



AALBORG UNIVERSITY
STUDENT REPORT

Software Defined Radio

Getting started with the USRP N200 and LabVIEW

Mini project
within Modulation Theory and Techniques

Thomas Kølback Jespersen
tkje13@student.aau.dk

Hand in: **5. december 2015**
Project period: **5th semester**
Supervisor: **Flemming Bjerger Frederiksen**
Number of pages: **37**

Aalborg University
Electronics and IT

Copyright © Aalborg University 2015

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

Preface	iii
1 Software-Defined Radios	1
1.1 Complex signal representation	2
1.2 I and Q samples	3
1.3 SDR hardware	4
1.4 Modulation scheme	5
2 The USRP	6
2.1 USRP hardware	6
2.2 Connecting the USRP to a PC	9
2.2.1 PC Network settings	9
2.3 Control options	10
3 LabVIEW and USRP	13
3.1 What to install	13
3.2 NI-USRP Configuration Utility	14
3.2.1 Configuring multiple USRP modules	15
3.2.2 Firmware update	16
3.3 Running your first example	17
3.3.1 niUSRP EX Spectral Monitoring example	17
3.4 USRP blocks within LabVIEW	18
3.4.1 Receiver example	20
3.4.2 Transmitter example	21
4 AM Modulation	23
4.1 AM modulation scheme	23
4.2 USRP as AM modulator	25
4.2.1 Single tone modulation	25
4.2.2 Audio-file modulation	26
4.3 USRP as AM demodulator	28
5 Conclusion	31
Bibliography	32
A AM modulation in LabVIEW	34
A.1 Single tone modulation with USRP	35
A.2 Audio file modulation with USRP	36
A.3 Audio and spectrum demodulator with USRP	37

Preface

The main purpose of this mini project and document is to describe the fast approaching software defined radio technology and how this can be used to easily design and test specific modulation schemes.

It is not a requirement that the reader has a former knowledge within software defined radios, SDR in short, but it is required that the reader has some basic knowledge within radio communication, transmission theory and modulation techniques.

The basics behind software defined radios and how a modulated signal is represented as complex samples, is described initially.

The guide and description found within this document is based on the USRP N200 SDR module. Two modules will be used throughout the guide to allow for both transmission and reception, as each module only allows a one way channel. The setup and configuration of the modules is described whereafter each module is connected to a Windows PC equipped with National Instruments LabVIEW software.

Finally a practical test with the implementation of an AM-modulator and demodulator on the two modules is described. By the use of LabVIEW a single tone sine wave or audio-file is modulated and transmitted thru one of the USRP modules. The other module acts as a demodulator and displays the demodulated frequency spectrum and time domain samples, together with the playback of the audio thru the PC audiocard.

In general this document serves as a getting started guide with the USRP N200 software defined radio modules and LabVIEW. Please notice that this document is not based on the LabVIEW Communications System Design Suite, even though this software is most commonly used with the USRP modules.

Chapter 1

Software-Defined Radios

Software-defined radios is an emerging technology which has gained a lot of publicity in the past few years. The reason is the uniqueness of being able to model and control complicated analog RF tasks, such as modulation and demodulation, simply by using software and programming environments. Beforehand these tasks required extensive knowledge within the analog world and expensive tools to build and test designs, whereof multiple design-spins were necessary before everything was right.

Software developers have always had the advantage of easily being able to test the software for bugs or flaws before deployment. Software-Defined Radio describes the technique of using a universal hardware frontend to receive and transmit RF signals with waveforms defined in software applications. The software-defined radio therefore enables developers to do the same thing with RF designs, as the developers are enabled to change carrier frequencies, modulation schemes and data to be transmitted on the fly. A benefit that is useful in both the research and development stage as well as in the deployed stage. As GNU Radio founder Eric Blossom says: "Software radio is the technique of getting code as close to the antenna as possible. It turns radio hardware problems into software problems." [4].

Software-defined radios, SDR in short, are widely used today. An SDR can be programmed to any RF specific task such as GSM or LTE networking, FM transmission, GPS tracking, WiFi or Bluetooth communication etc.[1] In the recent satellite by GOMSpace [5] they have installed an SDR to be able to update the satellite on the fly for mission specific RF tasks. When a task finishes, they can reprogram the satellite to another task, even though the communication or tracking is using another frequency or band.

1.1 Complex signal representation

From signal processing and communication theory we know that a physical signal can be represented as the real part of its' corresponding complex signal:

$$s(t) = \text{Re}\left(a(t)e^{j\theta(t)}\right) = a(t)\cos(\theta(t)) \quad (1.1)$$

Where $a(t)$ is the time varying amplitude of the signal and $\theta(t)$ is the time varying phase of the signal. A constant phase change over time corresponds to a sine-wave signal with a given frequency:

$$\theta(t) = 2\pi f_c t + \phi(t) \quad (1.2)$$

Where f_c is the constant frequency, such as the carrier frequency of an RF signal, and $\phi(t)$ is any time varying changes done to the phase of this constant frequency signal.

In general an RF signal corresponds to a modulated carrier wave, where the modulation consists of time dependent changes of the amplitude and time dependent changes phase of the carrier frequency. The actual message signal to be transmitted and the desired modulation scheme defines these time dependent amplitude and phase changes. Let $a(t)$ denote the amplitude changes and $\phi(t)$ the phase changes, both due to the message signal, and let f_c denote the carrier frequency, then the combined RF signal can be written in its' complex form:

$$s(t) = a(t)\cos(2\pi f_c t + \phi(t)) = \text{Re}\left(a(t)e^{j\phi(t)}e^{j2\pi f_c t}\right) \quad (1.3)$$

Notice how the carrier frequency is just a complex multiplication. This allows us to remove the carrier frequency part and extract the complex envelope also known as the baseband signal or complex baseband.

$$\tilde{s}(t) = a(t)e^{j\phi(t)} \quad (1.4)$$

The baseband signal is generated from the message signal and therefore contains a modulated version of the message signal which can be extracted. The actual relationship between the modulation amplitude, $a(t)$, and modulation phase, $\phi(t)$, depends on the chosen modulation scheme.

When the baseband signal is combined with the carrier frequency the actual RF signal to be transmitted is denoted as:

$$s(t) = \text{Re}\left(\tilde{s}(t)e^{j2\pi f_c t}\right) \quad (1.5)$$

So to sum up, a baseband signal contains all required information about a message signal and the chosen modulation scheme. The baseband signal can be combined with any desired carrier frequency to generate signal to be transmitted, or the baseband signal can be extracted from a received signal to demodulate the message content. [17]

1.2 I and Q samples

Software-defined radios represent the baseband signal as two real numbers, by splitting the signal into its' real and imaginary part. These are called the I and Q samples. The I samples being the real part and the Q samples being the imaginary, also known as the quadrature samples. [13]

$$\tilde{s}(t) = a(t)e^{j\phi(t)} = I(t) + jQ(t) \quad (1.6)$$

Using the angle-addition identity of cosine with the equation for the RF signal, equation 1.3, the RF signal can be rewritten into:

$$s(t) = a(t)\cos(2\pi f_c t + \phi(t)) \quad (1.7)$$

$$= a(t)\cos(2\pi f_c t)\cos(\phi(t)) - a(t)\sin(2\pi f_c t)\sin(\phi(t)) \quad (1.8)$$

From which we can separate the carrier frequency part and the baseband signal part, given by its' I and Q samples:

$$s(t) = I(t)\cos(2\pi f_c t) - Q(t)\sin(2\pi f_c t) \quad (1.9)$$

From this we see that the I and Q samples correspond to the amplitude of the cosine and sine part of the phase changes to the signal, and that the I and Q samples are therefore 90 degrees apart. This agrees with the complex equation of the baseband signal, equation 1.6, as the real and imaginary part is 90 degrees apart.

$$I(t) = a(t)\cos(\phi(t)) = \text{Re}(a(t)e^{j\phi(t)}) \quad (1.10)$$

$$Q(t) = a(t)\sin(\phi(t)) = \text{Im}(a(t)e^{j\phi(t)}) \quad (1.11)$$

Because of this 90 degree separation of the I and Q samples, they are also referred to as the "in-phase", $I(t)$, and "quadrature", $Q(t)$, components of the signal.

