# SDR All-channels Receiver for FHSS Sensor Network

Tomáš Jakubík, Jiří Jeníček
Faculty of Mechatronics, Informatics and Interdisciplinary Studies
Technical University of Liberec
Email: tomas.jakubik@tul.cz, jiri.jenicek@tul.cz

***Abstract*—This paper introduces an idea of a spread spectrum wireless sensor network with minimal current consumption and minimal input delay. As a first step towards this goal, this paper describes design and basic testing of needed base station receiver. The receiver can receive multiple Gaussian Frequency Shift Keying (GFSK) devices communicating at once on different Frequency Hopping Spread Spectrum (FHSS) channels. The intended frequency is 868 Short Range Device (SRD) band, but the results could be applied to different symbol rates and frequencies. Simulations were made to verify that the receiver is able to receive multiple packets at once. Live tests on RTL-SDR showed that the receiver has parameters comparable to a commercial GFSK chip.**

***Keywords*—FHSS, Low-power, Sensor Network, SDR**

## I. Introduction

This article intends to bring new ideas into the area of home automation and security. There we can find small cheap devices communicating on SRD sub-GHz band (known also as 868 MHz band). Large portion of those devices are sensors which need to live from few µA. We can see two major directions of development in this area. There is high-speed Wi-Fi which is too power hungry for sensor network. And there are various IOT networks, but none is reliable and fast for a device like smoke detector.

Existing solution for home alarm is typically a single frequency network with sleeping sensors and always listening hub. Sleeping sensor conserves energy and starts transmitting immediately when input changes. This way, it is not easy to introduce spread spectrum to improve on signal reliability and bandwidth usage.

One of the simplest forms of spread spectrum, FHSS, requires synchronization that increases power consumption or takes a long time to initialize. It is possible to design FHSS network with fast synchronization, as we already explored [1], but certain disadvantages remain.

### A. FHSS network with sleeping devices

Different approach is to put another specialized receiver into the hub. A receiver listening on all channels would eliminate the problem with synchronization. Sensor device could sleep as in single frequency solution. On signal change, the sensor could choose randomly one of its preferred channels and start transmitting immediately. Network hub would receive the message, know the important information without any delay, and respond on the same frequency with regular transceiver. The response might be a simple acknowledge or FHSS synchronization data.

The task at hand is to create a receiver for hub that works on many channels concurrently. It would not be profitable to design new radio integrated circuit for the amounts produced in this area of electronics. The remaining solution is to use existing mass produced electronics for Software Defined Radio (SDR) and put necessary development into software. For the development we used an RTL-SDR receiver [2], but the ideal target would be an embedded MCU and demodulator.

## II. Design

The lowest level modulation, binary GFSK with $BT = 0.5$, is given by the available hardware for sensors. Frequency deviation, symbol rate and channel spacing could be set to needs of the receiver.

### A. Receiving Symbols

Simplified binary FSK signal might look as

$$s = A \cdot sin\big(2\pi t \cdot (F_0 \pm F_{dev})\big) \tag{1}$$

where $F_0$ is carrier frequency, and $F_{dev}$ is frequency deviation. The sign can switch each $T_s$ symbol duration. Signal $s$ is received, demodulated, sampled and digitized. This analogue part of the SDR was investigated thoroughly and is still under heavy development [3]. The digital signal first enters a Fast Fourier Transform (FFT) to receive all channels at once.

The FFT at the input of the software part has bins set to frequency $F_0$ of each channel. Signal $s$ won't fit precisely to any bin. Best it can do is

$$s = A \cdot sin(2\pi t F_0 \pm 2\pi t F_{dev}) \tag{2}$$

where $2\pi F_0$ is the bin frequency and $A$ with

$$\pm 2\pi t F_{dev} = \varphi(t) \tag{3}$$

are amplitude and phase, products of the FFT bin.

The phase is linearly changing with time. Practical implementation would have to take care of continuity of the phase between symbols. For Minimum Shift Keying (MSK), which is a special case of FSK, or GFSK this comes naturally. In such case, this can be called Continuous Phase Modulation (CPM).

