Enabling Ambient Backscatter Using Low-cost Software-defined-radio

Maximilian Stiefel
Master Programme Embdedded Systems
Uppsala University
maximilian.stiefel.8233@student.uu.se

Elmar van Rijnswou
Master Programme Embdedded Systems
Uppsala University
elmar.vanrijnswou.9818@student.uu.se

Carlos Prez-Penichet
Communication Research Group
Uppsala University
carlos.penichet@it.uu.se

Ambuj Varshney
Communication Research Group
Uppsala University
ambuj.varshney@it.uu.se

Christian Rohner
Communication Research Group
Uppsala University
christian.rohner@it.uu.se

Thiemo Voigt
Communication Research Group
Uppsala University
thiemo.voigt@it.uu.se

***Abstract*—Backscatter communication is attractive for energy-constrained devices due to its very low power requirements. Ambient backscatter takes this aspect to the limit by leveraging existing radio frequency signals for the purpose of communication without the need for generating energy expensive carrier signal. In this paper we investigate the use of ambient television broadcast signals for communication. As opposed to state-of-the-art restricted to operations under conditions of strong signal strength, we demonstrate a low cost software defined radio as receiver enables operations even in conditions when ambient signals are weak in strength. We build the system using low-cost off-the-shelf microcontroller, and RTLSDR software-defined radio receiver. We also conduct survey of signal strength of TV broadcast in a mid-sized swedish city, and observe that our system can operate in most parts of the city.**

I. Introduction

This demo file is intended to serve as a "starter file" for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

II. Bachground

A. RTL2832U

III. Design

A. Sender

B. Receiver

The receiver was the most work intensive part of the design. C++ has been used to implement the receiver. We provide the code publicly available under https://github.com/s3xm3x/backscatterBASKReceiver. In figure 1 one can see the architecture of the receiver from an abstarct point of view. A highly sophisticated bus system has been developed to exchange data between the different components of the system. The signal flow is from the left to the right. *Librtlsdr* (cf. [2]), which is based on *libusb*, is used to transfer the data from the TV stick into our program. As described under II-A, the RTL2832 mixes down the high frequency to a intermediate

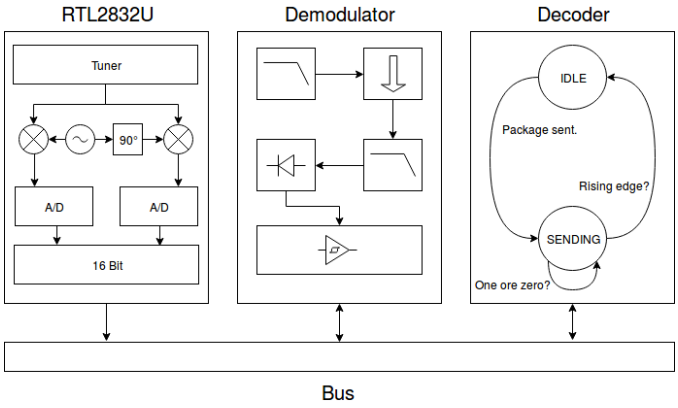

Fig. 1. This figure shows the receiver architecture from an abstract point of view.

frequency (both values are in SW adjustable). Subsequently the data is mixed with sine and cosine, low-pass filtered and finally transfered in the digital world. This data is available as two 8 Bit values. It is a well-known fact, that this technique, which is commonly known as I/Q demodulation, results in a real and an imagenery part. With this two values phase and absolute value of the data can be determined for every sample. So every sample consists out of a real and and imaginary part. And can be seen as:

$$I + j\,Q = abs(I, Q) \cdot e^{ang(I,Q)} \tag{1}$$

So the first block, which is entitled with RTL2832U, provides the interface to the TV dongle. This block is represented as one object. It can set the frequency f_{tuned}, where the receiver is listening. Also it can set the sampling frequency f_{samp} and the analog gain of the tuner. The raw samples are pushed to the bus. Demodulator is the block called, which is responsible to convert the sampled signals into something, which looks like ones and zeroes. Thereforre the real and imgainary values first have to be deinterleaved out of the sample message. Now they are represented as integer values between 0 and 255 (8

bit). So one converts them to floats, whereas 127.5 equals 0. After that the values have to be downsampled. Usually we were sampling with 250 kHz. This is still much more than we need assuming an additive white gausian noise (AWGN) channel with a $S/N = 10\,dB$ (cf. equation 2). So we decided to reduce the sampling frequency to 25 kHz.

$$C = B \cdot log_2(1 + \frac{S}{N}) = 25\,kHz \cdot log_2(11) \approx 86.5\,kbit/s \quad (2)$$

To do this it is important to use an anti-aliasing filter, as the Shannon-Nyquist theorem has to be satisfied:

$$f_{sampling} > 2 \cdot f_{highestfregencyinsignal} \quad (3)$$

A FIR filter naivly implemented with floats has been used to achieve this. 10 kHz is the cut-off frequency of the first filter. All filters have been constructed with the internal filter design tool of GNU radio. After downsampling another filter is used to suppress noise. This filter has a cut-off frequency of 3 kHz. The last demodulation step is to take the signal, rectify it (cf. equation 4) and decide with a software-defined Schmitt trigger wether a sample is 1 or zero.

$$abs(I, Q) = \sqrt{I^2 + Q^2} \quad (4)$$

These samples are broadcasted on the bus. A registered listener receives the processed samples. This registered listener is the decoder, which is represented as an object as well. Thereupon the decoder decides when a frame is sent or when the channel is idle. On the basis of a defined threshold the decoder decides wether a certain sample pattern of ones and zeros is a 1 bit or a zero bit. The baudrate value and the sampling rate value lead to a certain length of one bit. For example a sample rate $f_{sampling}$ of 25 kHz combinded with a baudrate of 1 kbit/s lead to 25 samples/bit.

IV. Evaluation

A. Signal Strength Variation within the City

B. Signal Strength Variation over Time

C. Communication Performance

V. Discussion

Acknowledgment

The authors would like to thank...

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Steve Markgraf et. al., *librtlsdr*, https://github.com/steve-m/librtlsdr. Github: 2012-2017.