IMPLEMENTATION OF FSK MODULATED SIGNAL RECEIVER USING SOFTWARE DEFINED RADIO

Jan Pospisil
Doctoral Degree Programme (2), FEEC BUT
E-mail: xpospi90@vutbr.cz

Supervised by: Radek Fujdiak
E-mail: fujdiak@feec.vutbr.cz

Abstract: This paper discusses recent trends in a wireless communication system using Software Defined Radio (SDR). SDR, in this case, the BladeRF 2.0, in combination with a software layer written in the Python language, is taking care of signal demodulation. BladeRF 2.0 is a device that provides a powerful waveform development platform expected by industry professionals. The BladeRF 2.0 is used to design a Frequency Shift Keying (FSK) receiver (in the form of binary data). As a transmitter, we created a custom-designed end-device using CC1101 RF (Radio Frequency) module. The scenario in the current experiment is that the SDR acts as a receiver for the transmitting end-node.

Keywords: BladeRF, SDR, Software Defined Radio, signal demodulation

1 INTRODUCTION

The RF (Radio Frequency) signal is used for wireless communication between electronic devices. It's first necessary to apply a suitable modulation technique to convert a baseband signal into an RF signal to transmit it through an RF environment. Modulation is the process of changing one or more properties of a periodic waveform, called a carrier signal, with a separate signal called a modulation signal, which usually contains information to be transmitted. Within the signal transmission, its failures occur; this is due to, e.g., interference, attenuation, and reflections. Frequency Shift Keying (FSK) is a frequency modulation system in which digital information is transmitted through a discrete change in carrier frequency. Thus, the value of logical zero is one frequency, and the value of logical one is another frequency.

In contrast to relative work articles where the authors of [1] focus on the implementation of Quadrature Phase Shift Keying (QPSK) modulation using Field-Programmable Gate Array (FPGA) and verify the results using simulation. In another paper, the authors apply QPSK modulation [2], capable of partial re-configuration and theoretical verification. The authors of the Frequency modulation (FM) modulator [3] focus on the FPGA implementation.

In the field of Software Defined Radio (SDR), a relatively large number of articles are available, including the ones mentioned above, which deal with the implementation of signal processing in various variants. The topic of SDR technology is relevant, especially in the last 20 years. However, most scientific publications focus on theoretical and simulation results and thus lack real-life implementation. In contrast to the above, our paper focuses on implementing a demodulator verified in real-life conditions. Due to the relatively high price of SDR hardware, the real-life deployment of SDR technology is still slow. SDR devices are mostly used for scientific research. As part of the future vision, SDR equipment should replace complex single-purpose systems. The main advantage is saving funds when implementing a new communication technology into the already deployed infrastructure that uses SDR. Where, in contrast to conventional single-purpose systems, the upgrade can be performed mainly in the software.

2 SOFTWARE DEFINED RADIO (SDR) AND THE END-DEVICE

The SDR device receives a signal from the ambient RF environment at configured parameters in the form of the carrier frequency (f), bandwidth (BW), sampling rate (SR), and gain (G). After setting these parameters, the device starts capturing the digitized analog signal. In this case, it is necessary to perform the implementation of signal processing by the software.

Table 1: BladeRF 2.0 micro A9 basic parameters [4].

	Frequency range	Bandwidth	Sampling rate
BladeRF 2.0	47 MHz – 6 GHz	56 MHz	61.44 Msps

The BladeRF 2.0 micro xA9 device (see Figure 1) has, in addition to the parameters mentioned in Table 1, also a USB 3.0 port (Cypress FX3) and allows 2x2 Multiple-Input Multiple-Output (MIMO) communication. This device's advantage is the presence of an Altera Cyclone FPGA circuit that can be used for direct signal processing [4].