The first derivative of the phase is

$$\frac{d\varphi(t)}{dt} = \pm 2\pi F_{dev} \tag{4}$$

ISBN 978-80-261-0722-4, 

and that is exactly the bit of information we are interested in. We look at sign of $\frac{d\varphi(t)}{dt}$ and decide on the received symbol. In practice, it has to be replaced by discrete approximation, in the simplest by a difference of two consecutive samples. Each sample $\varphi(t)$ must be in range of $\langle 0, 2\pi)$. If it goes above or below, it wraps around. The same has to be done with the difference. That limits phase difference which can be detected to

$$\varphi\left[\frac{t}{T_c}\right] - \varphi\left[\frac{t-T_c}{T_c}\right] \in \langle -\pi, \pi) \tag{5}$$

where $T_c$ is duration of the samples at the output of FFT. Maximum detectable frequency deviation is

$$2\pi T_c \cdot max(F_{dev}) = \pi \tag{6}$$

$$max(F_{dev}) = \frac{1}{2T_c} \tag{7}$$

Normally, the sampling time could be synchronized with the received signal to minimize the intersymbol interference [4] [5]. Here, several individual devices transmit their symbols randomly shifted in time. It wouldn't help to synchronize sampling to only one of them, and interpolating the signal for each channel separately would be too complicated. For a $30\,\mathrm{ppm}$ quartz crystal, $\frac{T_c}{2}$ shift happens after 8333 symbols or 1042 bytes. That is several times more than longest packet of most similar systems, so the symbol synchronization was simplified to the bare minimum. We selected $T_s = 2T_c$ and sum two neighboring samples.

The condition for maximal received frequency deviation can now be expressed as

$$max(F_{dev}) = \frac{1}{2T_c} = \frac{1}{T_s} \tag{8}$$

which compared to MSK modulation ($F_{dev} = \frac{1}{4T_s}$) gives three-quarter margin for noise and oscillator imperfections.

Channel size, respective size of the FFT frequency bin, is equal to sampling frequency at the output, $F_{ch} = 1/T_c$. The size must be greater than bandwidth used by the communication to fit the signal inside provided channel. It is hard to estimate bandwidth of GFSK, but Carson's rule can give a rough estimate of

$$F_{bw} = 2 \cdot \left(\frac{BT}{T_s} + F_{dev}\right) \tag{9}$$

where $BT$ is Gaussian filter bandwidth bit period product. For GMSK (GFSK and MSK together) with $BT = 0.5$ it would simplify to $F_{bw} = \frac{1.5}{T_s}$. That gives another condition on sampling rate

$$F_{ch} = \frac{1}{T_c} \geq F_{bw} = \frac{1.5}{T_s} \tag{10}$$

$$T_s \geq 1.5T_c \tag{11}$$

which is satisfied by previously selected $T_s = 2T_c$ and a small reserve remains for the bandwidth estimate.

Using higher $F_{dev}$ than GMSK would increase the signal bandwidth, and force us to increase $T_s$ to $T_c$ ratio and decrease the amount of useful data. Since GMSK has been successfully used for decades, there is no need to increase $F_{dev}$.

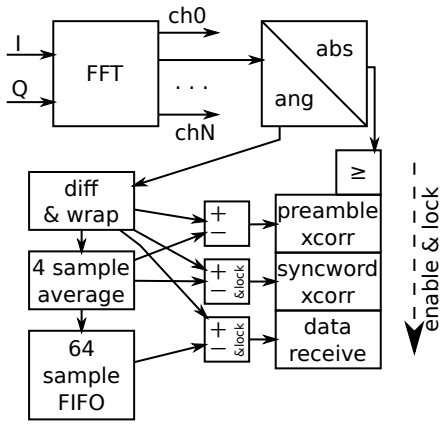


Fig. 1. Basic scheme.

