TECHNISCHE UNIVERSITÄT DARMSTADT

RADIO DATA SIGNAL TRANSMISSION WITH SOFTWARE DEFINED RADIOS

MATTHIAS SCHULZ

Read this manual and do the preparation exercises in the text before coming to the lab!

Secure Mobile Networking Lab
Department of Computer Science

Built: 2014/11/01 22:56

CONTENTS

ACRONYMS

RDS	Radio Data System
VHF	Very High Frequency
FM	Frequency Modulation
USRP	Universal Software Radio Peripheral
SDR	Software-Defined Radio
PI	Programme Identification
PTY	Programme Type Code
TP	Traffic Programme
TA	Traffic Announcement
MS	Music Speech
DI	Decoder Identification
DAC	Digital-to-Analog Converter
DUC	Digital Upconverter

1 MOTIVATION

Did you ever asked yourself how the stereo in your car is able to know the name of the radio station you tuned in? Or how a navigation system—without an internet connection—gets its traffic information? No matter if you did, this lab shows you how the system works by giving you a real hands-on exercise including the implementation of a digital wireless communication system with Software-Defined Radios (SDRs). In the following, we introduce you to the Radio Data System (RDS) for VHF/FM sound broadcasting in the frequency range of 87.5 to 108.0 MHz [2]. The system is used to transmit digital information along with an audio signal. In this lab, we first give you an introduction into the system. Then you are supposed to generate your own RDS signal in MATLAB and transmit it to a radio receiver using the Universal Software Radio Peripheral (USRP).

Figure 1: The result of this lab on a display of an off-the-shelf kitchen radio.

2 BASEBAND CODING

In this section, we focus on the data that is transmitted using RDS. The data itself is represented in form of a bit stream, that is grouped into groups

of 104 bits. Each of these groups contains four blocks that consist of a **16-bit information word** and a **10-bit checksum word** plus an offset word (Figure 2).

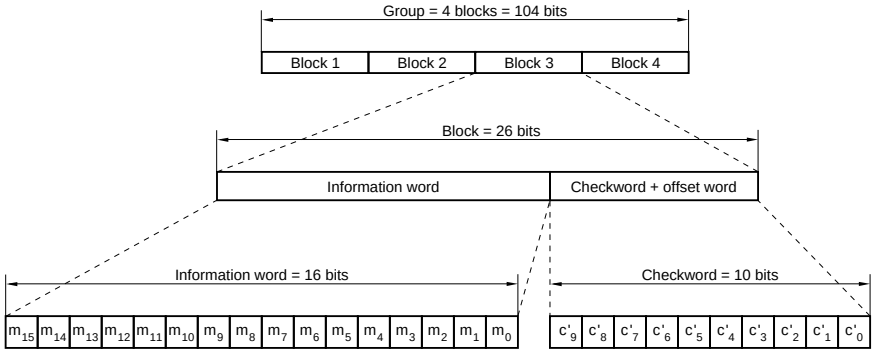

Figure 2: Structure of a baseband group [2]

The checkword allows to perform error detection and correction in the receiver. The offset word, which is added to the checkword, is used for synchronization of groups and blocks. Further information can be found in [2]. For this lab, we provide you a MATLAB script to generate the checkword as well as a function to synchronize groups and blocks and to interpret the received data groups.

Different group types exist that describe the information that is contained in the blocks. Their distinction is explained below. There are group types that contain, for example, the name of a radio station (type 0A, 0B), radio text (type 2A, 2B), clock-time and date (type 4A) or traffic information (type 8A). The first block in each group contains the 16-bit Programme Identification (PI) code that consists of a 4-bit country code, a 4-bit coverage area code and an 8-bit programme reference number. In version B groups, the PI code is repeated in block three. In version A groups, the third block can be used to transmit other information.