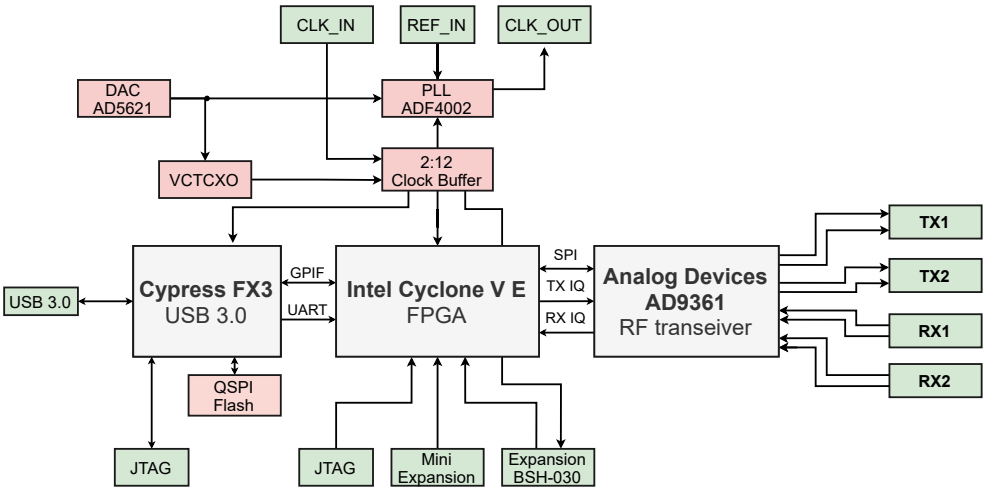

Figure 1: Function block diagram of the BladeRF micro xA9 device [4].

The BladeRF contains the RF 2×2 transceiver AD9361. The Rx signal path from AD9361 passes downconverted signals to the baseband receiver section. The baseband Rx signal path is composed of two programmable analog low-pass filters, a 12-bit ADC (Analog-to-Digital Converter), and four stages of decimating filters (see Figure 2) [5].

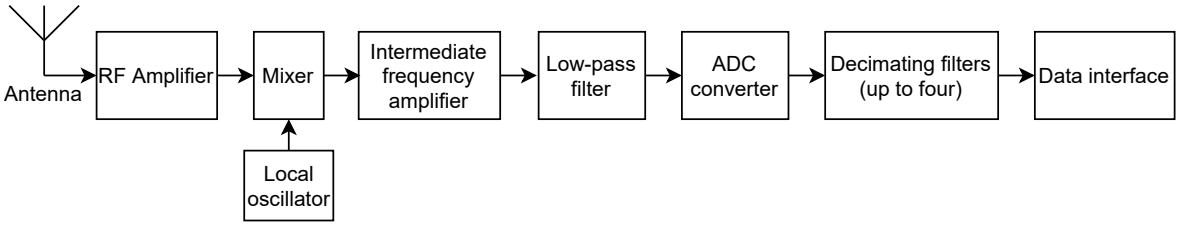

Figure 2: Functional block diagram of one RX channel of AD9361 [5].

The output from the BladeRF is the data flow of the digitized captured radio signal samples. The data flow consists of a sequence of individual samples in the form of two 16-bit values that represent the component I and Q. The output data rate depends directly on the sampling frequency. When the maximum sampling frequency is set (61.44 MHz), the device then generates ≈ 2.0 Gbps. As the data flow increases, so do the demands for the computer's performance for processing this data. As part

of an advanced implementation, the computational load can be reduced by the FPGA engagement for the necessary signal processing that might be performed directly in the BladeRF. The computer then receives the already processed data. The FPGA signal processing implementation is considering to be done in the future. For the real-life transmitting end-device and BladeRF SDR device, see Figure 3.

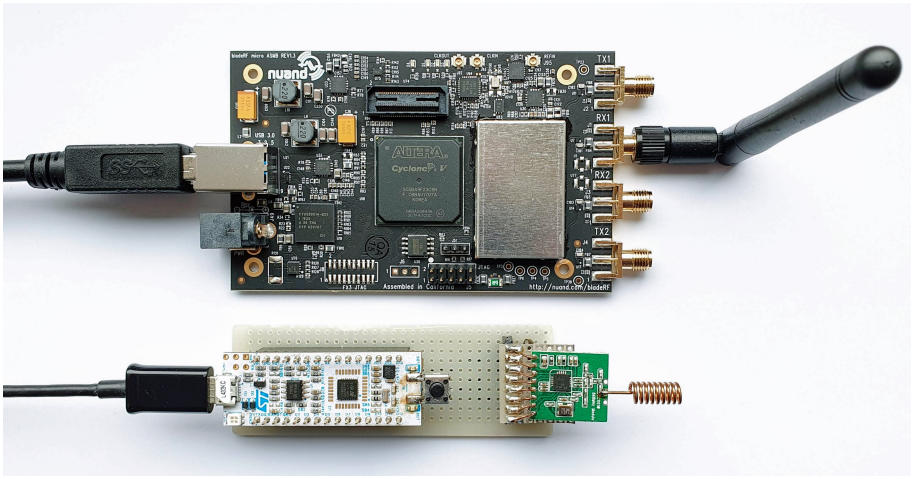

Figure 3: End-device periodically sends data using FSK modulation on the left and the receiving SDR BladeRF on the right.