An RTL-SDR dongle has limited IQ sample rate of $2.4\,\mathrm{MSample/s}$ which limits the received bandwidth. To comply with EN 300 220 [6], the size of the FFT has to be 64 as it is the lowest possible power of 2. Some of these channels can be permanently silent to avoid neighboring communication or reserved channels. Channels are $37.5\,\mathrm{kHz}$ apart, symbol rate has to be $1/T_s = 18.75\,\mathrm{kBaud/s}$ and frequency deviation $F_{dev} = 4.6875\,\mathrm{kHz}$. Channels were put from $865\,\mathrm{MHz}$ up, so no LDC/HR reserved channel is overlapped. LDC/HR channels cannot be used by FHSS.

### *B. Processing Packets*

Each packet starts with preamble and syncword. Preamble is an alternating sequence used for synchronization. Syncword is a pseudo-random value, selected beforehand, to know when preamble ends, useful data start and to differentiate between systems with the same modulation. We've chosen running cross-correlation to detect preamble and syncword. Match on the syncword also sets the symbol synchronization. After syncword, it is common to include length of the useful data in the packet. At the packet's end, there is usually a Cyclic Redundancy Check (CRC).

To compensate offset of carrier frequencies, we have used an averaging filter. The averaging filter has length of $4$ samples. Length in multiples of $4T_c$ will get frequency offset from the alternating preamble. The offset is stored in a barrel buffer of 64 values. At time of syncword match the last value is used for symbol decision. The buffer delays valid offset value, because at time of syncword match, the filter has already processed the syncword.

Now the simplified scheme of the receiver can be seen at Fig. 1. Individual blocks can be enabled only at time they are needed to save on computational power. Only the FFT needs to be always on. The conditions enabling next stage are: absolute signal power, preamble cross correlation and syncword cross correlation. Everything must be protected by a hysteresis or failsafe timeout. Absolute signal power is expected to keep high during the whole packet, so there can be a condition resetting everything on packet failure. But both cross correlations should fall before

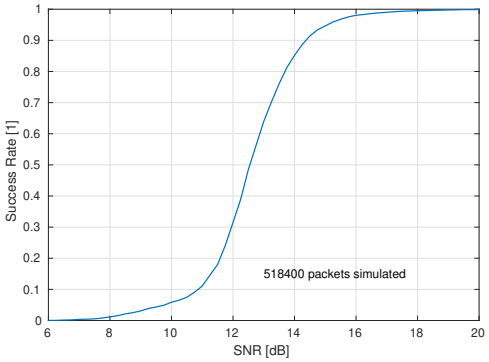


Fig. 2. Simulation, AWGN.

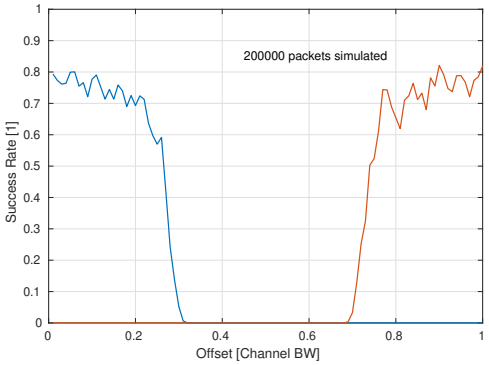


Fig. 3. Simulation, offset dependency.

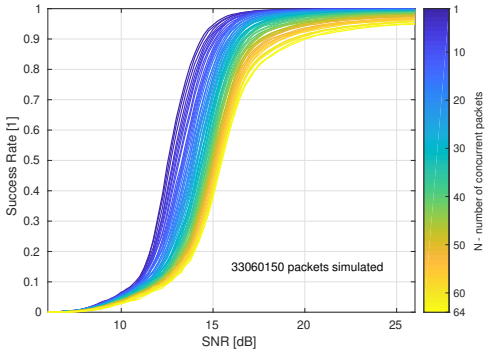


Fig. 4. Simulation, concurrent packets.

the packet ends, so there needs to be a protective timeout resetting the receiver if something fails.

## III. Simulation

The receiver implemented in Matlab Simulink was tested by a simulation. The RTL-SDR module returns preprocessed baseband complex signal in single floating point type. It is equivalent to synthetic $e^{2\pi i f t}$ or to the output of the *comm.GMSKModulator()* function. That simplifies the test and doesn't require almost any changes in the receiver model.