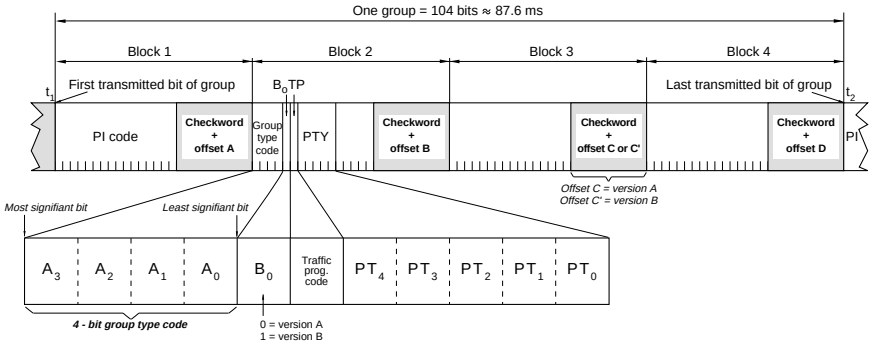

Figure 3: Structure valid for every group [2]

Figure 3 illustrates the strucute of the first two blocks. It is valid for every group. The first four bits in block two represent the aforementioned group type code and the fifth the group version. The sixth bit is the Traffic Programme (TP) code. The following five bits describe the Programme Type Code (PTY) (e.g. News (1), Sport (4) or Pop Music (10)).

In the first part of the lab exercise you will generate groups to set the programme service name, that will be displayed on an RDS enabled radio receiver. Therefore you will create type 0B groups, that are illustrated in Figure 4. As aforementioned, blocks one and three are equal in version B groups. The 16 bits in block 4 represent two 8-bit ASCII characters. The position of those two characters on an eight character display is defined by the last two bits in block two (C_1 and C_0).

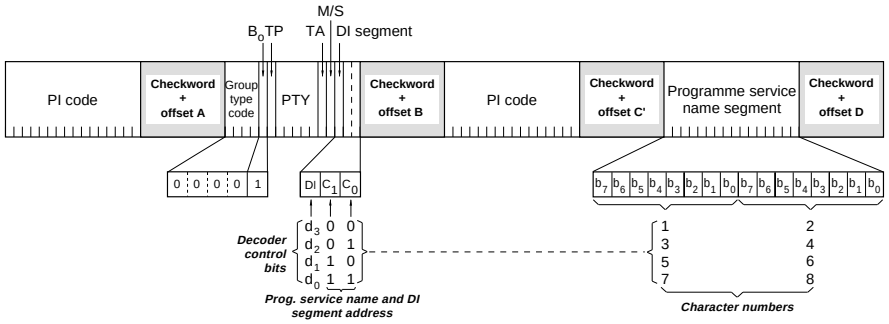

Figure 4: Structure of type 0B group [2]

In a type 0B group there are three additional bits, that we did not explain yet. The Traffic Announcement (TA) code is connected to the TP bit and allows to highlight traffic information. The Music Speech (MS) switch code allows to define if music or speech is transmitted. The Decoder Identification (DI) control code is a bit more complex, as the meaning changes according to C_1 and C_0. You can find the possible options in [2].

To prepare for the lab, please fill out Table 1. Each row corresponds to the information word of one block. The aim is to transmit the following information. You should lookup the correct bits in [2].

Fill out Table 1 before comming to the lab.

- The programme service name should be "#SEEMOO#" (without "")
- 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
- You can choose your own programme reference number

During the lab a MATLAB script calculates the checkword for you and adds the offset.

3 BASEBAND MODULATION

After generating the bit stream including checkwords and offsets, the bits have to be modulated to symbols. At the receiver, these symbols have to be mapped back to the corresponding bits. This step has to cope with the difficulty that the wireless channel introduces phase shifts, that might lead to inverted bits after recovery. RDS solves this problem by applying differential encoding to the bit stream, according to Table 2 *(The initial previous output is zero)*. Here, the value of a data bit is represented in the change between two encoded bits. A receiver can extract the data bits by differentially decoding the signal, according to Table 3.

