On the Design of Software and Hardware for a WSN Transmitter

Jo Verhaevert, Frank Vanheel and Patrick Van Torre

University College Ghent, Faculty of Applied Engineering Sciences,
Schoonmeersstraat 52, B-9000 Ghent, Belgium

Tel.:+32 9 248 88 22, Fax:+32 9 243 87 77, {jo.verhaevert, frank.vanheel, patrick.vantorre}@hogent.be

Software defined radios (SDR) are booming. However, for a final breakthrough these systems need to be versatile, inexpensive and easy to program. In this paper a next step is taken to meet all these requirements. Our hardware consists of a computer with an affordable data acquisition (DAQ) card and a cheap self-made single-stage up-converter. The software is written in the slow learning-curve graphical programming environment LabVIEW. To prove the versatility of our SDR transmitter concept, we send packets with the wireless sensor networks (WSN) protocol IEEE 802.15.4, which are received by an existing packet sniffer.

Introduction

Over the last decade a great number of new standards have appeared and all of those need to be implemented and verified in some way. Software Defined Radio (SDR) is not only a good answer, but also a versatile and efficient solution for future upcoming standards. In SDR, some hardware components are replaced by software, such as mixers, filters, modulators and amplifiers. Other components are impossible to be realised in software, for example some parts of the physical layer, and are therefore implemented in hardware. Existing SDR, like the GNU open source radio project [1], is versatile, but the available hardware is expensive. We decided to build our own hardware and hence also own software, resulting in a loss of flexibility and versatility. Necessary design tips are described. Here, we build the transmitter pushing both the software and the hardware to their limits. Although other modulation techniques and hence standards are possible, the 2.45 GHz ISM frequency band is selected in this paper, together with the Wireless Sensor Networks (WSN) protocol IEEE 802.15.4 [2].
This paper is organised as follows. A first section will describe the used hardware, whereas the following section will give indications for the implementation of the IEEE 802.15.4 protocol in software. In a next section, results will be described showing that everything works fine. Finally, in the last section conclusions will be drawn.

Hardware

The used hardware is based on three main parts: a standard PC, a data acquisition (DAQ) card and a self-made up-converter printed circuit board (PCB). Figure 1 gives a schematic view. The PC (with LabVIEW [3] as programming language) generates baseband signals, which are converted via the PCI-bus of the PC through the DAQ card into analog signals. In this work, a National Instruments PCI 6110 (or DAQmx card) (shown in Figure 2a) [4] has been chosen, which contains both analog and digital inputs and outputs. If two analog outputs were used, a sample rate of 2.5 MS/s could be achieved.

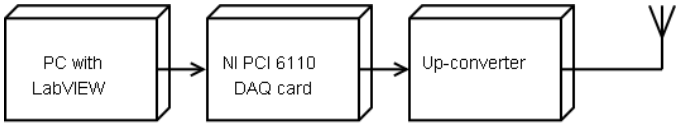
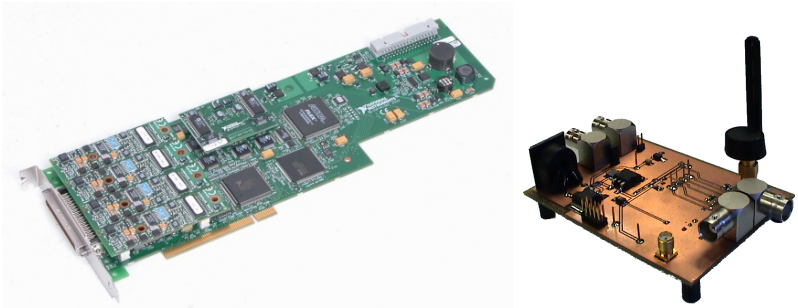


Figure 1: The block scheme of the used hardware.

The up-converter is used to shift these signals to the 2.45 GHz ISM frequency band, tunable by the software and generated with a classical digital Phased Locked Loop (PLL) (which will be explained later). The PCB also contains a quadrature modulator, responsible for the mixing of the I- and Q-signal and for the frequency shift to an RF signal. In this design, an AD8349 [5] is used. This high performance quadrature modulator is used as a single-stage up-converter with a high output power and very low noise floor. Because the modulator requires differential signals, whereas the output of the DAQ card is single-ended, two additional differential amplifiers are integrated. A picture of the design can be found in Figure 2b, where most of the connectors are used for debugging purposes.



