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 Pérez-Penichet
Uppsala University
carlos.penichet@it.uu.se

Ambuj Varshney
Uppsala University
ambuj.varshney@it.uu.se

Christian Rohner
Uppsala University
christian.rohner@it.uu.se

Thiemo Voigt
Uppsala University
and SICS Swedish ICT
thiemo@sics.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

April 7, 2017

II. Background

In this section we examine some background aspects that are essential to our work. [1]

A. Software Defined Radio

Traditionally basic radio components like mixers or filters have been realized in hardware. Nowadays as soon as one works in the baseband frequency the computational resources of a personal computer are sufficient to realize these components with signal processing done in software. [2]

B. RTL2832U

The Raeltek RTL2832U is a terrestrial digital video broadcast (DVB-T) demodulator IC, which is at the core of a wide range of USB-based DVB-T receiver dongles. One prominent feature of this chip is that it allows to retrieve the raw I/Q samples via USB. This is originally intended for the chip to work as a simple DAB/FM software receiver. Ham radio enthusiasts have combined their efforts to create a software driver (the *librtlsdr*, cf. [1]) that allows DVB-T dongles based on the RTL2832 to be converted into a low-cost wideband SDR receiver. The cost of traditional SDR's has been in the range of a few hundred to thousands of USD. Albeit much less powerful than the typical SDR, an RTL2832U-based DVB-T receiver can be bought with antenna for less than 10 USD. An RTL-SDR does not only consist of the RTL2832U. The second essential component is the tuner, which allows the user to select the received frequency and outputs that signal at an intermediate frequency (IF). The IF can then be fed into the demodulator for digitalization. In the case of the used hardware this tuner is a *Rafael Micro R820T/2* which offers a tuning range from 42 to 1002 MHz [2]. It subsumes a low-noise amplifier (LNA) and filters. It presents a low noise figure of 3.5 dB and features 65 dBc of image rejection. Figure 3 [3], shows a photograph of the RTL-SDR used in our work.

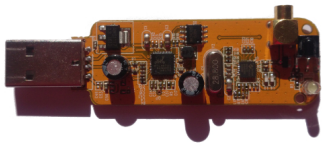

Fig. 1. *RTL-SDR* hardware with the DVB-T I/Q demodulator *Raeltek RTL2832U* (left IC) and the tuner with integrated LNA *Rafael Micro R820T/2* (right IC).

C. Quadrature Demodulation

The received signal can be interpreted as

$$s_{IF}(t) = I(t) \cdot cos(\omega_0 t) + Q(t) \cdot sin(\omega_0 t) \quad (1)$$

This is multiplied with a cosine (and a sine as well) from a local oscillator.

$$s_{IF}(t) \cdot s_L(t) = I(t) \cdot cos(\omega_0 t) \cdot cos(\omega_0 t) + Q(t) \cdot sin(\omega_0 t) \cdot cos(\omega_0 t) \quad (2)$$

[1] **Carlos: Put a simple intro to each section so that they don't look empty before the first subsection.**

[2] **Carlos: We need to expand on this introduction to SDR, assuming we need it and not just assume everyone knows what that is.**

[3] **Carlos: We need a much better photo here.**

With $2\,cos(a)cos(b) = cos(a-b) + cos(a+b)$ and $2\,sin(a)cos(b) = sin(a+b) + sin(a-b)$ follows

$$2 \cdot s_{IF}(t) \cdot s_L(t) = I(t) \cdot \big[1 + cos(2\omega_0 t)\big] + Q(t) \cdot \big[sin(2\omega_0 t) + sin(0)\big]. \quad (3)$$

One can see, that the interesting in-phase part (I) in this case is represented by a DC value after mixing. With a low-pass filter this DC value can be separated from the undesired rest. An analogous procedure is done with the quadrature part (Q) when mixing with a sine. [4]

III. Design

A. Transmitter

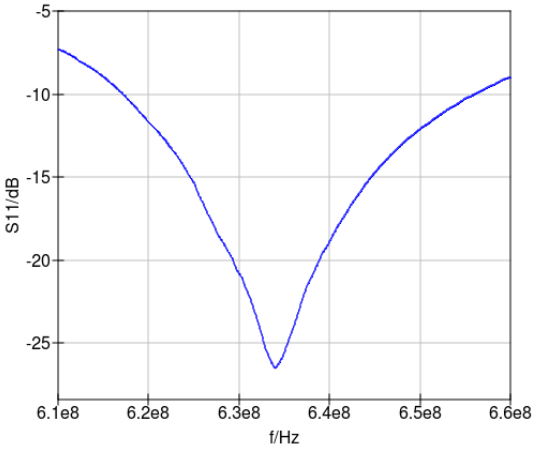

Fig. 2. Input reflection coefficient Γ_{in} also known as S_{11} of the self-made backscattering ground-plane antenna.