3.1 *Manchester Encoding*

Besides differential encoding, Manchester encoding is applied to the bits before transmission. In Manchester encoding, every bit is represented as a transition from -1 to +1 or from +1 to -1. The former transition represents a binary zero, the latter a binary one. The bit rate is 19 kHz/16 = 1187.5 Bits/s, where 19 kHz is the pilot frequency (see Section 4). Assuming a baseband

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Group 1 Block 1																
Group 1 Block 2																
Group 1 Block 3																
Group 1 Block 4																
Group 2 Block 1																
Group 2 Block 2																
Group 2 Block 3																
Group 2 Block 4																
Group 3 Block 1																
Group 3 Block 2																
Group 3 Block 3																
Group 3 Block 4																
Group 4 Block 1																
Group 4 Block 2																
Group 4 Block 3																
Group 4 Block 4																

Table 1: Preparation for lab

Differential encoding		
Previous **output** (at time t_{i-1})	New input (at time t_i)	New output (at time t_i)
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: Differential encoding

sampling frequency of $f_s = 19\,\text{kHz}$, each modulated bit is represented by 16 samples, where half of them is set to -1 and half of them is set to +1, according to the Manchester encoding scheme (compare Figure 5). After Manchester encoding, a lowpass filter is applied to bandlimit the RDS signal.

After modulating the baseband RDS signal, it is upconverted to 57 kHz, combined with the 19 kHz pilot signal and optional audio signals. Then it is frequency modulated and again upconverted to radio frequency, where it will be transmitted. The following section describes the baseband signal generation.

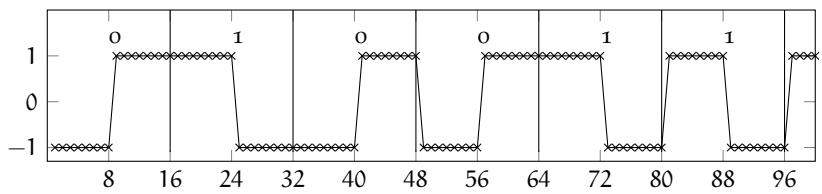

Figure 5: Manchester encoded bit stream 010011 at 1187.5 Bits/s and 19 kHz sampling rate.

Differential decoding		
Previous input (at time t_{i-1})	New input (at time t_i)	New output (at time t_i)
0	0	0
0	1	1
1	0	1
1	1	0

Table 3: Differential decoding

4 BASEBAND SIGNAL

Figure 6 illustrates the ideal spectrum in the baseband before frequency modulation. From 300 Hz to 15 kHz, we find the sum of the left and the right audio channels. A mono receiver only plays back this signal. At 19 kHz we find the pilot tone that is used to upconvert the substraction of the left and the right audio channels to 38 kHz and the RDS signal to 57 kHz.

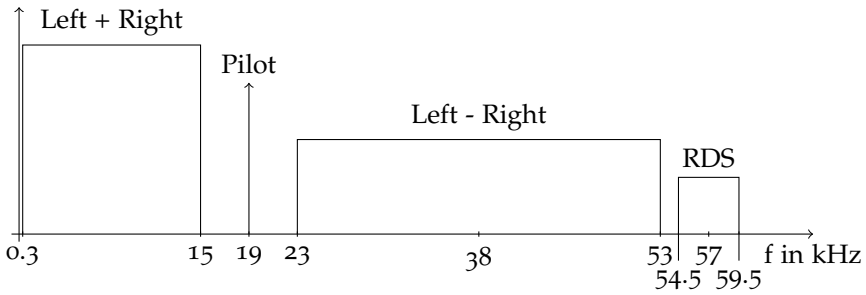

Figure 6: Spectrum of the baseband signal before frequency modulation

Upconverting a band-limited baseband signal to a target frequency is simply performed by multiplication of the baseband signal by a cosine function at the desired target frequency.