(a) (b)

Figure 2: The used hardware: the DAQmx card (a) and the transmitting PCB (b).

Like already said, the PCB also contains a classical digital PLL, which is programmed via the digital outputs of the DAQ card. This Phase Locked Loop is a tuning system with a closed loop that generates an output signal as function of the frequency and the phase of the input signal. Usually, this is done automatically by changing the frequency of a Voltage Controlled Oscillator (VCO), resulting in a signal with the same frequency and phase as the input signal [6].
In the following paragraphs the design and the operation of the PLL are described, following the design rules suggested in [7]. As can be seen in Figure 3, the input signal is a clock signal which is a crystal oscillator in this design. This input signal will be compared with the divided output signal in the phase comparator (which is part of the frequency synthesizer). In order to generate high frequencies, this division is necessary. When the phase and/or the frequency of both signals differs, the frequency synthesizer generates an output signal. It will be higher if the frequency/phase is too small and lower if frequency/phase is too large. In order to reduce the jitter coming from glitches of the charge pump in the frequency synthesizer, this signal is also filtered with a low pass filter. The high frequencies of the glitches are hence cut off with the filter. The input signal changes pass rather fast through a filter with a large bandwidth, resulting in an unstable PLL. A trade off between a fast and a stable PLL should be made. For our application, a fast PLL is not necessary (it is only used to change the carrier frequency), and hence

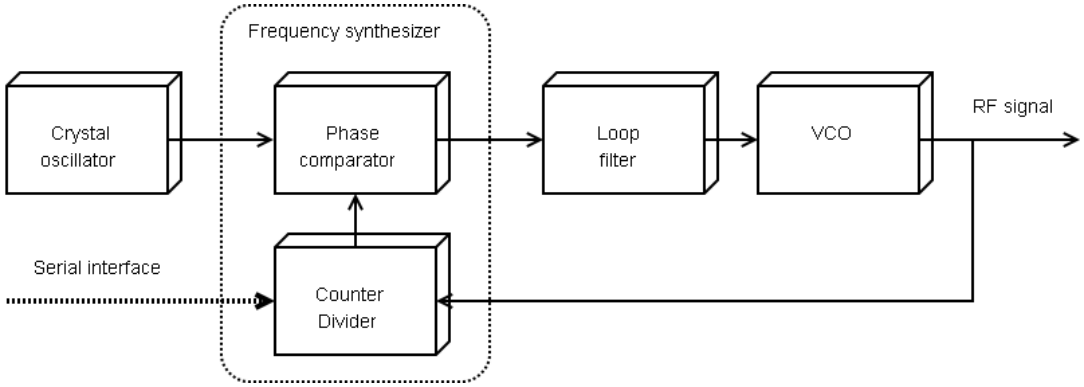


Figure 3: The block scheme of the PLL.

a second order filter with a limited bandwidth of 144 Hz is selected. After filtering this signal in the loop filter, the Voltage Controlled Oscillator (VCO) gives the output signal. The chosen VCO is the CVCO55BE-2400-2670 [8], where a frequency of 2.3 GHz to 2.67 GHz can be obtained using a tuning voltage of 0 V to 15 V.
In fact, here is ADF4113HV [9] the used frequency synthesizer. It is a chip with several programmable counters, a phase detector and a charge pump. The counters are programmed by the integrated serial interface, via the LabVIEW software and the digital outputs of the DAQ card. This solution gives us the possibility to generate highly accurate a frequency in a great range. The frequency synthesizer requires therefore an as stable as possible oscillation frequency of 5 MHz to 150 MHz. Hence a standard crystal oscillator of 10 MHz is selected. An external reference signal is also possible, but this is a less flexible solution.