The actual rate of change in the message signal is usually many factors less than the rate of change in the combined RF signal, due to the high carrier frequency. In other words the bandwidth of the message signal is usually in an size of megahertz while the carrier frequency is in region of sub-gigahertz or gigahertz. The I and Q samples allow us to sample or generate samples at a lot lower rate as the high-frequency part, due to the carrier, has been removed. This exact feature is used within software-defined radios, to allow a small bandwidth message signal to be generated and modulated in software and transmitted over the air with a high-frequency carrier.

1.3 SDR hardware

To allow a software application to change the characteristics of an RF communication channel, the hardware has to be designed for dynamic adjustments and the processing of the I and Q samples. In principle, a universal hardware serves as an interface between the baseband and the RF. The waveform of the baseband signal that has to be transmitted, is fully generated through software, as well as a received baseband signal is fully processed and demodulated within software algorithms. In SDR, the processing power required for signal processing, including modulation, encryption, channelling, quadrature encoding etc., is outsourced to a universal host, eg. a PC.

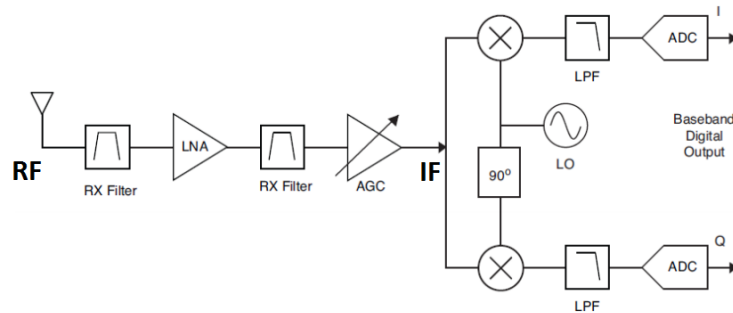


Figure 1.1: Software-defined radio RF frontend for reception.

Depending on whether the SDR is used for transmission or reception, the analog part, also known as the RF frontend, takes care of combining the baseband signal with a carrier frequency or extracting the baseband signal from the carrier frequency. An RF frontend for reception is shown in figure 1.1. The frontend includes two tunable RF filters, to filter away any unnecessary nearby frequencies, and a Low Noise Amplifier (LNA) and an Automatic Gain Control (AGC). The right part of the RF frontend takes care of the conversion from the IF signal, taken from the output of the AGC, to the complex baseband signal, given by its' I and Q signals. This is done by using a quadrature demodulation technique with a Local Oscillator, tuned to the desired carrier frequency. This quadrature demodulation implements the extraction of the I and Q samples from the complex equation, equation 1.9. These I and Q signals are then sampled by the two ADC's in the SDR hardware. See page 12 of reference [17] for more information.

Due to the quadrature conversion and splitting happening in hardware, the ADC's within the SDR can sample the I and Q samples at a lot lower rate than if the actual RF signal with the carrier frequency had to be sampled. For transmission this is the same, as the baseband signal to be transmitted can be generated and sent to the SDR hardware at a lower sample rate than the actual carrier frequency. Thereafter the RF frontend takes care of quadrature modulating the baseband signal with the desired carrier frequency.

1.4 Modulation scheme

The description of the different kind of modulation techniques and how to implement them on a software-defined radio is not a part of this document. As described, a specific modulation scheme is used to generate the baseband signal, given by its' I and Q samples. When defining the I and Q samples for a specific message signal and modulation scheme, you have to consider the following two equations, equation 1.12 and 1.13:

$$s(t) = a(t)\cos(2\pi f_c t + \phi(t)) = \text{Re}\left(a(t)e^{j\phi(t)}e^{j2\pi f_c t}\right) \quad (1.12)$$

$$\tilde{s}(t) = a(t)e^{j\phi(t)} = I(t) + jQ(t) \quad (1.13)$$

Though to give an idea on how to implement a simple modulation scheme, AM modulation and demodulation is described and implemented in chapter 4.

Chapter 2

The USRP

The USRP, which is short for Universal Software Radio Peripheral, made by Ettus [3] is one out of many software-defined radio modules available on the market targeted for development and research. The module comes in different varieties, formfactors and with different specifications such as bandwidth, throughput and frequency range. The USRP Network series allows one or multiple SDR units to be connected to a host machine thru a Gigabit Ethernet connection, enabling an easy setup for both small setups and scaled setups. Within this document two USRP N200-KIT equipped with the XCVR2450 daughterboard will be used for testing.



Figure 2.1: USRP N200-KIT [14]

2.1 USRP hardware

The main hardware inside a USRP unit mainly consists of an FPGA with DSP functionality. Furthermore the hardware includes multiple high speed ADCs for sampling a received signal and high speed DACs for generating a signal for transmission. The FPGA configures a local oscillator to the desired carrier frequency and processes the samples to and from the DACs and ADCs from the incoming or outgoing data on the Ethernet link. Some of the USRP modules includes a high-precision clock to be used as the clock reference. In general the USRP main hardware supports any carrier frequency between DC and 6 GHz usually, only limited by the actual RF frontend.

On top of the main hardware with the FPGA, a daughterboard is installed which includes the hardware for the analog RF frontend. Different daughterboards can be purchased each with specific frequency range, bandwidth and precision. Especially the frequency range can be important as the daughterboards usually only support a narrow frequency range. The daughterboards are separated into four families, UBX, WBX, SBX or CBX, who are supported by the USRP family. Some USRP units supports multiple daughterboard families at the same time, such as the N200-KIT which supports all four families

Identifier	Frequency range	Area of application
Transceiver		
RFX900	750 to 1050 MHz	GSM (Low Band)
RFX1200	1150 to 1450 MHz	GPS
RFX1800	1,5 to 2,1 GHz	DECT, GSM (High Band)
RFX2400	2,3 to 2,9 GHz	WLAN, Bluetooth
XCVR 2450	2,4 - 2,5 and 4,9 - 5,9 GHz	WLAN
Transmitter, Receiver		
Basic TX, Basic RX	1 to 250 MHz	Misc baseband operations
TVRX Receiver	50 to 860 MHz	VHF, DAB

Figure 2.2: Frequency range of several USRP/2 daughterboards. [4]

Daughterboards within the same family is interchangeable allowing fast change of frequency range or precision.