Combining the left-plus-right, the upconverted left-minus-right, the pilot and the upconverted RDS signals leads to the complete baseband signal as illustrated in Figure 6. Before upconverting the baseband signal to the carrier frequency in the range of 87.5 to 108.0 MHz, the baseband signal is frequency modulated. This step is described in the following section.

5 FREQUENCY MODULATION

Using Frequency Modulation (FM), a message signal $m(t) \in [-1, 1]$ modulates the carrier frequency f_c of a radio signal:

$$s(t) = A_c \cos(2\pi f_c t + \theta_m(t)) \tag{1}$$

As frequency is the derivative of phase, we get the instantanious frequency by deriving the argument of the cosine function used to generate the radio signal:

$$2\pi f = \frac{d}{dt}(2\pi f_c t + \theta_m(t)) = 2\pi f_c + \frac{d}{dt}\theta_m(t) \tag{2}$$

$2\pi f_c$ is a constant term describing the carrier frequency, whereas $\frac{d}{dt}\theta_m(t)$ is the time dependent instantanious frequency that we intend to change according to the amplitude of our message signal $m(t)$:

$$\frac{d}{dt}\theta_m(t) = m(t) \tag{3}$$

To get the phase $\theta_m(t)$, we have to integrate:

$$\theta_m(t) = \int_0^t m(\tau)\,d\tau \tag{4}$$

This term can be used in Equation 1 as phase. Additionally, we append a sensitivity factor, which is set to $2\pi f_\Delta$:

$$s(t) = A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t m(\tau)\,d\tau\right) \tag{5}$$

f_Δ is the frequency deviation and describes the maximum frequency deviation from the carrier f_c. During our experiments, we choose f_Δ to be 75 kHz.

6 TRANSMISSION BY SOFTWARE-DEFINED RADIOS

Equation 5 describes the radio signal that is supposed to be injected into the transmitter's antenna. When using software defined radios, we do not directly generate the radio signal at the carrier frequency. Instead, we create it at low frequencies and then use a quadrature modulator in the SDR to upconvert this so-called baseband signal to the center frequency f_c of our transmitter.

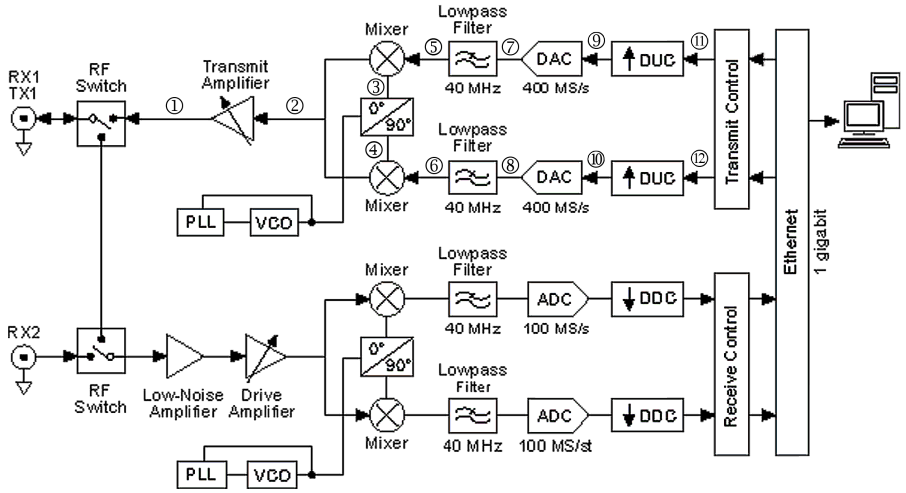

Figure 7: NI USRP-2920 block diagram [1]

In Figure 7, you see the block diagram of the NI USRP-2920, which is the SDR you use during the lab session to transmit the radio signals. At position ① you find the $s(t)$ signal described in Equation 5. To simplify the equations, we define the amplification of the transmit amplifier to be A_c, so that the signal at ② is:

$$s'(t) = \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t m(\tau)\,d\tau\right) \tag{6}$$

