448G DDC: -2M

Ch 1:: 0 Ch 2:: 1 Analog BB2: 2.448G DDC2: -2M

Dump ✓ AutoGain

OK

Fig. 4: Screenshot of the transceiver software. The amplitude (red) and the phase derivative (blue) of a frame start is displayed.

and filters" architecture. Besides a control loop and an optional GUI, a GnuRadio application consists of a signal processing chain, that is signal processing classes derived from a standard signal processing class, and a description of the signal flow. The signal processing classes may be signal sources, signal sinks and signal processors, which each may have 0 or more inputs and 0 or more outputs. An application contains information on how the classes are connected together, forming a signal processing graph. During runtime, the data stream of the SDR hardware is split into chunks of some kSamples length, and these chunks are then passed sequentially to the signal source classes representing the SDR hardware. A chunk overlap may be requested to ease the implementation of, for example, FIR filters. The graph description defines the sequence of the signal processing.

The implementation of the IEEE802.15.4 transmitter is straightforward: the frame data is first enriched with synchronisation bytes, start-of-frame and length byte, and with the checksum. Then the bytestream is parsed 4-bit-wise and converted into a symbol stream. Finally each symbol is replaced with a 32 bit spreading code, which is alternating fed into the I- and the Q stream, as defined by the IEEE802.15.4 standard. This spreading allows the symbols to be detected even when there is a considerable bit error rate of 10% and more. The resulting I- and Q bitstream has a rate of 1 MSamples per second each, and is converted into the $\sin^2$ values using an interpolation factor of 4 values each. The Q stream is delayed by 500 nsec as defined by the standard and explained in fig. 3. Since there are only 4 values of the $\sin^2$ function used, they are kept as constants and no computation is necessary. Finally the interpolated IQ stream with a rate of 4 MSamples is transferred to the FPGA for RF processing and transmission.

The receiver implementation is more complex because the IQ signal from the receiver can not directly be used to reconstruct the IQ values of the transmitter. The reason is that the receiver does not know the unmodulated carrier phase of the transmitter. Furthermore, there typically exists a small frequency shift between the transmitter and the receiver. based on the crystal tolerances of the oscillators. A common approach to detect PSK signals is the deployment of a circuity known as „Costas-Loop“. This is a special PLL circuity which, by advanced averaging methods reconstructs the original carrier signal. In a SDR setup such methods may also be implemented in software. In the special case of O-QPSK, there exist also a much easier solution, the differential phase demodulation. Here one does look only at

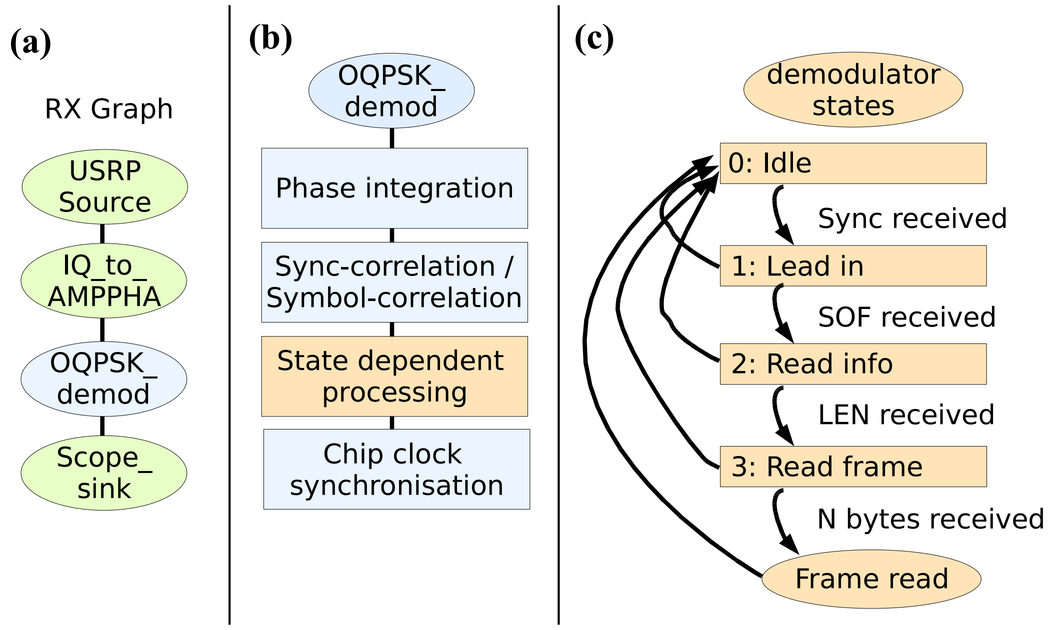


Fig. 5: Software overview of the RX path: (a): Graph showing signal processing classes and flow of operation. (b) Tasks of the demodulator class. (c) Demodulator state machine.