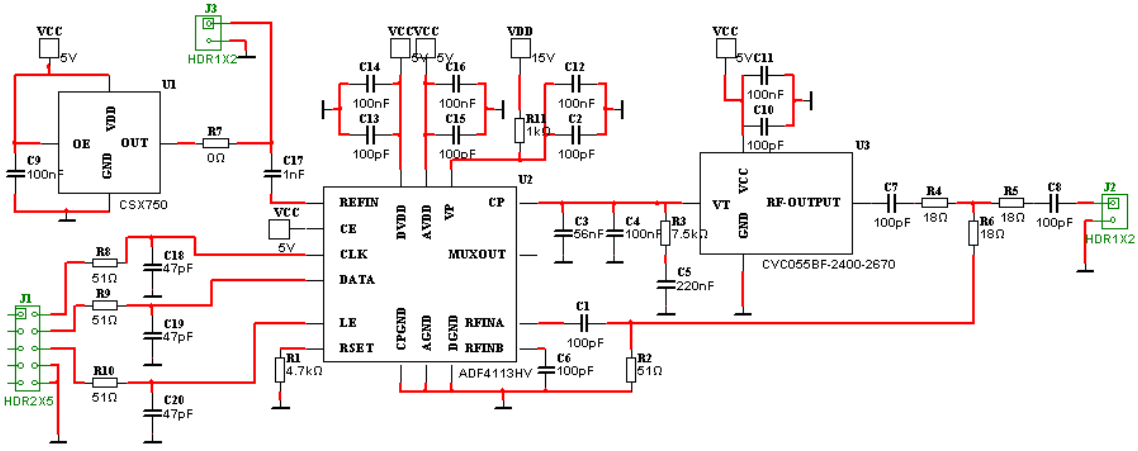


Figure 4: The electronical design of the PLL.

The final electronical design of the PLL with all necessary peripheral components is shown in Figure 4. This is also realised in a part of the transmitting PCB in Figure 2b.

Software

Also in the software some additional design steps should be taken, which are described in this paragraph. As already said in the introduction, we are focusing here on the IEEE

802.15.4 standard in the 2.45 GHz frequency band. The binary information to be transmitted is in a first step translated into different symbols (as indicated in Figure 5) and in a next step into chips (chip length 32) with predefined PN sequences. The standard also describes for every chip bit a half-sine pulse shaping, in order to reduce the effective bandwidth. A side effect is the additional side lobes, which need to be filtered in the final signal in order to lower the power in the other frequency bands. Finally, the modulation scheme is O-QPSK (Offset Quadrature Phase Shift Keying), where the offset is necessary to reduce the amplitude fluctuations between the 4 different phases and hence 4 different symbols. The offset is realised by an additional time shift of a half symbol length between the I- and Q-signal, resulting in the fact that both signals cannot change in the same time slot and hence reduces the maximal phase transitions to 90°. All those steps are implemented in LabVIEW and fed into the DAQ card. The outcome of the software is two signals (I and Q) and serves as input for the transmitting PCB.

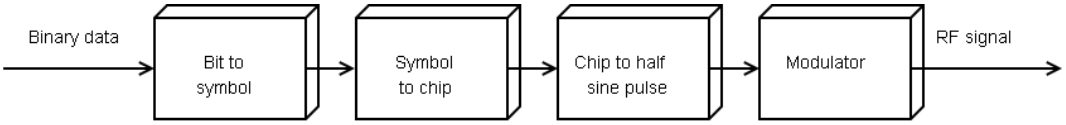


Figure 5: The block scheme for an IEEE 802.15.4 signal.

In order to validate the already taken design steps, the software is written to generate a simple ACK (Acknowledgment) frame. It contains 3 main parts: a SHR (synchronisation header), a PHR (physical layer header) and a PSDU (physical layer service data unit). The SHR is used for synchronisation and contains a preamble sequence of 32 binary zeros and a fixed sequence as start-of-frame delimiter of 1100101. The PHR gives the frame length in bytes of the following PSDU, which is limited to 127 bytes (resulting in a maximal length of 7 bits). An ACK frame always contains 5 bytes, hence the PHR is 0100000. For an ACK frame, the PSDU only contains a header and a footer. This header starts with 3 bits frame type (which is 010 for an ACK frame), followed by the security bit (for an ACK frame always 0) and a frame pending bit (no additional frames are expected for an ACK frame, hence 0). Then there is a bit reserved for an ACK frame request, which is for an ACK frame of course equal to 0. The intra PAN id indicates that the frame remains in the same network, which is always the case for an ACK frame and hence results in 1. Normally, this is followed by the address fields, but for an ACK frame no addresses are included. Instead, a sequence number is added. In our application a random sequence number is selected. The PSDU footer is a frame check sequence (FCS) based on a 16 bit CRC (Cyclic Redundancy Check).