To generate $s'(t)$, a quadrature modulator is used. It upconverts the inphase component ⑤ by multiplication by a $\cos(2\pi f_c t)$ signal ③ and it upconverts the quadrature component ⑥ by multiplication by a $-\sin(2\pi f_c t)$ signal ④. The upconverted signals are summed up to get ②. Show—**and document your derivation in your lab report**—that the quadrature modulator generates $s'(t)$, if the inphase component is:

Document your mathematical derivation of the quadrature modulator in your lab report.

$$I(t) = \cos\left(2\pi f_\Delta \int_0^t m(\tau)\,d\tau\right) \tag{7}$$

and the quadrature component is:

$$Q(t) = \sin\left(2\pi f_\Delta \int_0^t m(\tau)\,d\tau\right) \tag{8}$$

Inphase and quadrature components can be combined into complex numbers. Hereby, the inphase component becomes the real and the quadrature component becomes the imaginary part. The resulting complex baseband signal then represents the frequencies around the carrier frequency f_c. Hence, it consists of frequency components at negative and positive frequencies. Equation 9 describes the frequency modulated complex baseband signal:

$$fm_{BB}(t) = I(t) + jQ(t) = \exp\left(j2\pi f_\Delta \int_0^t m(\tau)\,d\tau\right) \tag{9}$$

7 DIGITAL SIGNAL PROCESSING

When processing signals at a computer, they have to be discrete in time and amplitude. Therefore, signals are sampled at equidistant points in time, where the sampling frequency $f_s = 1/T_s$. Additionally, the signals amplitude is quantized into discrete steps. In Figure 7, the digital signals ⑨ and ⑩ are passed through Digital-to-Analog Converters (DACs) with a fixed sampling frequency of $f_s = 400\,\text{MS/s}$. The resulting analog signals ⑦ and ⑧ contain steps at each sampling point. Lowpass-filtering these signals limits their bandwidth and removes the steps resulting in smooth signals.

Sampling the signals given in Equations 7, 8 and 9 replaces the integrals over continuous signals by sums over sampled signals, where n is the discrete time variable:

$$I(n) = \cos\left(2\pi f_\Delta T_s \sum_{k=0}^{n} m(kT_s)\right) \tag{10}$$

$$Q(n) = \sin\left(2\pi f_\Delta T_s \sum_{k=0}^{n} m(kT_s)\right) \tag{11}$$

$$fm_{BB}(n) = \exp\left(j2\pi f_\Delta T_s \sum_{k=0}^{n} m(kT_s)\right) \tag{12}$$

As mentioned above, the DACs have a fixed sampling rate of 400 MS/s. Assuming that each complex sample is representable by 32 bits, feeding the DACs requires bitrates of $400 \cdot 32\,\text{Mbit/s} = 12.8\,\text{Gbit/s}$. As it is not possible to reach these high bitrates over 1 Gbit Ethernet, the USRPs contain Digital Upconverters (DUCs). They increase the sampling rates of the incoming signals ⑪ and ⑫ by a given factor, which highly reduces the amount of data that has to be transmitted from computer to USRP.

Reaching the end of this introduction, you are now ready to get your hands dirty in our real hands-on lab exercise. Have fun!

8 EXERCISE

In the preceding sections you learned about how to generate and modulate an RDS signal, as well as how an SDR works. In this section, you will apply your knowledge to generate the baseband signal in MATLAB and transmit it with a USRP so that it will be received by an off-the-shelf FM radio receiver. In the following, there are some **bold** task descriptions regarding your lab report. You should write your report after the lab session, but you might need to save some results for the report.

8.1 *What you should bring to the lab*

You should bring your results of the preparation tasks. To increase the fun factor of the lab, you can (but do not need to) bring your own music in an audio file (MP3, M4A, WAV, ...).