A backscattering antenna is the most important part of backscatter transmitter. Our antenna is optimized for usage with the television signal, which has its center frequency at 626 MHz. It is a ground plane antenna, which has been built up on a large unetched PCB plate. This rectangular PCB, with dimensions of 200 mm x 300 mm, is covered with a copper layer on one side and left blank on the other. A hole in the middle of the PCB serves for connecting a wire to a 50 Ω micro strip transmission line. The wire, which acts as the vertical antenna element and has a length of 350 mm (roughly 0.73 λ), is soldered to the transmission line. The transmission line has been calculated for 626 MHz using an FR4 substrate with a thickness of 1.6 mm. According to micro strip line theory the length of the line changes the phase and the width of the line changes the impedance (cf. Chapter 3 in [3]). The design equations given in [3] are implemented in various programs e.g. KiCAD, which has been used for solving the design equations. The given parameters lead to a micro strip line width of 3 mm. An open-stub, attached directly at the juncture where the vertical antenna element is attached to the transmission line, is used for fine tuning of the input impedance. The stub length has been adjusted manually (shortened) while measuring the reflection coefficient with a vector network analyzer (VNA). Open-stub matching is discussed in Chapter 5 in [3]. Furthermore the resonance frequency has been tuned by cutting the vertical antenna element piece by piece while measuring with the VNA until S_{11} had the minimum close to the desired frequency (Figure 2[5]). In the plot one can also see, that the antenna is quite narrowband[6]. For backscattering this is not necessarily a disadvantage, since everything that is within the operating frequency of the antenna is backscattered. With our narrowband antenna the undesired backscattering of out-of-band signals is reduced. With our approach we were able to achieve an input impedance at 634 MHz of $Z_{in} = (52.2 + j4.3)\Omega$. At 626 MHz S_{11} is still approximately -17 dB.

Besides the antenna, the transmitter consists of an *MSP430* MCU, that sends data to the receiver. This transmitter controls an RF switch through an I/O pin. The switch can alternate the impedance connected to the antenna between matched (50 Ω) and open. The software now steers this switch with a two 2 MHz rectangular timer signal to transmit a data 1 and is just leaves it open to transmit a data 0. Therefore the result is a frequency shift[7] of the television signal by 2 MHz for a 1 and no shift for a 0. Hence a classical binary amplitude shift keying is the implemented modulation technique.

B. Receiver

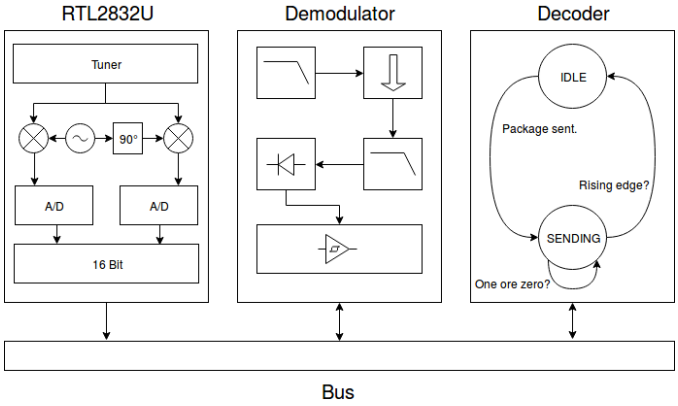

Fig. 3. Receiver architecture from a system point of view. Signal flow is from left to right.