On most daughterboards, the signal is filtered, amplified and tuned to a baseband frequency depending on the IF bandwidth and local oscillator frequency. There also exists the so called Basic Rx/Tx boards with no frequency conversion or filtering. They only provide a very basic and mini RF frontend with direct connection to the motherboard. A list of commonly used daughterboards can be seen in the table in figure 2.2.

As an example the overall modular functionality and signal flow of a USRP N210 with a WBX daughterboard, can be seen on figure 2.3. This figure only contains the reception part of the module and daughterboard. A similar but mirrored version exists for the transmitting part.

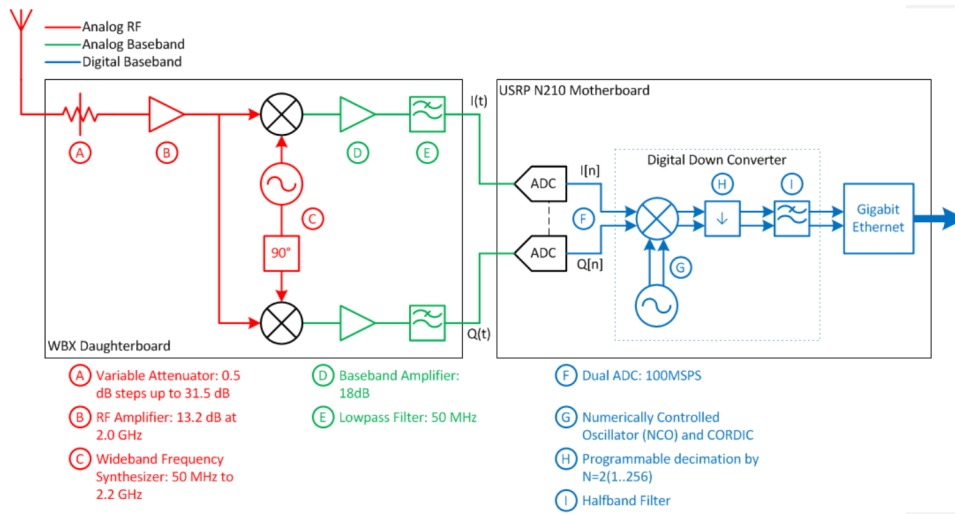


Figure 2.3: Modules within a USRP N210 and WBX daughterboard setup.

The left side of the figure contains the RF frontend on the WBX daughterboard while the right side contains the USRP main hardware including the ADC used for sampling. Most importantly is the frequency synthesizer within the RF frontend. This synthesizer generates a quadrature representation of the carrier frequency, to be mixed with the received signal. This splits the I and Q signals from the carrier, as described in chapter 1.3. After some low pass filtering, limited to the IR bandwidth of 50 MHz, the I and Q signals are sampled with the ADCs. Finally the samples are down converted to be transmitted thru the Ethernet link.

For the networked series of USRP modules the I and Q samples are transmitted over the Gigabit Ethernet link, which obviously limits the data rate. The theoretical data rate for a Gigabit Ethernet link is 125 MB/s. Using 4 byte complex samples, 16-bit I and 16-bit Q, and respecting the Nyquist criterion, leads to a usable complex RF bandwidth of about 31,25 MHz. A realistic limit of the usable bandwidth is 25 MHz though, mainly limited by the Ethernet link. [4]

The USRP N200-KIT module used within this document is listed with the following specifications:

- ADCs: 100 MS/s 14-bit
- DACs: 400 MS/s 16-bit
- Mixer: programmable decimation- and interpolation factors
- Max. BW: 50 MHz
- PC connection: Gigabit Ethernet (1000 MBit/s)
- RF range: DC – 5.9 GHz, defined trough RF daughterboards

Some of the larger units allows full-duplex communication by allowing two RF channels and antennas to be configured for either transmission or reception.

This is not the case for the USRP N200 modules though, which can only serve one purpose at a time, either being a transmitter or receiver. So to test a transmitter and receiver setup with these modules, two units will be necessary.

2.2 Connecting the USRP to a PC

To complete the modulation and demodulation test described within this document, two USRP N200 modules has to be connected to the same host PC at the same time. This can be done either thru a dedicated unmanaged Gigabit ethernet switch or by using a MIMO cable between the two units [8]. Please have in mind that if a Gigabit switch is used, the connection should not be shared with any other devices, routers or computers. Furthermore please make sure that the Ethernet cables used includes all cable pairs (eg. CAT6) to allow the required Gigabit bandwidth. The host PC should as well contain a Gigabit Ethernet interface.

If the ethernet switch option fails to succeed the USRP N200 includes a single MIMO port in each unit that can be interconnected. By combining two MIMO ports, the USRP units will be sharing a single ethernet interface and the MIMO interface will also allow special multiple antenna configurations.

MIMO is short for Multiple Inputs Multiple Outputs. In general by using a MIMO configuration you can increase the wireless system performance without increasing power consumption. When you use multiple antennas, the transmitted signal progresses through different wireless channels (from the transmitter antennas to the receiver antennas) and creates a capacity gain by exploiting channel diversity.

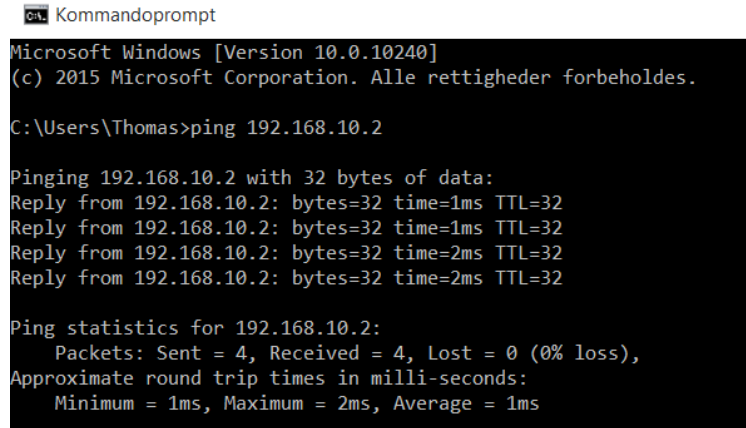
For the setup described within this document the MIMO cable is only used to share the ethernet connection. Connect a MIMO cable between the two units and connect a Gigabit ethernet cable between the host PC and one of the units. Both units should now be available on the network when the network settings has been changed accordingly.

2.2.1 PC Network settings

The first step of the connection process is to change the network settings of the host PC to match the default gateway address of the USRP modules and the use of static IP addresses. From the factory the USRP modules are configured with the IP address: 192.168.10.2. Go to the network settings and change your computer IP and netmask to the settings listed below. On a Windows PC this is done thru the Network control panel, which can be entered by writing *ncpa.cpl* in a command window. In this control panel select the properties for your ethernet card and go to the TCP/IPv4 properties.

- IP: 192.168.10.1
- Netmask: 255.255.255.0
- Gateway: 192.168.10.1

The USRP modules replies on ICMP ping requests why the connection can be tested by pinging the module IP address. If the IP address of the USRP unit has not a ping request to 192.168.10.2 should result in a reply, see figure 2.4.