8.2 *Development methodology*

We provide you with a MATLAB script, that guides you through the programming process. It contains comments to help you programming. In each task, you will program one component of the transmitter (see blocks in Figure 8). To directly test, if your code is correct, we already implemented each receive block, but commented out the code. So whenever you are done with the development of a transmit block, uncomment the receive block code to test if you successfully fulfilled the task, before proceeding to the next task. The receiver blocks require the signals to be stored using specific variable names. You can find these variable names in the transmitter sections (e.g. `bitstream = ...;`). In this lab manual, we might give hints on which MATLAB functions might be useful to fulfill the task at hand. If you do not know a function, use MATLAB's `help` or `doc` commands.

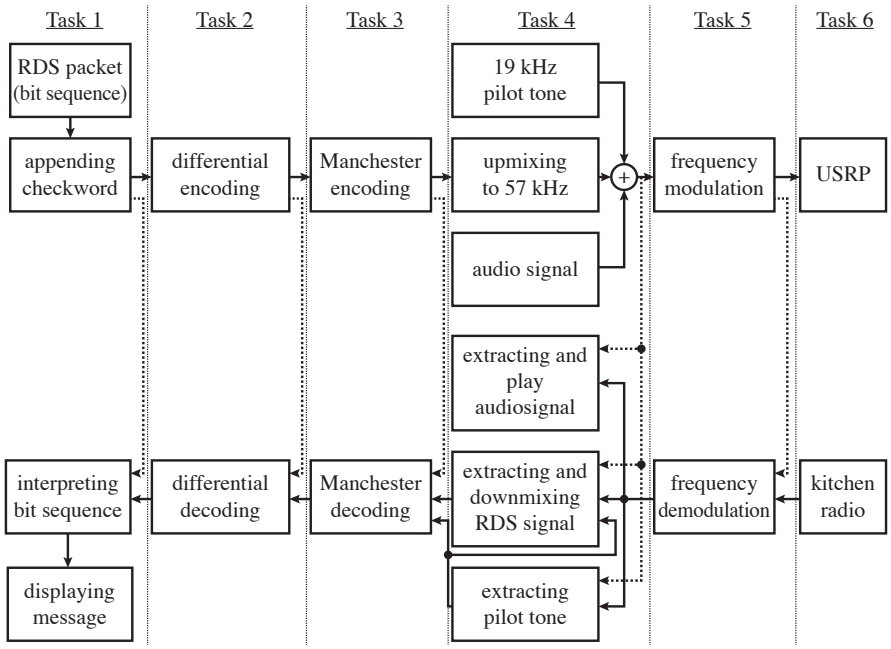

Figure 8: Block diagram of the RDS transmitter and receiver. Each transmit block developed in the lab has a corresponding block in the receiver to test it's functionality.

8.3 *Task 1: Bitstream Generation*

In Section 2, we asked you to fill out Table 1 which contains the bit stream of the RDS messages for the lab. Now open the *labexercise_student_short.m* file and perform the following:

- Copy the content of your Table 1 into the `groups` variable in MATLAB.
- Uncommon the code under *Task 1 Test: Information Extraction* and see if you can extract the correct station name and other bits that you set. The output always contains group-wise information. In the first row are the bits in hexadecimal representation, followed by four lines of binary representation; one line per block. The next two lines contain the decoded message. The station name is between ==> and <==.
- **Add a representative decoder output to your lab report.**

Decoder output for your lab report

8.4 *Task 2: Differential Encoding*

According to Section 3 and Table 2, the created bit stream has to be differentially encoded.

- Take your bitstream from *Task 1* and differentially encode it according to Table 2. The initial "previous output" is zero.
- Unmomment the code under *Task 2 Test: Differential Decoding* and check that the RDS groups can still be extracted.