C++ has been used to implement the receiver. We provide the code publicly available under https://github.com/s3xm3x/backscatterBASKReceiver. In figure 3 we show the architecture of the receiver from an abstract 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. [1]), which is based on *libusb*, is used to transfer the data from the

[4] **Carlos: We need to explain this more clearly and possibly more concisely.**

[5] **Carlos: Can we replace all these figures by PDFs?**

[6] **Carlos: We should report something like the 3dB bandwidth rather than saying something non-specific like this**

[7] **Carlos: We need to explain this in the background**

TV stick into our program. As described in section II-B, the RTL2832 mixes down the high frequency to a intermediate frequency (both values are software adjustable). Subsequently the data is mixed with sine and cosine, low-pass filtered and finally transferred to the digital domain. 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 decomposition of the signal into in-phase and quadrature components which are generally represented as real and imaginary parts respectively. With this two values, the phase and absolute value of the data can be determined for every sample. And can be seen as:

$$I + j\,Q = abs(I, Q) \cdot e^{j \times ang(I,Q)} \tag{4}$$

So the first block, which is entitled 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. The demodulator block is responsible for converting the sampled signals into ones and zeroes. Therefore the real and imaginary 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 Gaussian Noise (AWGN) channel with a $S/N = 10\,dB$ (cf. equation 5). So we decided to reduce the sampling frequency to 25 kHz.

$$C = B \cdot log_2(1 + \frac{S}{N}) = 25\,kHz \cdot log_2(11) \approx 86.5\,kbit/s \tag{5}$$

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

$$f_{samp} \geq 2 \cdot f_a \tag{6}$$

f_a is the highest frequency in the signal. A FIR filter naively 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 rectify the signal (cf. equation 7) and decide with a software-defined Schmitt trigger whether a sample is a 1 or a 0.

$$abs(I, Q) = \sqrt{I^2 + Q^2} \tag{7}$$

These samples are broadcast on the bus. A registered listener receives the processed samples. This registered listener is the decoder, which is represented as an object as well. After the decoder received the samples it decides when a frame is sent or when the channel is idle. On the basis of a defined threshold the decoder determines whether a certain sample pattern of ones and zeros is a 1 bit or a 0 bit. The baudrate and the sampling rate lead to a certain bit duration. For example a sample rate $f_{sampling}$ of 25 kHz combined with a baudrate of 1 kbit/s leads to 25 samples/bit. The decoder also includes a function to correlate the received bitstream with an expected pattern. Hence it is able to determine the bit error ratio (BER). Furthermore the code we have written so far subsumes a simulator to simulate ideal data coming out of the demodulator to test the algorithms behind the decoder. Also a simulator for the RTL2832U has been written to replay recorded samples, which facilitates debugging as well. Another handy utility, which has been written in Octave, is a tool (an oscilloscope) to look at the data, which is written into files by the demodulator. An example of a recorded data stream with 25 kS/s can be seen in figure **??**.

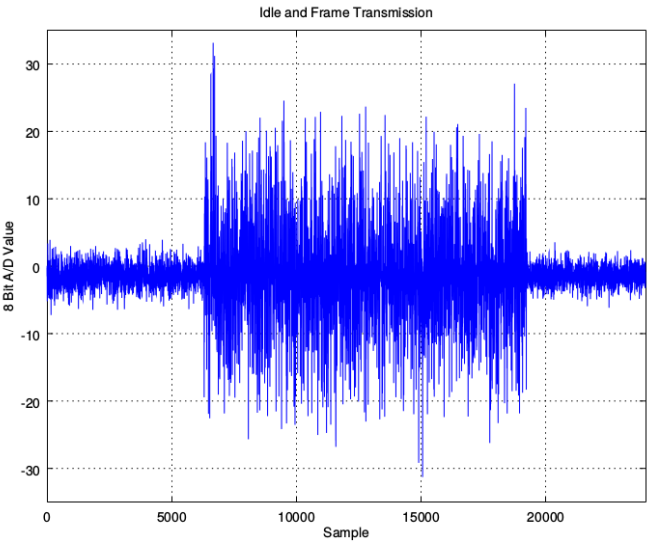

Fig. 4. Transmission of 101010.. with 1 kbit/s and idle state. Data from the Octave oscilloscope. Amplitude is quite low with an average of 12 % of the maximum, which is 127.5.

IV. Evaluation

The hard results, we collected so far are presented in this section. Of course the *RTL-SDR* is the main tool combined with intelligent signal processing in *Octave*.

