Radio Data Signal Transmission With Software Defined Radios

Lab Report

Daniel May
Simon Schmitt
Technische Universtität Darmstadt
daniel_nicolas.may@stud.tu-darmstadt.de
simon_johannes.schmitt@stud.tu-darmstadt.de

ABSTRACT

This lab report presents the creation and transmission of a Radio Data System signal. The signal is generated on the basis of given sample data according to the corresponding standard document. In the end, we could successfully receive and decode the signal using a customary FM radio.

1 INTRODUCTION

Radio signals have been used for more than a century now, to broadcast information wirelessly. The best known one is the FM radio, as it can be received by everyone with a common radio receiver. While the FM radio was originally developed to broadcast music and news reports from different radio stations, more enhanced receivers allowed to process additional information (e.g traffic updates in car radios) next to the existing ones. Therefore, the *Radio Data System (RDS)* was invented. In this lab report we will give a detailed insight into RDS and how it can be implemented. We have divided the report into a preparation section, to discuss the data that will be transmitted via RDS and several sections for the individual tasks we have solved.

2 IMPLEMENTATION

In the following we describe how RDS signals can be generated using MATLAB.

2.0.1 Preparation: Baseband Coding. Before the actual implementation we have to define and format the information we are about to generate.

Setup.

- The programme service name should be "#SEEMOO#"
- We want to transmit a Mono signal without Artificial Head
- The signal is not compressed
- We use a static PTY which is set to Pop Music
- We want to transmit Music
- We do not carry traffic announcements
- Our radio station is in Germany
- We cover a local area
- We use 1 as the program reference number

To be able to transfer this information it has to be converted into a binary stream using baseband coding. The standard documentation[1] of RDS defines that the stream of bits is split into multiple groups. Each of them consists of four blocks, which have predefined meanings. Since, each group can only transmit two symbols we need four groups to send the programme service name. The overall encoding is shown in the table1.

Group 1 Block 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 1 Block 2	0 0 0 0	1 0 0 1	0 1 0 0	1 0 0 0
Group 1 Block 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 1 Block 4	0 0 1 0	0 0 1 1	0 1 0 1	0 0 1 1
Group 2 Block 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 2 Block 2	0 0 0 0	1 0 0 1	0 1 0 0	1 0 0 1
Group 2 Block 3	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 2 Block 4	0 1 0 0	0 1 0 1	0 1 0 0	0 1 0 1
Group 3 Block 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 3 Block 2	0 0 0 0	1 0 0 1	0 1 0 0	1 0 1 0
Group 3 Block 3	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 3 Block 4	0 1 0 0	1 1 0 1	0 1 0 0	1 1 1 1
Group 4 Block 1	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 4 Block 2	0 0 0 0	1 0 0 1	0 1 0 0	1 0 1 1
Group 4 Block 3	1 1 0 1	0 0 0 0	0 0 0 0	0 0 0 1
Group 4 Block 4	0 1 0 0	1 1 1 1	0 0 1 0	0 0 1 1

2.0.2 Task1: Bitstream Generation. Within this first task we initialize the message we want to send. Therefore, we import the bit stream we previously prepared and append the corresponding checksum and offset. After decoding the message in the first test we get the following output, which shows the hexadecimal, binary and decoded representation.

Decoder Output.

```
d001_0de 094b_3e2 d001_372 4f23_212
1101_0000_0000_0001__00_1101_1110
0000_1001_0100_1011__11_1110_0010
1101_0000_0000_0001__11_0111_0010
0100_1111_0010_0011__10_0001_0010
00B (BASIC) - PI:D001 - PTY:
Pop Music (country:DE/LY/YU/__/__, area:Local, program:1)
==>#SEEMOO#<== -  -  -Music-STEREO - AF:p
```

2.0.3 Task2: Differential Encoding. Since we are about to use frequency modulation (FM) in one of the next steps it is recommended to perform a differential encoding. The problem with modulation is that a receiver of a signal can not determine the logic assigned to a phase shift, as it might be introduced by the wireless channel. Differential encoding provides unambiguous signal reception, because the encoding of the data depends not only on the current signal state (bit), but also on the previous one. ($y_i = y_{i-1} \bigoplus x$).