The end-device consists of an STM32L432KC microcontroller (MCU) powered via micro USB port, and CC1101 RF integrated circuit wired via Serial Peripheral Interface (SPI). Table 2 summarizes the end-device configuration. It should be noted that the device does not use any data encryption. Otherwise, the received data would not be readable without knowing the decryption key. In a real-life scenario, the test end-device might be replaced by any regular end-device that uses the FSK modulation after modifying the receiver's signal parameters, such as carrier frequency, bandwidth, deviation, and data rate. The testing transmission distance was for the best results (low error rate) in the length of centimeters.

Table 2: The end-device configuration.

Radio	Carrier frequency	Bandwidth	Data rate	Frequency deviation	TX power
CC1101	868 MHz	203 kHz	4800 bps	47.607 kHz	12.5 dBm

3 PYTHON SOFTWARE FSK DEMODULATION

We used the Universal Radio Hacker (URH) tool to receive the samples from BladeRF. URH stores the samples in the file with name extension: complex32s. As the extension indicates, the file contains complex samples composed of the two signed 16-bit integers for the I and Q values. The Python script further handles the processing part. For all the signal processing parts, see Figure 4.

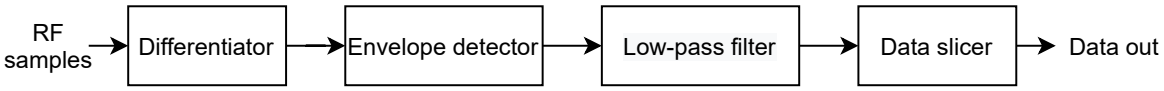

Figure 4: Basic blocks of signal processing for FSK demodulation based on envelope detection.

We smoothed the received signal (see Figure 5) by the moving average filter with a window length of 10 samples. Due to the receiver's high sampling rate (2.5 Msps) and transmitter's low data rate (4.8 kbps), a negligible signal degradation occurred by filtration.

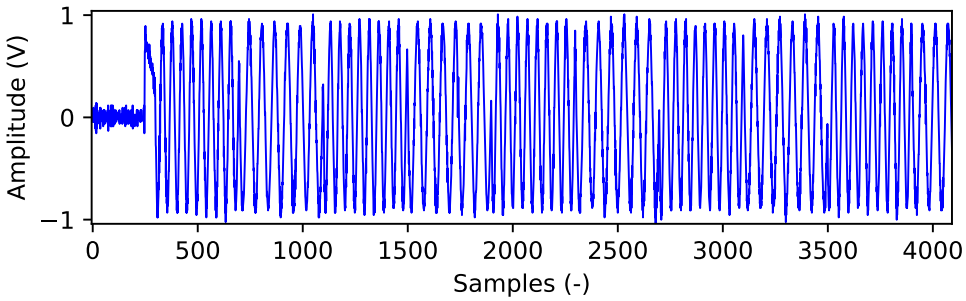

Figure 5: Raw FSK signal received samples – the first 4 000 samples.

In the second step, we applied the first-order discrete differentiation function. We achieved the envelope detection using a Hilbert transformation. Finally, we filtered the signal with a low-pass 100 tap FIR filter with a cutoff frequency of 1 x bitrate to obtain the optimal results (see Figure 6).

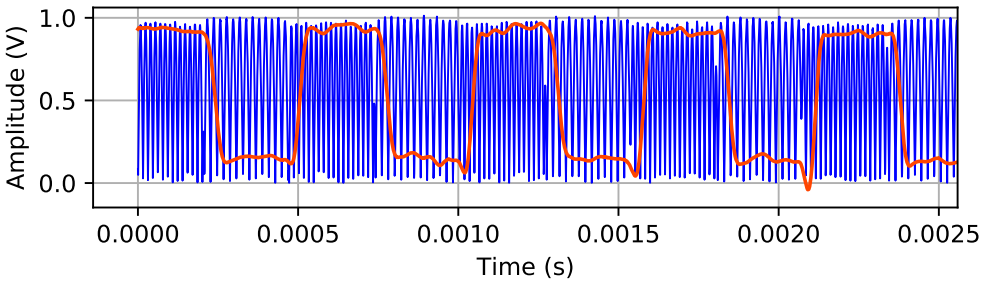

Figure 6: Hilbert transformation and low-pass filtering of the signal (red) compared to the smoothed FSK signal (blue) – the first 25 ms.

The last step in the signal processing part is the slicing and bit recognition. First of all, we took the mean of the entire signal as a decision threshold. Every time that signal crosses the mean value, the slicer creates an edge (see Figure 7 – the red line). This operation results in square signal output. At this moment, we cannot directly convert the square signal into bits due to the sequences of the same values. These need to be split first by the symbol length. The script goes through the edges and, given the symbol lengths (based on adaptive symbol length measurement), breaks the same-value sequences and recognizes each bit.