Results

In Figure 6, both the I- and Q-signal are plotted as function of the time, before any pulse shaping. This signal is a software generated ACK frame like described above and measured at the output of the DAQ card.
From both I- and Q-signal, the according spectral behaviour can be generated with a spectrum analyser. There is at this stage no up-converter involved, resulting in a baseband signal. For the I-signal it is plotted in Figure 7a and for the Q-signal it is in Figure 7b.

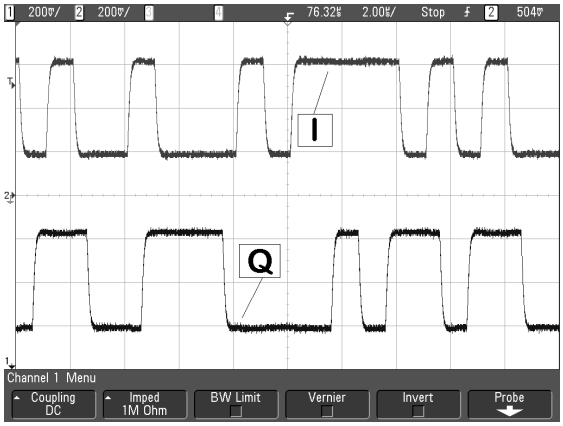
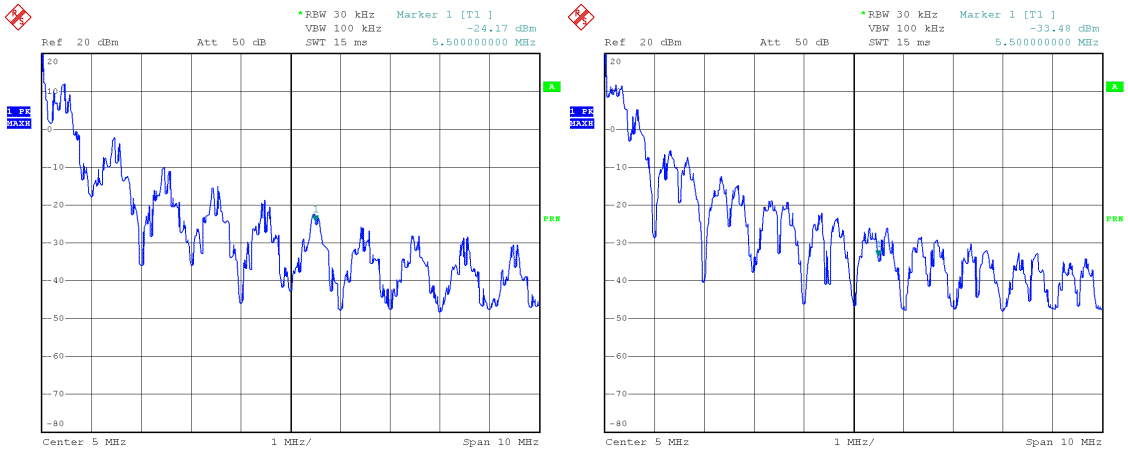


Figure 6: I-signal and Q-signal as function of the time.