8.5 *Task 3: Manchester Encoding*

After differential encoding, the bits are mapped to transitions between -1 and +1 with a bit rate of 1187.5 Hz and 190 kHz sampling rate. To accelerate the task, we already give you the implementation. Now do the following:

- Uncomment the code under *Task 3: Manchester Encoding*.
- Uncomment the code under *Task 3 Test: Manchester Decoding and Synchronization*. The RDS decode implementation of function uses is not optimal. Therefore, the internally used block size for processing an input signal might not lead to a correct synchronization and bit extraction. If you encounter problems, change the last number in the function call. Again, check if you can still decode your RDS messages.
- Plot the spectrum of the generated signal and the filtered signal (e. g. `plotfft(signal, samplingrate, 'b')`). **Save the two graphs for your lab report and discuss their differences in the report**.

Some graphs for your lab report.

- **In your lab report, explain what the given code does**.

Explain the code.

8.6 *Task 4: Generate Baseband Signal*

Now you will create the complete baseband signal, including a 19 kHz pilot tone, the RDS signal around 57 kHz, as well as a mono audio signal. We already provide the code for you to insert the audio signal.

- Generate a 19 kHz sinusoidal pilot tone with a sampling rate of 190 kHz.
- Generate a 57 kHz sinusoidal tone with a sampling rate of 190 kHz, that will be used to upconvert the RDS signal.
- Use the 57 kHz tone to upconvert the RDS signal.
- Uncomment the code to add the audio signal.
- Combine the 19 kHz pilot, the RDS signal and the given audio signal. Before adding the RDS signal, reduce its amplitude by a factor of two. If the RDS signal is too strong, some receivers cannot decode it correctly. If your RDS signal does not decode correctly, try to change its amplitude.

Another plot for your lab report.

- **Plot the spectrum and save it for your lab report. Indicate the different signals in the spectrum, according to Figure 6**.
- Uncomment and run the code under *Task 4 Test: Disassemble Baseband Signal*.

8.7 *Task 5: Frequency Modulation*

Now that you have a correct baseband signal, we use Equation 12 to frequency modulate the signal and get out a complex baseband signal. As the bandwidth of a frequency modulated signal increases beyond two times the frequency deviation, we first increase the sampling rate to 400 kHz.

- Change the sampling frequency of the baseband signal to `fs_fm` = 400 kHz (`resample`).

- Normalize the output to an amplitude of one.
- Implement Equation 12 and set the frequency deviation f_Δ to 75 kHz (use `cumsum` for $\sum$).
- Uncomment the frequency demodulation code under *Task 5 Test: FM Demodulation* and check, if your RDS messages are decodable.

8.8 *Task 6: Transmission by USRP*

For the transmission with the USRP we already created the code that you should uncomment. In the code we instantiate an `SDRuTransmitter` object passing the center frequency around which our baseband signal is transmitted, a gain setting, that should be kept at one, as well as an interpolation factor, that configures the DUC to increase the sampling rate of the baseband signal by an integer factor before transmission. Then we use a loop to send 1000 sample chunks of the frequency modulated signal to the USRP for transmission. At the end, we destroy the `SDRuTransmitter` object again.

- Uncomment the code for the transmission.
- Use an FM radio to tune in the selected center frequency to receive your radio transmission and look if the stations name shows up on the display.
- **Document your success with a picture of the radio's display in your lab report**.

Take a picture for your lab report.

8.9 *Task 7: Performing advanced SDR experiments*

If you liked the lab exercise and you are interested in working with SDRs, feel free to visit our website `http://seemoo.tu-darmstadt.de` and sign in for one of our labs or projects. Or contact us if you are interested in writing a master thesis using SDRs. We might be able to offer you a personalized thesis topic.

REFERENCES

[1] NI USRP-2920 Block Diagram. URL `http://zone.ni.com/reference/en-XX/help/373380A-01/usrphelp/2920_block_diagram/`. last visited: November 1, 2014.

[2] RDS Forum. The new RDS IEC 62106:1999 standard, December 1999.