All simulations were done with signal passed through Additive White Gaussian Noise channel. It is the simplest method, but it should be enough to test this receiver. Individual channels of this receiver are relatively thin, and the receiver doesn't presume any multipath effects nor frequency distortions. Frequency specific effects would be interesting with comparison to single frequency system, but that is not specific for this receiver and it has already been studied [7].

The input to all simulations were packets with 20 useful data bytes. That is 31 bytes in total, including all overhead. Perhaps it is more common to simulate bit error rates, but here the success depends on syncword match and frequency offset elimination, so whole packets were simulated. No error correcting code was used and single erroneous bit means the packet is not received successfully. The received data were compared with the data sent, because the CRC used (CRC-16-IBM) offers only a basic protection.

Packets were multiplied by step window passed through Gaussian filter to smooth the power rise and fall. Similar process is done by real transmitters. Additionally, a random time in range between 1 sample and 2 symbols was prepended before the packet. And complement of that time was appended after the packet to have constant length. This was to test the syncword match and the symbol detection.

### A. Receiving packets through AWGN channel

Fig. 2 depicts simulation of different Signal to Noise Ratio (SNR) and the resulting packet success rate. The SNR was corrected for the fact that the FFT filters out $\frac{63}{64}$ of the noise. The SNR level set to the AWGN channel was decreased by approximately $-18\,\mathrm{dB}$.

$$SNR_{offset} = 10 \cdot log_{10}\left(\frac{1}{64}\right) \approx -18\,\mathrm{dB} \qquad (12)$$

The figure shows that the receiver starts receiving between 10 and 16 dB of SNR with 15 dB for 0.95 success rate.

### B. Receiving with frequency offset

Fig. 3 shows how the receiver is influenced by the frequency offset of the transmitter. This simulation was done with random SNR on the AWGN channel in range between 12 and 16 dB.

The left curve on Fig. 3 falls after 0.2. The receiver stops receiving when transmitter shifts for more than $0.2 \cdot 37.5\,\mathrm{kHz} = 7.5\,\mathrm{kHz}$. This is perhaps slightly less than normal GFSK receiver would. It is also less than the predicted value of $\frac{0.75}{T_s} = 14.0625\,\mathrm{kHz}$, but that would be only at ideal conditions without any noise. The right curve is the same curve flipped, when the receiver starts receiving the packet on a wrong channel.

### C. Multiple packets received at once

Fig. 4 shows how the receiver works when multiple packets are running simultaneously. Instead of one packet in previous simulations, $N$ individual packets were generated separately and then summed together. In this simulation the random shift of the packets was increased to between 1 sample and 16 bytes. This should be closer to reality where the chance of packets starting at the exact same time is negligible.

Concurrent packets were randomly spread between available channels by using permutation and picking first $N$ elements. This was to prevent collisions, so there were no two packets at the same frequency at the same time.

Data at Fig. 4 shows decrease of success rate with increasing number of concurrent packets. The receival success rate for one single packet was quickly at 1, but is only slowly rising for 64 packets at once. Even at this case of entirely full radio spectrum, the receival success rate reaches $0.9$ above $20\,\mathrm{dB}$ of SNR. For the real scenario where only a few packets are expected at the same time, the change is negligible.

## IV. Real Test

This time the Matlab Simulink receiver was connected to RTL-SDR hardware. This device is constructed on a base of DVB-T USB dongle. The regular DVB-T receiver is switched to raw mode and sends IQ samples through USB to host PC. It is a cheap alternative to professional tools, essential for students to explore SDR.

The receiver was running in Matlab Simulink in an infinite real-time simulation. Transmitter was a custom electronics with STM32L0 MCU and CC1200 RF transceiver. It sent packets with 20 bytes of useful data. First four bytes were hardware address of the transmitter. Following was one byte with a channel on which the packet was sent. Next two bytes were 16 bit counter incremented with each packet. The counter was used to evaluate missing packets. For each received packet, number of received packets was incremented by 1 and number of missed packets was incremented by counter difference minus 1. The rest of the packet were fixed values to validate the received data.