A. Signal Strength Variation within the City

To observe the signal strength variation within the city the *RTL-SDR* has been used as well. *Octave* scripts have been written for this purpose. These scripts are available online as well under https://github.com/s3xm3x/RTLSDRspecAn. Basically the scripts perform a frequency sweep through the desired band. All samples are written into files. With another script these samples, which end up in one directory, can be transfered from the the time domain into the frequency domain. This is done using an FFT. The basic problem is, that the *RTL-SDR* does not have the huge sampling frequency, which is required to look at the huge band (of e.g. 20 MHz). So the Nyquist-Shannon sampling theorem, which is explained also in II-Bi, strikes again. The trick is, that we are no interested in real-time data. This allows us to only look at one part of the

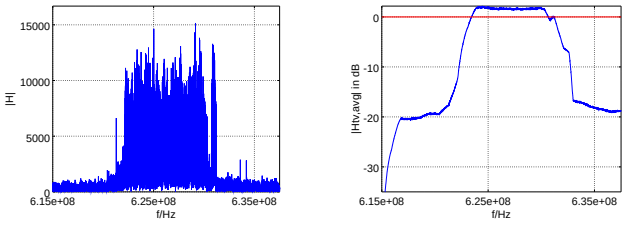

Fig. 5. Visualization of the signal processing, which is done. Right: Raw TV signal with $f_{\text{center}} = 626$ Mhz. Left: Smoothed signal, normalized to maximum average measured in decibel with average shown as red line.

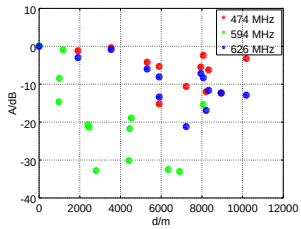

Fig. 6. Signal strength fading against distance relative to the spot where the maximum signal strength has been captured (TV towers). 474 MHz and 626 MHz belong to a tower in Uppsala, Vedyxa and 594 MHz belongs to a tower in Uppsala, Rickomberga. Distances have been calculated with the Haversine formula.

interesting band and sweep through everything within roughly two minutes. The sampling frequency used is 1.8 MHz. This results in a sampled band of 900 kHz. Hence the resulting 900 kHz bands are glued together (shifted in frequency). The negative frequncies of the FFT are not used at all. In this case they are not playing any role of improtance. The gain is tuned to 40.2 dB. To get the signal strength we simply take the average over a 10 MHz band arround the TV signal. This approach can be justified by the fact, that DVB-T is specified up to 10 MHz bandwidth. And the local TV signal provider is to its own statements using DVB-T 2. The signal $|H|$ is calculated as follows.

$$|H| = \left| \sum_{n=0}^{N} FFT\big(\Re\{s_{\text{band,n}}(t)\}\big) \right| \tag{8}$$

In the equation above N is the number of smaller bands which are accumulated into a big picture. The FFT is only carried out over the real values, which are received from the *RTL-SDR*. $s_{\text{band,n}}(t)$ is the signal in the time domain of one band. There are as many samples taken in one band as needed for a FFT size of x. In our case we chose x to be 2048. No window is used when glueing the data together. One can see, the effects of this a little bit, when taking a closer look at the plotted data. Rising and falling edges at the beginning of each band are visible. To get $|H_{tv,avg}|$ one has to normalize everything and convert calculate the power. Before doing that a simple smoothing algorithm (FIR) is applied on $|H|$.

$$|H_{tv,avg}| = 20 \cdot log(|H|) - 20 \cdot log(|H_0|) \tag{9}$$

$|H_0|$ is a defined value for normalization. This actually can be everything e.g. a signal with a defined power from a signal generator. In our case it is just the maximum value measured on a certain point.

B. *Signal Strength Variation over Time*

C. *Communication Performance*

V. Discussion

Acknowledgment

The authors would like to thank...

References

[1] "steve-m/librtlsdr." [Online]. Available: https://github.com/steve-m/librtlsdr

[2] Rafael Micro, "High Performance Low Power Advanced Digital TV Silicon Tuner. R820t, Rev 1.2." [Online]. Available: http://www.rafaelmicro.com/product/view/21

[3] D. M. Pozar, *Microwave Engineering*, 4th ed. Hoboken, NJ: Wiley, Nov. 2011.