Software Defined Modem Development for Narrowband PLC Systems

Martin Sigle, Bruno Ruprecht, and Klaus Dostert
Karlsruhe Institute of Technology (KIT) - Institute of Industrial Information Technology - Karlsruhe, Germany

Abstract

In this paper the usability of software defined transceivers for narrowband power line communication (PLC) in the frequency band up to 500 kHz is evaluated. For this purpose the open and free available software platform GNU Radio [1] was adapted and a hardware platform meeting the specific needs of a PLC transceiver was developed. The flexibility of the toolset is demonstrated by setting up a modem which acts as bridge between a proprietary modem using frequency shift keying (FSK) and a system using orthogonal frequency division multiplexing (OFDM). Finally effects of the overall system latency introduced by the software defined approach are discussed.

Index Terms

Power Line Communication, Narrowband, software defined radio, rapid prototyping, smart grid applications

I. INTRODUCTION

Power line communication is meant to be a key communication technology for smart grid applications. Especially for narrowband PLC (NB-PLC), i. e. communication in the frequency range up to 500 kHz, there are currently a number of proposals of different technologies for standardization. However there are still open questions and research is still ongoing. Especially the highly dynamic and varying channel conditions on power lines [2] requires tests in the field, already in a very early stage of the development process. A very versatile platform for implementation is needed for this purpose. In wireless communications, software defined radio (SDR) platforms get more and more popular for this task. The idea behind this approach is to implement as much functionality as possible in software using very general hardware. The software itself is dedicated to run on a general purpose processor (GPP). This leads to a high level of flexibility and portability. Furthermore the programming can be done in a high-level language. Since the software is not written for a specific hardware, components can be easily reused. Another important benefit, especially in development stage, is the simple option to probe and analyze any signal in the processing chain. Additionally the software defined approach offers also a great opportunity to implement cognitive strategies in NB-PLC applications. One popular SDR toolkit is the freely available GNU Radio platform [1]. The benefit of the open and standardized framework is that there are already a lot of components available and common non-proprietary development tools can be used. In the first part a short introduction of the GNU radio framework and our developed hardware platform will be given. In the second part we present an extension of the toolkit which adds the capability of synchronizing the system to the zero crossings of the mains voltage. This is a prerequisite for many PLC systems and even more for channel analysis. In the third part results of a case study - the implementation of an FSK-OFDM bridge - are presented. Finally effects of the overall system latency introduced by the software defined approach are discussed.

II. SDR DEVELOPMENT PLATFORM

A. Software

Having a look at the GNU radio framework from top to bottom, any application written for the framework consists of a set of blocks and a description of the signaling paths between the blocks. The tool used for signaling paths description is python, a high-level object-oriented scripting language. The framework which is controlling the actual data flow between the blocks as well as the high performance parts, i. e. the actual signal processing blocks, are programmed in C++ and compiled to libraries. The connection of both worlds is done by using the Simplified Wrapper and Interface Generator (SWIG). The connection to the outside is done by using special blocks called sinks and sources. These can be real hardware, a file, or even interfaces to connect via UDP or TCP. Already hundreds of blocks are publicly available ranging from simple operators up to blocks which do the entire modulation or demodulation task. For the configuration and connection of blocks a graphical user interface called GNU Radio Companion (GRC) is available (Fig. 4 depicts a block diagram generated with GRC).
B. Hardware

We have developed a general purpose hardware for sampling and reconstruction which is depicted in Fig. 1. The analog part consists mainly of a transceiver chip, a power frontend, a filter circuit in the receive path, and coupling circuits for sending and receiving respectively. Additionally, a circuit for the detection of the zero crossings is included. The digital part comprises a field programmable gate array (FPGA) which is doing mainly decimation and interpolation operations and a universal serial bus (USB) transceiver chip which interfaces the FPGA with a PC. The overall system is adapted to the architecture of the universal software radio peripheral version 1 (USRP1) [3]. Hence the firmware (USB transceiver and the FPGA) for the USRP1, which is also freely available, can be used with only few modifications. The Hardware itself allows sampling rates up to 80 mega samples per second (MS/s). For our studies, focused on the NB-PLC range, we fixed the sample rate to 1.3 MS/s.

III. SYNCHRONIZATION

In many NB-PLC the zero crossings of the mains voltage are used for symbol synchronization. Furthermore the properties of the power line channel are highly dependent on the instant of time within a mains cycle [2]. Hence it is important to know the exact phase of the mains voltage and to have the capability to send signals synchronized to the mains zero crossings. However when using a SDR approach for system development the latency between generating a signal and the actual point when it reaches the channel is not known and varies every time a device is connected (via USB in our case). This is due to a random sample buffer size when the hardware is initialized. After the initialization the number of samples and thus the latency remains constant. In a first step, after the hardware device has been plugged in, the samples in the loop are determined by using the digital loop-back capability of the transceiver chip. In a second step the number of samples can be increased or decreased to achieve the desired latency. Our experiments showed that a minimum of about 8000 samples (around 6 ms latency at 1.3 MS/s) should remain in the buffer to avoid under runs. This was without carrying out any optimizations regarding operating system or drivers. Fig. 2 depicts a schematic of the data flow and the control block which has been added.

IV. CASE STUDY: FSK-OFDM BRIDGE

The developed framework is used to build up a modem which acts as a bridge (at physical layer) between two entirely different technologies (Fig. 3). It allows a proprietary PLC modem using FSK to communicate with another modem using OFDM. This example illustrates both the flexibility and the feasibility of very rapid prototyping of the software defined approach.
The FSK receiver is implemented using only predefined blocks with the exception of the matched filters synchronization. Fig. 4 shows the top level block diagram of the receiver. The very simple OFDM transmitter was implemented within a single block to simplify matters.

![Figure 4 – FSK-Receiver: top level block](image)

Fig. 5 shows the captured signal of a received FSK data frame and the corresponding, translated, OFDM data frame. There is a delay of 50 ms which is mainly caused by the buffers used in the software implementation.

![Figure 5 – Received FSK data frame and translated OFDM frame](image)

V. CONCLUSION

Software defined modem development offers great advantages over conventional methods using dedicated hardware. Especially for NB-PLC, with its comparatively low data rates, currently available GPPs provide enough power to implement even complex communication systems. By adding the capability to be synchronized to the mains, the platform is very convenient for rapid prototyping as well as for channel investigations. The high flexibility offered by the framework and its usability makes it an ideal toolset not only for modem development but also for research and education in PLC. However there are also some constraints regarding real-time capability which we could also observe in our case study. Particularly in the context of medium access control (MAC) implementations latency cannot be neglected. Recently some interesting results have been published regarding GNU Radio and different MAC implementations [4].

REFERENCES