A screenshot of a Windows command prompt window. The title bar reads 'Kommandoprompt'. The window content shows the following text:

```
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. Alle rettigheder forbeholdes.

C:\Users\Thomas>ping 192.168.10.2

Pinging 192.168.10.2 with 32 bytes of data:
Reply from 192.168.10.2: bytes=32 time=1ms TTL=32
Reply from 192.168.10.2: bytes=32 time=1ms TTL=32
Reply from 192.168.10.2: bytes=32 time=2ms TTL=32
Reply from 192.168.10.2: bytes=32 time=2ms TTL=32

Ping statistics for 192.168.10.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

Figure 2.4: Ping request and response from USRP N200.

Some modules might be reconfigured to another address by other users, eg. 192.168.10.3. If there is no response to the ping request it does not necessarily mean that the unit is not connected. A scan of USRP units on the network and any changes to be made to their addresses if necessary, is managed by the NI-URSP configuration utility described in chapter 3.2.

2.3 Control options

Now that the USRP module has been connected to the PC it is time to decide which control software to use. The USRP is an Open Source software-defined radio platform why multiple controller options exists. The commonly used control options includes:

- UHD C/C++ library
- MATLAB and Simulink
- LabVIEW
- GNURadio

The low-level way of interacting with the USRP modules is thru the Open Source UHD library [16]. This library includes all the handles and drivers to initialize a communication with one or multiple USRP units thru an Ethernet interface or USB connection, depending on the USRP model. This library allows developers to create C or C++ applications, integrating the functionality of a software-defined radio. The UHD library can be used within MATLAB and Simulink as well, even though pre-compiled binaries for these tools are readily available for download within MATLAB [12].

Within this document the National Instruments LabVIEW software will be used to interact with the USRP modules. LabVIEW is a graphical programming environment used to setup labsystems, measurement and test benches and signal processing applications. LabVIEW is ideal for the signal processing applications, as the software itself is optimized for real-time execution of graphically programmed applications.

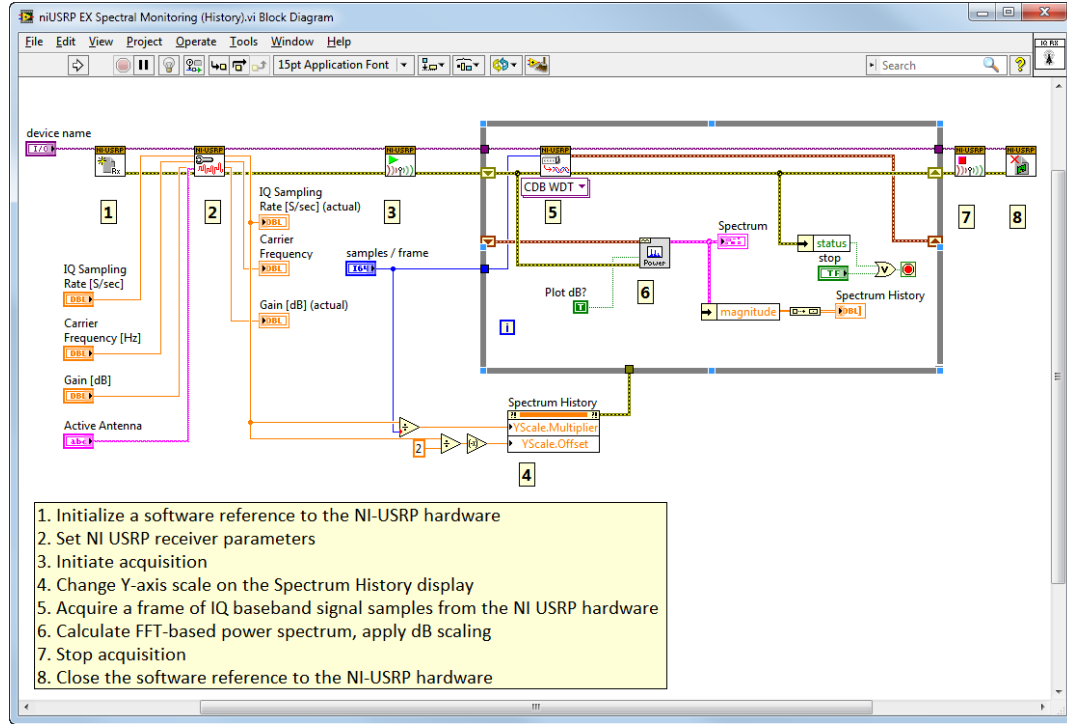


Figure 2.5: LabVIEW graphical programming environment. [10]

The support of the USRP modules is added by National Instruments [11] as a support package, as they recommend their SDR users to use the USRP modules from Ettus. The support within LabVIEW does not rely on the UHD library, which to some extent makes it more reliable.

One of the higher-level Open Source alternatives to LabVIEW is GNURadio which is also supported by Linux. GNURadio is a graphical programming environment for software-defined radio applications, see figure 2.6, supporting a variety of different SDR hardware such as the USRP modules from Ettus. Programming can be done in a similar fashion as LabVIEW, by dragging programming blocks into a workspace or writing customized Python code.

GNURadio is not as capable as LabVIEW when it comes to external interfaces and the amount of programming blocks. Though GNURadio is a great free alternative that supports many major modulation and demodulation schemes out of the box, including a wide variety of example projects.

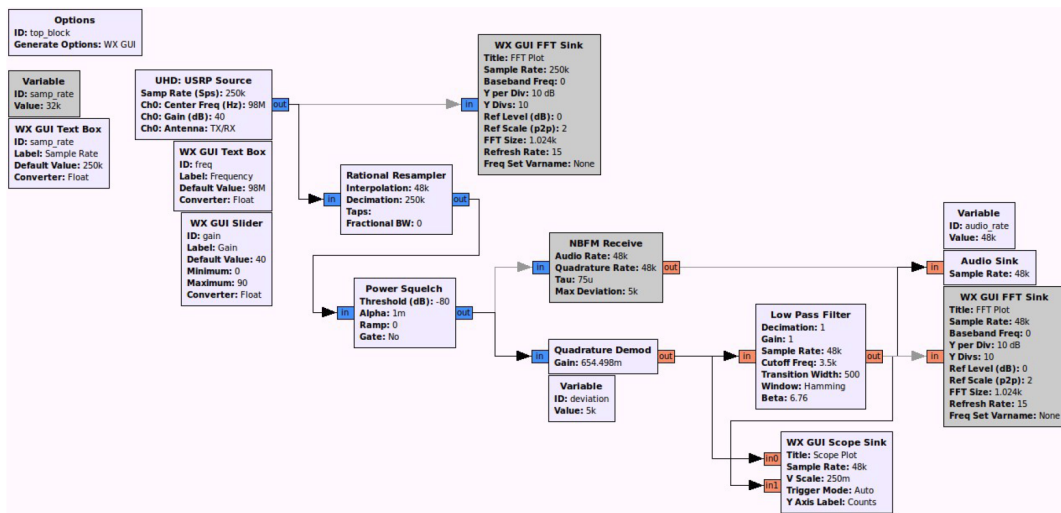


Figure 2.6: GNURadio open source graphical programming environment. [2]

On Linux systems GNURadio can be installed thru *apt-get* while Windows users will have to use Cygwin or similar. A great presentation of how to use the USRP modules with GNURadio can be found in reference [15]. Furthermore some recommended lab-exercises by Ettus for GNURadio can be found in reference [2].

Chapter 3

LabVIEW and USRP

This chapter is written as a step by step guide on how to install the right and get started using the USRP N200 module with LabVIEW. If you are already familiar with LabVIEW and the NI-USRP package, but interested in how to develop and test a modulation schemes thru LabVIEW, you can skip to chapter 4.

A basic knowledge about LabVIEW and its' components is a requirement to this guide. Otherwise a good general presentation about LabVIEW and how to integrate with the USRP can be found in reference [7].