2.0.4 Task3: Manchester Encoding. *Manchester Encoding*, also called **Phase Encoding (PE)**, is a line code. The information is represented by the edge of a signal that corresponds to the clock

SEEMOO PhySec (Lab RDS) 2017/18, February 2018, Darmstadt, Germany

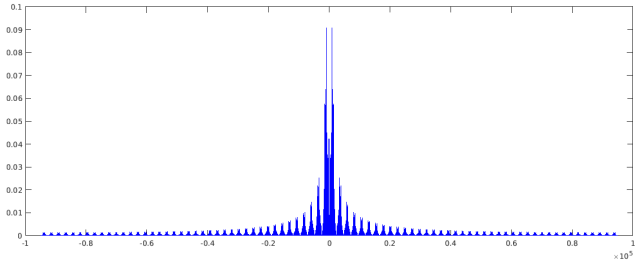

Figure 1

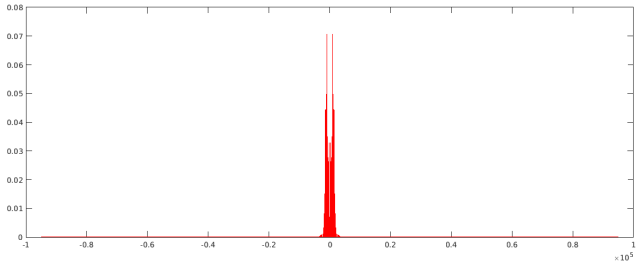

Figure 2

(generated in the first two lines of Task3). A binary zero becomes a low to high transition, a binary one a high to low transition. This type of representation is named after G.E. Thomas. After the differential encoded bitstream has been mapped to Manchester Encoded symbols, we get a RDS signal. However, as shown in Figure1 this signal also contains frequencies we actually dont need. Therefore, we apply a lowpass filter and normalize the resulting signal. This normalized signal is shown in Figure2.

```
bit_clk = [ones(80,1);-ones(80,1)];
bit_clk = repmat(bit_clk,length( ...
    bitstream_differentially_encoded),1);
rds = kron(bitstream_differentially_encoded, ...
    [ones(80,1);-ones(80,1)]) ...
    + kron(~bitstream_differentially_encoded, ...
    [-ones(80,1);ones(80,1)]);
f6 = fir1(300,2400/fs*2);
rds_filt = filter(f6,1,rds);
rds_filt = rds_filt/max(rds_filt);
```

2.0.5 Task4: Generate Baseband Signal. We will now generate the baseband signal. This signal represents the actual data we want to transmit. First of all, we have to change the frequency of the current signal to 57 kHz, since this is the expected frequency for a RDS signal. Therefore, we multiply the RDS signal with a 57 kHz sine wave. To generate a complete baseband signal we also have to insert a pilot and an audio signal. The pilot signal is generated with a 19 kHz sine wave and it is used to upconvert the RDS signal. The audio signal is generated as a mono signal, which means that we sum up the left and the right audio channel at a frequency of 0.3 to 15 kHz. The spectrum of the final baseband signal is shown in Figure3.

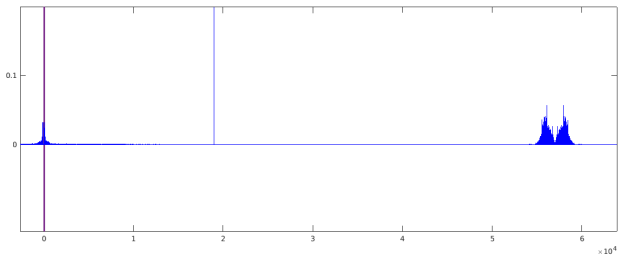

Figure 3

2.0.6 Task5: Frequency Modulation. In software defined radios, the generated baseband signal we generated in Task4 is first frequency modulated before it is upconvert to the carrier frequency. This is done by combining the carrier frequency and the baseband signal and adapting the frequency according to the amplitude of the baseband signal. In order to do the upconversion, a quadrature modulator is used. It seperately upconverts the inphase component

$$I(t) = \cos(2\pi f_\Delta \int_0^t m(\tau)\,\mathrm{d}\tau)$$

by multiplication by a $\cos(2\pi f_c t)$ signal and the quadrature component

$$Q(t) = \sin(2\pi f_\Delta \int_0^t m(\tau)\,\mathrm{d}\tau)$$

by multiplication by a $-\sin(2\pi f_c t)$ signal. For readability reasons, we will use the symbols $\omega = 2\pi f_c$ and $\theta_m(t) = \int_0^t m(\tau)\,\mathrm{d}\tau$ in the following. First, we calculate the upconversion of the inphase component