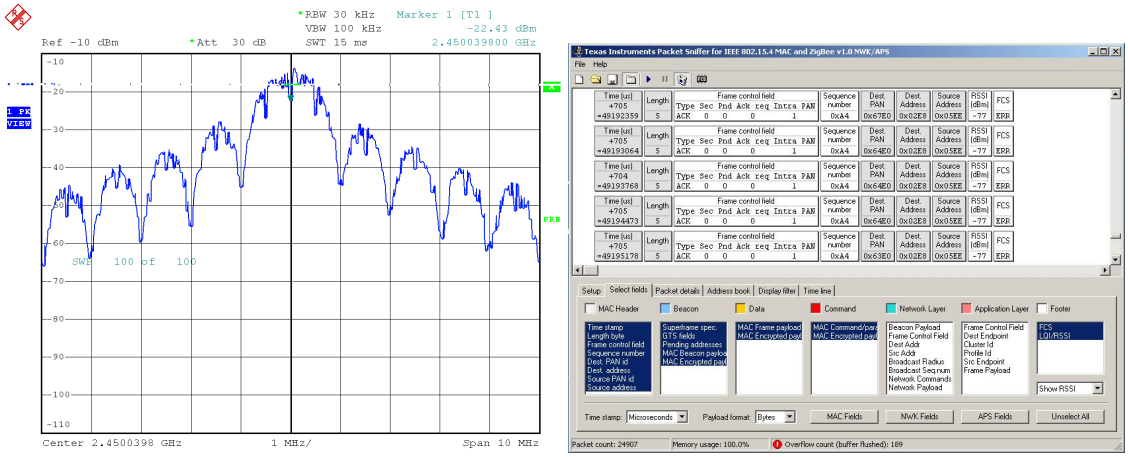
(a) (b)

Figure 7: Spectrum of the I-signal (a) and Q-signal (b).

The signals of Figure 7 are used as inputs of the transmitting PCB. This up-converter mixes and modulates both signals. The spectral output is given in Figure 8a, which shows that the bandwidth is exactly 2 MHz, meeting the prescriptions of the standard. Please note that in these figures, the side lobes of the signal have not yet been filtered.

In order to check the validity of a transmitted data packet, we are using a commercially available packet sniffer. The used hardware is the CC2420DK Development Kit from Chipcon [10], while the packet sniffer software is freely available. In Figure 8b, transmitted acknowledgment (ACK) frames, like described above, are sniffed. For measurement purposes, different ACK frames are repeated. Please note that the frame check has not yet been implemented correctly, resulting in an error. The packet sniffer software also gives the RSSI (Received Signal Strength Indicator), as an indication of the received power. The distance in the lab measurements was limited to approximately and a fix 2 m, resulting in rather large and fix values for RSSI.

In order to not disturb the neighbouring channels, two constraints for the transmitted power spectral density should be considered. One constraint (relative limit) is that the power of frequencies differing more than 3.5 MHz should be 20 dB lower than the car-



(a) (b)

Figure 8: The spectral output (a) and the Chipcon packet sniffer (b).

rier frequency and the other constraint (absolute limit) should be lower than −30 dBm. According to Figure 8a, both the absolute and the relative limit are fulfilled. This means that no power is lost in frequencies outside the considered band and that our design is according to the standard.

Conclusion

The design of hardware and software for a WSN transmitter is described here. Based on a DAQmx card (for the necessary I- and Q-signal) and a transmitting PCB, which mixes and modulates the signals, an RF signal is generated. We also proved with time and frequency domain figures and with real transmitted acknowledgment frames that everything works.

References

[1] GNU Radio, [Online], Available: http://gnuradio.org/trac

[2] IEEE 802.15 Wireless Personal Area Networks, IEEE Standards Association, [Online], Available: http://standards.ieee.org/getieee802/802.15.html

[3] LabVIEW, National Instruments, [Online], Available: http://www.ni.com/labview

[4] PCI-6110/6111 Specifications, National Instruments, [Online], Available: http://www.ni.com/pdf/manuals/370980a.pdf

[5] AD8349, Analog Devices, [Online], Available: http://www.analog.com/static/imported-files/Data_Sheets/AD8349.pdf

[6] Phase-locked loop, [Online], Available: http://www.national.com/AU/files/PLL_Building_Blocks.pdf

[7] Roland E. Best, "Phased-Locked Loops, Theory, Design and Applications", Second Edition, 1993, McGraw-Hill, ISBN 0-07-911386-9

[8] CVCO55BE-2400-2670, Voltage Controlled Oscillator, Crystek Microwave, [Online], Available: http://www.crystek.com/microwave/admin/webapps/welcome/files/vco/CVCO55BE-2400-2670.pdf

[9] ADF4113, RF PLL Frequency Synthesizers, Analog Devices, [Online], Available: http://www.analog.com/static/imported-files/data_sheets/ADF4113HV.pdf

[10] Quick Start Instructions CC2420 Development Kit, Chipcon, [Online], Available: http://focus.ti.com/lit/ug/swru044/swru044.pdf