The transmitter started by $2\,\mathrm{ms}$ receiving, followed by a transmission of the packet. The receiving part was implemented as Listen Before Talk (LBT) (sometimes called Clear Channel Assessment (CCA)) to prevent collisions from influencing the results. This was repeated on different channels. The transmitter was incrementing frequency channels by one on each packet. This simplifies processing of the received data, as all missed packets can be pinned to a given channel. Random channel selection, as in FHSS, would require knowing the random selection at the receiver.

Several tests were made inside an old university building. For the transmitter on the same table approximately $1\,\mathrm{m}$ apart, $99.90\,\%$ of packets were received. Interesting is, that few packets were received as copies on neighboring channel. There is a significant distortion in the signal, because the receiver gain is set to maximum. The reason, as before, is not to attenuate other transmitters by tuning to one.

When the transmitter was one floor up and $30\,\mathrm{m}$ far, the success rate reached even higher $99.98\,\%$.

The test at Fig. 5 was done with transmitter two floors up, about $50\,\mathrm{m}$ far and behind complicated wall structure. $95.03\,\%$ of packets were successfully received. At this distance, the transmission on a single channel wasn't received by the same transceiver CC1200. The Fig. 5 shows that the receiver is bad at the border channels close to the Nyquist frequency.

The overall performance of the receiver would require much more elaborate measurements and laboratory environment. Still, it is seen that the receiver works at a reasonable distance, considering thick walls and floors of an old building. When compared with single channel chip receiver, this SDR receiver is at least comparable if not better.

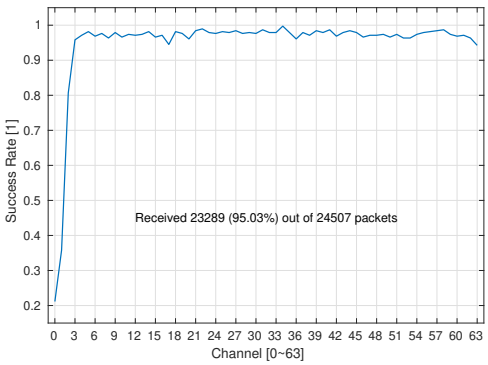


Fig. 5. Real test at range limit of CC1200.

## V. Conclusion and Future Goals

The created receiver is able to function as a hub in the concept of low-power FHSS network. The next step is to shrink it to a convenient size.

In this configuration the 64 sample FFT must be done in $T_c \approx 26\,\mu\mathrm{s}$ which could be done on Cortex-M4 MCU [8]. As the embedded platform we choose embedded kit LPC-Link2 and custom module with R820T, the demodulator used in RTL-SDR. The MCU has fast ADCs and enough computational power.

The receiver implemented on an embedded platform should be an affordable simple solution for companies developing small wireless electronics. Price of this solution will be slightly higher than for ordinary FHSS, but the advantages will likely compensate for the investment.

## Acknowledgment

This work was supported by Student Grant Scheme 2018 of Technical University of Liberec.

## References

[1] T. Jakubík and J. Jeníček, "Asymmetric Low-power FHSS Algorithm," in *Proceedings of the IEEE 13th International Workshop On Electronics, Control, Measurement, Signals and their Application in Mechatronics*, 2017.

[2] M. A. Wickert and M. R. Lovejoy, "Hands-on Software Defined Radio Experiments with the Low-cost RTL-SDR Dongle," in *2015 IEEE Signal Processing and Signal Processing Education Workshop*, 2015.

[3] A. Collins, *All Programmable RF-Sampling Solutions*, Xilinx, 2017.

[4] M. Rice, *Digital Communications: A Discrete-Time Approach*. Pearson Education, Inc., 2009.

[5] M. ur Rehman Awan and P. Koch, "Combined Matched Filter and Arbitrary Interpolator for Symbol Timing Synchronization in SDR Receivers," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, 2010.

[6] *EN 300 220-2*, ETSI Std., Rev. 3.1.1, 2017.

[7] N. H. Motlagh, "Frequency Hopping Spread Spectrum: An Effective Way to Improve Wireless Communication Performance," *Advanced Trends in Wireless Communications*, 2011.

[8] *Digital signal processing for STM32 microcontrollers using CMSIS*, STMicroelectronics, 2016.