the phase change between two samples and therefore no information on the absolute phase is needed. As we know from fig. 3 (e), between two subsequent half-chips the phase difference in the O-QPSK is always +90° or -90°. The phase value can be obtained from the IQ values for example via a lookup table, and the needed accuracy of the phase value is low. By subtracting two subsequent phase values one obtains the differential phase value. A value of +90° represents a 1, a value of -90° a 0 bit. In Figure 4 a screenshot of the IEEE802.15.4 software is shown. The blue channel corresponds to the received amplitude, which should stay more or less constant during reception of a data frame. The red trace shows the differential phase signal. It is the start of the bit sequence already shown in fig. 3 (b).

The graph describing the classes and connections for the RX path of the IEEE802.15.4 transceiver software is outlined in figure 5a. The USRP source class receives the digitized data from the RF hardware and provides it to the IQ_to_AMPPHA class. Here the IQ raw data is converted into phase and amplitude data. Provided a fast enough (1.8 GHz dual core) processor, this can be done by floating point arithmetic. But finally a lookup table (16 bit address, 8 bit for I and 8 bit for Q) has been used, thereby of course ignoring 6 further available bits from the decimator. No significant reduction in detection accuracy of the despreading algorithm has been identified, as long as the gain adjustment was accurate enough. The 2.4 GHz receiver board of the USRP does not provide an "inner loop" automatic gain adjustment, and therefore the gain was adjusted by the detected symbol signal level, but the high loop latency did not allow to adjust the gain within the length of a single frame.

As already explained, the O-QPSK demodulator (Fig. 5(b)) works on the phase difference. The difference is then integrated again up to the half-chip duration of 500 nsec. The correlation procedure is dependent on the receiver state. If there is no packet reception in progress, it is looked for a synchronisation sequence. This is performed by shifting the differential bits through a shift register and comparing it with the differential symbol 0 signature, which is "E077AE6C". If a configurable number of bits confirms to this signature, the state machine recognizes a synchronisation event and switches to the further reading states. Now the demodulator does not look any more for the sync bits, but performs a proper correlation with the 16 differential sequences by summing over the phase difference values multiplied with a sign indicated by the bit value of the differential symbol. (The 16 differential sequences are derived from the 16 original spreading sequences of the standard). The symbol with the highest correlation value is selected as the received symbol. If the correlation value lies below a configurable threshold, the frame reception is cancelled and the state machine returns to the idle state. The "successful" correlation value is also compared with the one obtained for the symbol when the correlation starts one sample earlier or later. If the value for the shifted correlation is higher, an adjustment sample is inserted or removed from the stream, thereby reaching a continuous synchronisation with the transmitters chip rate.

Conclusion

It has been shown that an IEEE802.15.4 transceiver can be successfully modelled using the USRP radio hardware and the GnuRadio software, and implementing the needed signal processing functionality including spreading, IQ generation, phase detection and differentiation, start detection, symbol correlation, and symbol synchronisation. The system and application is a powerful tool for the experimental study of modern radio technology and it is used as a versatile test- and monitoring tool for IEEE802.15.4 systems. An open but interesting point yet is the implementation of further receiver algorithms and quantitative comparison of these. The author is scientist at the CEESAR competence centre (www.ceesar.ch) at Lucerne University of Applied Sciences. CEESAR performs applied research with industry partners, especially in the areas building automation and energy management and will soon open the iHomeLab (www.iHomeLab.ch), a laboratory home to investigate- and present trends in modern homes.

Contact CEESAR

Prof. Alexander Klapproth
Head of CEESAR
Lucerne University of Applied Sciences
Switzerland

Tel.: +41 41 349 35 99
EMail: mailto:info@ceesar.ch
Web: http://www.ceesar.ch   http://www.iHomeLab.ch

Acknowledgements

This work is funded by KTI/CTI, Berne, Switzerland and by the Hasler Stiftung, Berne, Switzerland.

References

[1] Ettus Research: www.ettus.com

[2] CEESAR ZigBee radio modules see: www.ceesar.ch

[3] The GnuRadio Project homepage: gnuradio.org/trac

[4] IEEE Standard 802.15.4: www.ieee.org
802.15.4 (TM) IEEE Standard for Information technology Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), Published by The Institute of Electrical and Electronics Engineers, Inc. 3 Park Avenue, New York, NY 10016-5997, USA

[5] Examples for microcontrollers with integrated IEEE802.15.4 radio:
Texas Instruments CC 2430, Ember EM250, RadioPulse Mango MG2400