A LabVIEW programming environment consists of both a user interface part and a programming part. The user interface part works with a drag and drop palette of visualization blocks, such as buttons, input fields, knobs, graphs, numerical displays etc. The programming part is done graphically, similar to flowchart programming or block-based programming, where individual functions is separated into blocks with dedicated input and outputs. When running a LabVIEW application the host PC will run the programmed application in real-time, taking in and processing any requested measurements from external interfaces such as the USRP module.

3.1 What to install

Initially you will have to make sure that you have the right version of LabVIEW and toolkits installed. Within this document the original LabVIEW software will be used, rather than the newly released LabVIEW Communication System Design Suite, intended for software-defined radio applications. The main reason for this is to show the support for the USRP modules within the original LabVIEW, which many universities have a license for already.

For use with signal processing applications and especially software-defined radios, the LabVIEW for Data Acquisition is recommended. This software can be found from the following link: <http://www.ni.com/download-labview/measurement-type/>.

To use the USRP modules with LabVIEW, the NI-USRP package would have to be downloaded and installed. This package includes all the required blocks to initialize a connection with a USRP module together with the right blocks for transmitting or receiving I and Q samples. The package used in this document is version 14.0 which can be downloaded here: <http://www.ni.com/download/ni-usrp-14.0/4999/en/> A newer version, 15.0, has been released. This version has not been tested to be compatible with this document, but only few changes has been made, mainly making the package compatible with LabVIEW 2015.

Finally to enable further modulation blocks, the National Instruments Modulation Toolkit is recommend. This is not a requirement to follow this guide, but is recommended as it includes most of the commonly used modulation schemes in easy to use blocks. This toolkit can be downloaded from the following link: <http://sine.ni.com/nips/cds/view/p/lang/da/nid/210568>.

When everything has been installed properly you should be able to start both LabVIEW and the NI-USRP Configuration Utility, which will be our primary applications within this document.

3.2 NI-USRP Configuration Utility

With the use of the NI-USRP Configuration Utility it is possible to change the IP address of the connected USRP modules. With a single module connected thru the Ethernet cable a screen as shown in 3.1 should appear.

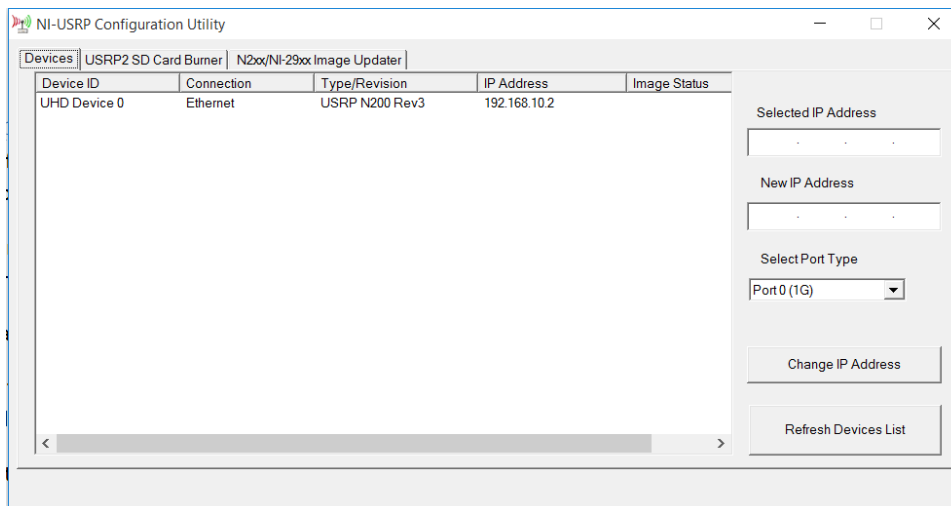


Figure 3.1: NI-USRP Configuration Utility with one USRP module connected.

3.2.1 Configuring multiple USRP modules

To use both modules at the same time, as required to do transmission and reception at the same time, the IP address of each module has to be unique. To make it simple the two modules should be configured to have IP address *192.168.10.2* and *192.168.10.3* as the host PC has address *192.168.10.1*. Thru the NI-USRP configuration utility it is possible to change the IP address of a desired module. Highlight the module in the list, which updates the IP address fields to the right, as shown in figure 3.2. Change the IP address to the one for the first module and press the 'Change IP Address' button.

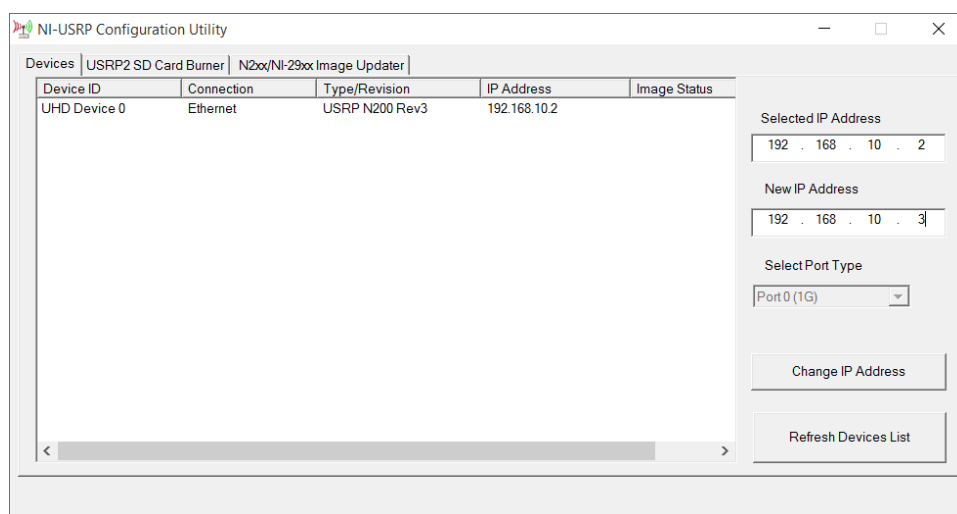


Figure 3.2: Change IP address thru NI-USRP Configuration Utility

Repeat the step by connecting the second module using a MIMO cable and refresh the list of devices. Change the IP address of the second module to the desired one, eg. *192.168.10.3*. When both modules have been connected and configured, the configuration window should look like the one in figure 3.3. Verify that each module has a unique IP address.

Device ID	Connection	Type/Revision	IP Address	Image Status
UHD Device 0	Ethernet	USRP N200 Rev4	192.168.10.2	
UHD Device 1	Ethernet	USRP N200 Rev3	192.168.10.3	

Figure 3.3: Two USRP modules connected and configured properly.

The IP address of both modules has now been configured and verified to be correct and unique. This allows the following LabVIEW applications to access the modules to transmit or receive I and Q samples.

3.2.2 Firmware update

If any of the modules is listed as requiring a firmware update, the 'N2xx/NI-29xx Image Update' tab has to be used. Within this tab it is possible to select and load a desired firmware image and FPGA image, see figure 3.4. For the examples and projects within this document, the basic USRP firmware image, that comes with LabVIEW, has to be used. This is as well the firmware image that the USRP modules is pre-programmed with.