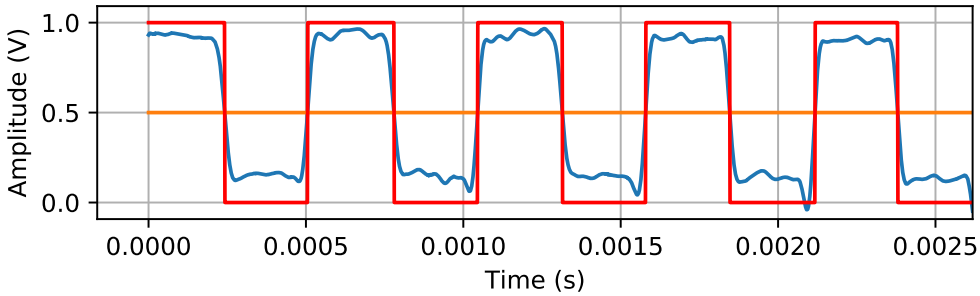

Figure 7: Slicing (red) envelope signal (blue) based on the threshold mean value (orange) and the edge detection – the first 25 ms.

The following Figure 8 shows the whole received signal with the edge detection. The bit recognition result can be seen more in detail in Table 3. As can be observed at the beginning of the frame, a preamble (4 B) is received, then a header containing synchronization data and payload length (5 B). Then a user payload (14 B) is received. Finally, a footer in the form of a Cyclic Redundancy Check (CRC, 2 B) is added.

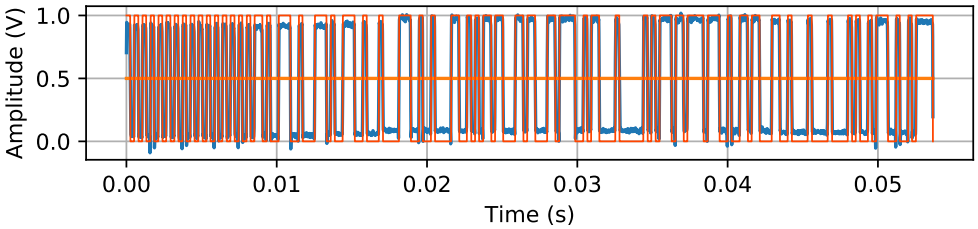

Figure 8: Slicing filtered envelope signal (red) based on the threshold mean value (orange) and the edge detection - complete view.

Table 3: Received data (byte index, hexadecimal and ASCII value).

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Hex	AA	AA	AA	AA	D3	91	D3	91	0E	48	65	6C	6C
Ascii	ª	ª	ª	ª	Ó	Â	Ó	Â	Â	H	e	l	l
Index	13	14	15	16	17	18	19	20	21	22	23	24	
Hex	6F	2C	20	57	6F	72	6C	64	21	0A	5C	97	
Ascii	o	,		W	o	r	l	d	!	<LF>	\	Â	

4 CONCLUSION

Using a software-defined radio, we successfully put a system for receiving the FSK modulated signal into operation. The advantage of this design is the possibility of a feasible way to modify the signal processing part (i.e., modulator/demodulator). Compared to all the hardware used in a traditional communication system, we created this in the software. We successfully transmitted the data from the real-life end-device to the BladeRF SDR and decoded it. Therefore, it is clear that other signal modulation types, even more complex, are possible to be implemented. Our future focus is on developing the signal processing directly on the FPGA within the SDR to reduce the computer's load.

ACKNOWLEDGEMENT

The described research is part of the grant project registered under no. TJ02000332 and funded by the Technology Agency of the Czech Republic.

REFERENCES

[1] QPSK Modulator and Demodulator Using FPGA for SDR MUKESH, Mandadkar; ABHISHEK, Lokhande; BHAMBARE, R. R. QPSK modulator and demodulator using FPGA for SDR. *International. Journal of Engineering Research and Applications*, 2014, 4.4: 394-397.

[2] KUMAR, KA Arun. FPGA implementation of PSK modems using partial re-configuration for SDR and CR applications. In: *2012 Annual IEEE India Conference (INDICON)*. IEEE, 2012. p. 205-209.

[3] HATAI, Indranil; CHAKRABARTI, Indrajit. A new high-performance digital FM modulator and demodulator for software-defined radio and its FPGA implementation. *International Journal of Reconfigurable Computing*, 2011, 2011.

[4] *Nuand LLC*, BladeRF 2.0 [online]. [Accessed 8 March 2021]. Available from: `https://www.nuand.com/bladerf-2-0-micro/`

[5] AD9361 Functional Block Diagram [online]. [Accessed 8 March 2021]. Available from: `https://www.analog.com/en/products/ad9361.html`