$$\begin{aligned}&\cos(2\pi f_\Delta \theta_m(t)) \cdot \cos(\omega t)\\ &= \frac{1}{2}(\cos(2\pi f_\Delta \theta_m(t) - \omega t) + \cos(2\pi f_\Delta \theta_m(t) + \omega t))\end{aligned}$$

and second the upconversion of the quadrature component

$$\begin{aligned}&\sin(2\pi f_\Delta \theta_m(t)) \cdot -\sin(\omega t)\\ &= \sin(2\pi f_\Delta \theta_m(t)) \cdot \sin(-\omega t)\\ &= \frac{1}{2}(\cos(2\pi f_\Delta \theta_m(t) + \omega t) - \cos(2\pi f_\Delta \theta_m(t) - \omega t))\end{aligned}$$

These two signals are summed up to generate the complete signal that will still be amplified by the transmit amplifier and then sent. Thus we get

$$\begin{aligned}&\frac{1}{2}(\cos(2\pi f_\Delta \theta_m(t) - \omega t) + \cos(2\pi f_\Delta \theta_m(t) + \omega t))\\ &\quad + \frac{1}{2}(\cos(2\pi f_\Delta \theta_m(t) + \omega t) - \cos(2\pi f_\Delta \theta_m(t) - \omega t))\\ &= \frac{1}{2}(\cos(2\pi f_\Delta \theta_m(t) - \omega t) + \cos(2\pi f_\Delta \theta_m(t) + \omega t)\\ &\quad + \cos(2\pi f_\Delta \theta_m(t) + \omega t) - \cos(2\pi f_\Delta \theta_m(t) - \omega t))\\ &= \frac{1}{2}(2\cos(2\pi f_\Delta \theta_m(t) - \omega t))\\ &= \cos(2\pi f_\Delta \theta_m(t) - \omega t)\end{aligned}$$

Replacing the symbols from before again, we get the signal

$$s'(t) = \cos(2\pi f_c t + 2\pi f_\Delta \int_0^t m(\tau)\,\mathrm{d}\tau)$$

Figure 4

This can be combined into complex numbers by using the inphase component as real and the quadrature component as imaginary part. Since we procress the signal digitally, we need it to be discrete which is done achieved by sampling it and quantizing its amplitude. Applying this leads to the following frequency modulated baseband signal, where the integral is replaced by a sum:

$$fm_{BB} = exp(j2\pi f_{\Delta} T_s \sum_{k=0}^{n} m(kT_s))$$

2.0.7 Task6: Transmission by USRP. Finally, we setup a FM radio, connect to a USRP device and send the frequency modulated RDS signal. As shown in Figure4 the FM radio receives and decodes the signal and displays the programme service name *#SEEMOO#* on the screen.

3 CONCLUSION AND TAKE-AWAY

In this lab we generated and sent our own RDS Signal. We started from encoding the raw information we wanted to transmit in text form to a binary form and further prepared this data for transmission by applying a differential and a Manchester encoding. Then we generated the baseband signal, frequency modulated and upconverted it and finally transmitted it using an USRP. The customary FM radio we used to receive the signal showed the correct programme service name.
In the preparation for and during our work on this lab we learned how to build a RDS signal from the scratch using the standard documentation. Furthermore we learned which mechanisms are used to ensure that the receiver can decode the signal with as little loss of information as possible and how to implement them using MATLAB. More precisely, we learned about the usage of the encoding schemes, the combination of different signals and frequency modulation.

4 FUTURE WORK

Using this prototype implementation we could think of several questions one could investigate. First, is it possible to send our own RDS signals to a driving car. Second, can existing RDS signals be manipulated in any way, e.g. by jamming. Third, is there any possibility to overwrite the existing RDS signals sent by real radio stations to accept our own data instead of the official one. Moreover, the implementation could be extended by an additional module that also allows to receive RDS signals using MATLAB. This module could be used to analyze RDS traffic in different locations.

REFERENCES

[1] 1999. *The new RDS IEC 62106:1999 standard.* Technical Report. RDS Forum.