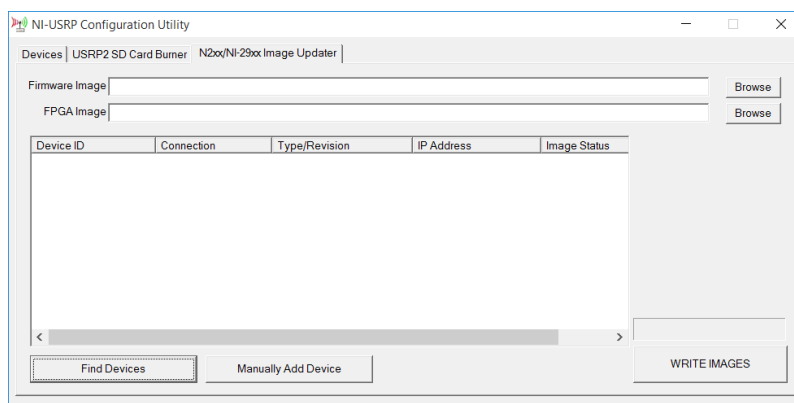


Figure 3.4: 'N2xx/NI-29xx Image Update' tab within the NI-USRP configuration utility.

To update any of the connected USRP modules press the 'Find Devices' button and highlight the module that has to be updated. Chose the firmware image and FPGA image, found within the installation folder of the NI-USRP package, by using the two 'Browse' buttons. See figure 3.5. Both images should match the USRP module, in this case the N200. When both firmware files have been selected, the 'Write Images' button will initialize the programming which takes a couple of minutes to complete.

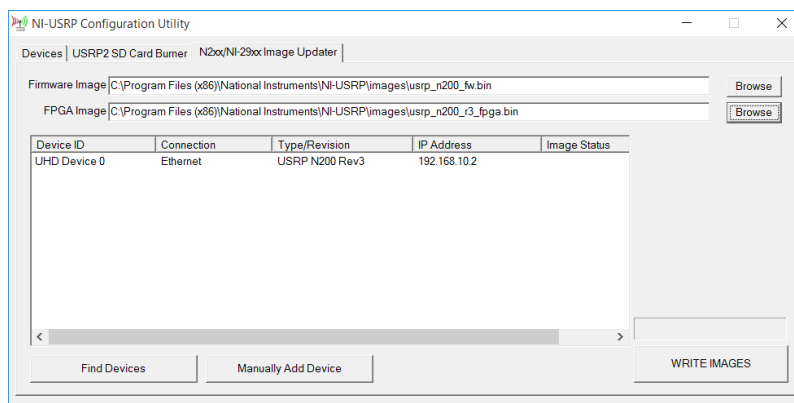


Figure 3.5: Selected firmware files for USRP N200 module.

Both modules should now be connected, running and properly visible within the NI-USRP configuration utility, each with individual IP addresses. This allows the examples, that comes with the NI-USRP package, to be tested.

3.3 Running your first example

The NI-USRP package comes with a large list of example projects, both for transmission and reception. The examples is located within the LabVIEW installation folder rather than the NI-USRP package installation folder:

C:\Program Files (x86)\National Instruments\LabVIEW 2015\examples\instr\niUSRP

If this is the first experience with LabVIEW, it is a good idea to study the general interface, block palette, help window and especially the shortcuts. One usefull shortcut is CTRL+H which opens the help window that in programming mode provides information about the specific block being hovered. Another usefull shortcut is CTRL+E which switches between the visualization panel and programming mode.

3.3.1 niUSRP EX Spectral Monitoring example

To test the reception of RF signals with one of the USRP modules the *niUSRP EX Spectral Monitoring (Interactive).vi* example gives a good impression of any nearby RF signals. Open the example thru the regular LabVIEW file dialog.

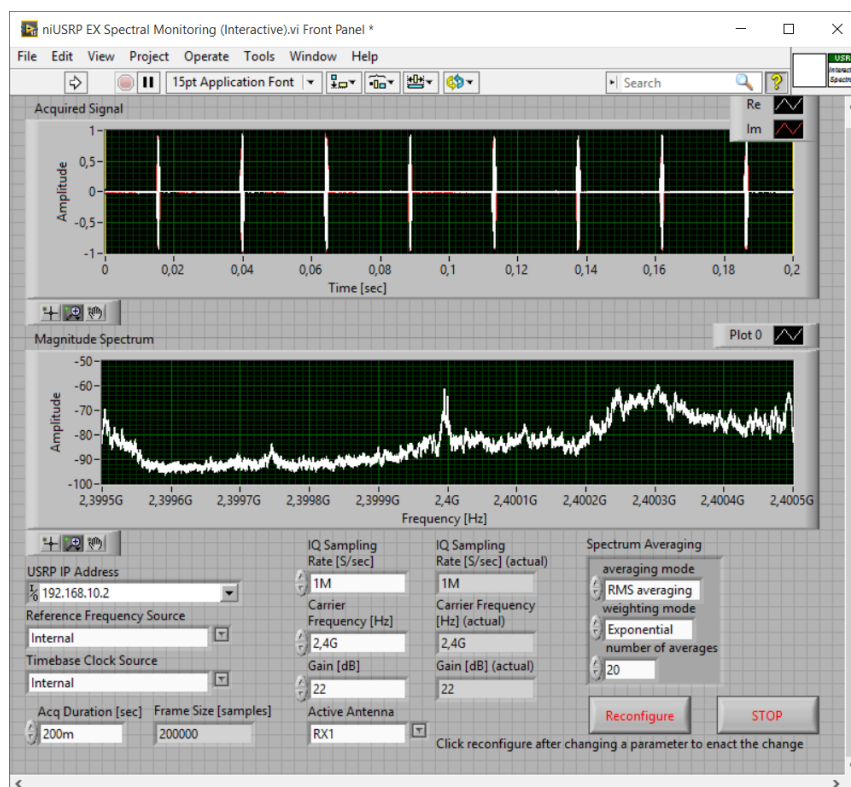


Figure 3.6: Spectral monitoring example.

This opens the visualization panel of the example, including the input fields for the required settings. Type in the IP address of one of the USRP modules, set the carrier frequency according to your antenna and daughterboard, set the IQ sampling rate to 1M or similar and the acquisition duration a couple of hundred milliseconds. Press the Go button, the right-pointing arrow in the top, and the connection to the USRP module will be established.

Depending on the surrounding RF signals both the time domain signal and magnitude spectrum will display the signal content. As an example, multiple WiFi channels would be noticeable within the magnitude spectrum as closely located but unique spikes. On figure 3.6 the signal from a 2,4 GHz RC transmitter is shown. It is very clear that a package is transmitted every 200 ms.

Take into consideration that the IQ sampling rate defines the frequency of how fast the I and Q samples is sampled, and therefore also defines the measurable signal bandwidth. The limitations of a sampled signal, some of them given by the Nyquist rate, also applies for the I and Q samples. Depending on the signal that has to be measured, the I and Q sampling rate has to be set accordingly.

3.4 USRP blocks within LabVIEW

Being within the visualization panel of a LabVIEWproject, the CTRL+E shortcut can be used to switch into programming mode. The window should change into a page of interconnected blocks. This is the programming mode where each block contains an individual functionality while being assembled in a larger configuration gives the functionality of a running application with inputs and outputs.

The NI-USRP package provides a couple of new palettes with the blocks to control the USRP modules. Open the palette window by going to View → Functions Palette. Within this palette the NI-USRP blocks are located under Express → Output → Instrument Drivers → NI-USRP. This should result in a palette menu with six items, five of them being categories, see figure 3.7. From left to right these items are: Rx, Tx, Property Node, Synchronization, Utility and niUSRP.

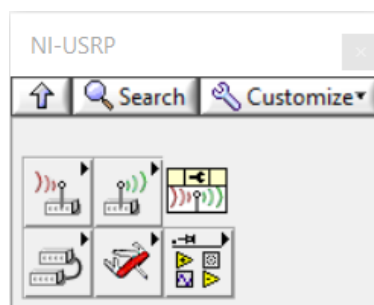


Figure 3.7: NI-USRP block categories.

The two most important items are the Rx and Tx categories. When making a transmitter application, only the blocks within the Tx category is required to make a functional application. For receiver applications only the blocks within the Rx category is required.

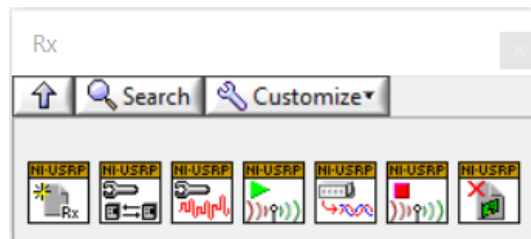


Figure 3.8: NI-USRP receiver blocks.

By clicking one of the categories the contained blocks will be displayed. For receiver applications seven different blocks can be used, see figure 3.8. Starting from the left these blocks are: Open Rx Session, Configure Number of Samples, Configure Signal, Initiate, Fetch Rx Data (poly), Abort, Close Session.



Figure 3.9: NI-USRP transmitter blocks.

From left to right the blocks of the transmitter palette are: Open Tx Session, Configure Signal, Write Tx Data (poly), Close Session.

The only main difference between the receiver and transmitter blocks is the Initiate and Abort blocks which is used to start and stop the event handler of incoming I and Q samples. Otherwise both the opening, configuration and closing is the same.

To get more information about a specific block, including which inputs and outputs it contains, open the help window by pressing CTRL+H, and hover the mouse above the block. The help window of the Configure Signal block can be seen in figure 3.10. Inputs are displayed to the left of the block and outputs to the right.

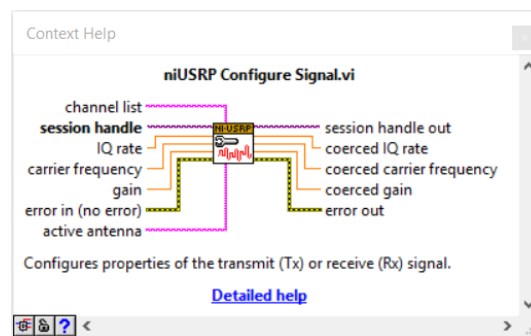


Figure 3.10: Configure Signal help window.

The session handle and error input/output is the two general wires used to connect all the used USRP blocks together. The session handle is generated with the Initiate block which the error in takes in and forwards any previous error, to abort everything if an error occurs.

As seen in the help window the Configure Signal block is used to configure the IP address, input or output channel, gain, carrier frequency and IQ sampling rate.

While this is a very short introduction of the individual USRP blocks, this gives a basic knowledge of the USRP block layout. This knowledge enables the general understanding of the following programming figures of a receiver and transmitter example.

3.4.1 Receiver example

Any USRP receiver application relies on the common usage flow in figure 3.11.

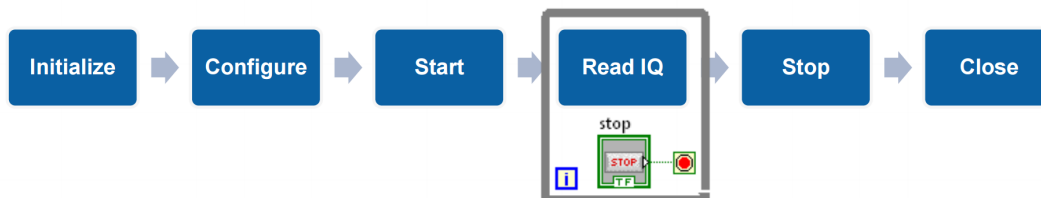


Figure 3.11: General usage flow of receiver applications.

The first two steps is to initialize a session handle for the USRP module and configure the handle with the required settings, described above. For a receiver application a start trigger is required to initiate the reception event handler. Within a while-loop the received I and Q samples are read thru the 'Fetch Rx Data (poly)' block. Finally when the reception is finished the event handler has to be stopped and the session terminated.

Applying the usage flow to the LabVIEW programming environment, results in a receiver example capable of receiving and displaying the I and Q samples, shown in figure 3.12.

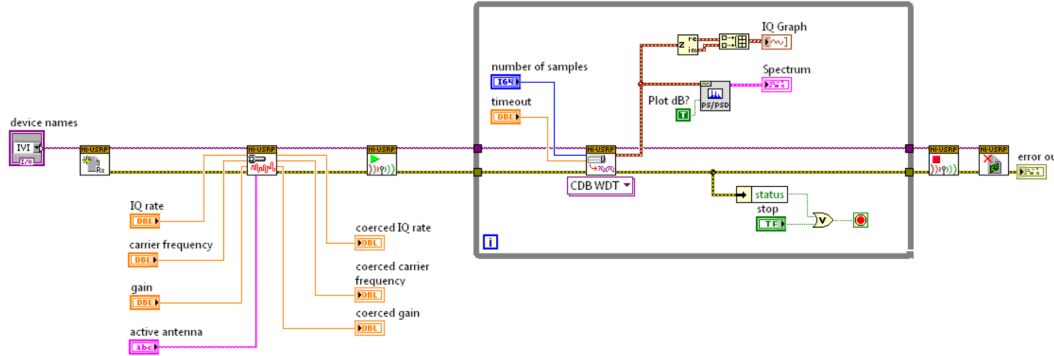


Figure 3.12: IQ receiver example with USRP blocks.

The reading of the I and Q samples happens within the 'Fetch Rx Data (poly)' block with the 'CDB WDT' writing beneath. The 'CDB WDT' writing tells LabVIEW that the acquired data should be output as complex doubles, which is then split into the real and imaginary part in the z-separator above. The power spectrum of the complex signal is shown as well thru the pink Spectrum graph block.

3.4.2 Transmitter example

Any USRP transmitter application relies on the common usage flow in figure 3.13.

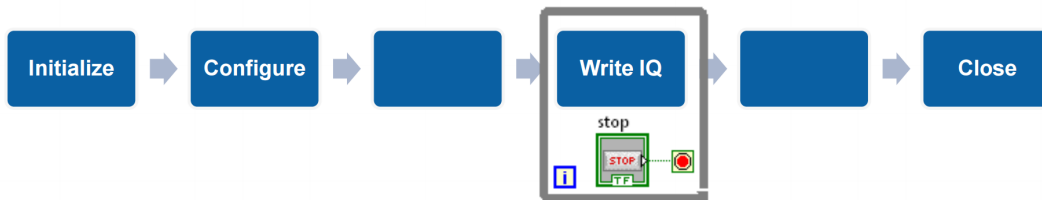


Figure 3.13: General usage flow of transmitter applications.

The steps are identical to the receiver usage flow, except for the Start and Abort which is not required for transmission. Within the while-loop the generated I and Q samples are provided to the 'Write Tx Data' block for transmission. This block includes a buffer, why generated samples can be supplied in chunks to keep the buffer loaded at all times.

Before testing the transmission of I and Q samples, make sure an antenna is installed in the right SMA connector on the USRP module. RF1 can be used for both transmission and reception, but only one way at a time, so it is recommended to install the antenna in the RF1 SMA connector.

Implementing the usage flow within the LabVIEW programming environment, results in a transmitter example capable of transmitting I and Q samples from an external complex dataset given a specific sample rate. This example is shown in figure 3.14.

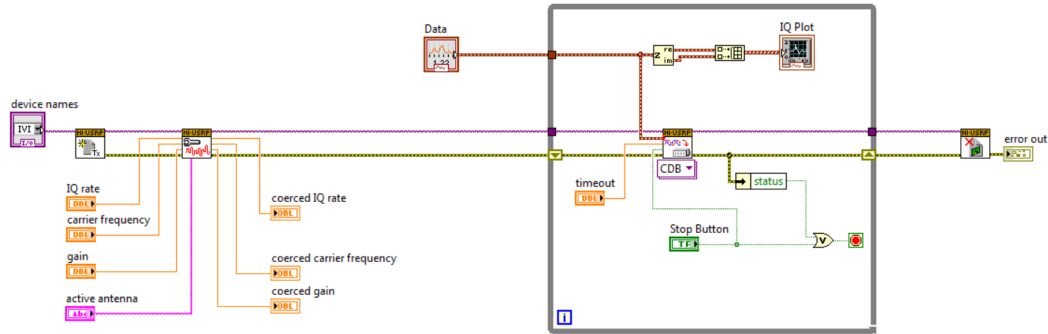


Figure 3.14: IQ transmitter example with USRP blocks.

Within the while-loop the samples are written continuously until the stop button is pressed or a timeout or error occurs. The dataset could be changed into an internally generated set of samples to allow the LabVIEW application to generate and modulate the signal.

This concludes the introduction of the LabVIEW environment with the NI-USRP package to allow communication and control of the USRP modules.

Chapter 4

AM Modulation

With both USRP N200 modules running and connected to a host PC, with a working installation of LabVIEW, it is now possible to implement and test the simple AM modulation scheme. Throughout the following chapter the theory behind AM modulation will be presented briefly followed by the implementation of a single tone AM modulator and a audio file AM modulator. Finally an AM demodulator with frequency spectrum visualization, time domain visualization and audio-playback will be implemented.

The LabVIEW project files for the modulators and demodulator can be downloaded from the following link: http://www.tkjelectronics.dk/downloads/sdr/usrp_am A video showing the implemented modulators and demodulator on two USRP N200 modules can be seen here: <https://www.youtube.com/watch?v=uBmhNn2sRAI>

4.1 AM modulation scheme

One of the simplest modulation schemes is AM modulation, short for Amplitude Modulation. The general concept of AM modulation is to change the amplitude of the carrier signal according to the message signal. This results in a carrier frequency envelope corresponding to the actual message signal.

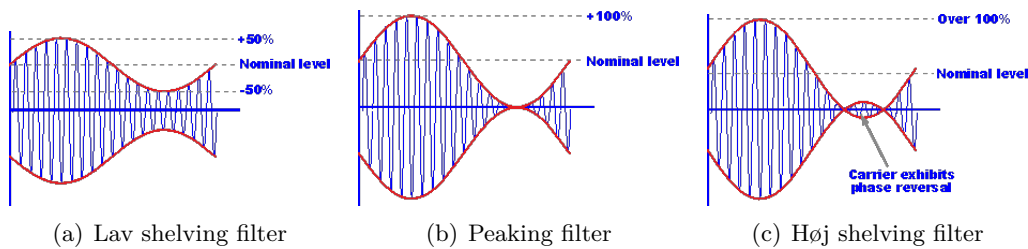


Figure 4.1: AM modulation with different modulation index. [6]

When describing the baseband signal of an AM modulated signal, the amplitude

of the baseband is directly proportional to the message signal while the phase is constant.

$$a(t) = A_c[1 + k_a m(t)] \quad (4.1)$$

$$\phi(t) = 0 \quad (4.2)$$

Where A_c is the baseband signal gain and k_a is the modulation index, see figure 4.1. For the USRP modules the baseband signal amplitude has to be kept within ± 1 before being transmitted as I and Q samples. At 100% modulation and with a normalized message signal, the baseband signal gain has to be set to $A_c = 0.5$.

The notation of the I and Q samples, from equation 1.6, can be used to extract the corresponding I and Q samples when using AM modulation, given by the message signal.

$$I(t) = A_c[1 + k_a m(t)] \quad (4.3)$$

$$Q(t) = 0$$

The content of the I and Q samples, given by equation 4.3, can be implemented in LabVIEW by simple math blocks. This gives the modulator part.

For the demodulator part the I and Q samples will be received from where the message signal can be extracted. Due to the signal transmission and quadrature demodulation, the constant phase has not necessarily been kept zero. Therefore the message signal can not be extracted directly from the I samples. A combined magnitude of the complex baseband signal can be used to get rid of the phase offset and extract the message signal.

$$|I(t) + jQ(t)| = A_c[1 + k_a m(t)] \quad (4.4)$$

As the baseband signal might get attenuated due to the signal transmission, the baseband signal gain is unlikely to be the same. As long as the modulation index is 100% or below, we can use the maximum and minimum value from a frame of samples to scale and offset the samples to get the message signal.

$$mag(t) = |I(t) + jQ(t)| \quad (4.5)$$

$$\text{Scale} = \frac{2}{\max(mag(t)) - \min(mag(t))} \quad (4.6)$$

$$\text{Offset} = \text{Scale} \cdot \min(mag(t)) + 1 \quad (4.7)$$

With the known scale factor and offset of the baseband magnitude, these factors can be applied to extract the message signal.

$$m(t) = \text{Scale} \cdot mag(t) - \text{Offset} \quad (4.8)$$

These equations can be implemented in LabVIEW as well to form the demodulator. The implementation of the modulator and demodulator follows in section 4.2 and 4.3.

4.2 USRP as AM modulator

The implementation of the modulator is done in two steps to show the difference between the modulation of a single tone modulator and an audio waveform. The LabVIEW programming of both modulators can be found in appendix A.

4.2.1 Single tone modulation

For a single tone AM modulator the message signal is simply represented by a sine wave.

$$m(t) = \sin(2\pi f_m t) \quad (4.9)$$

Within LabVIEW a waveform generator block can be used to create this single sine wave. The generated waveform is converted into the required I and Q samples by using equation 4.3. The actual implementation of this equation is done inside the waveform generator by setting the amplitude to 0,5 and offsetting the signal by 0,5. The waveform generator output is combined into a complex double as the real part, where the imaginary part, being the Q samples, are set to 0.

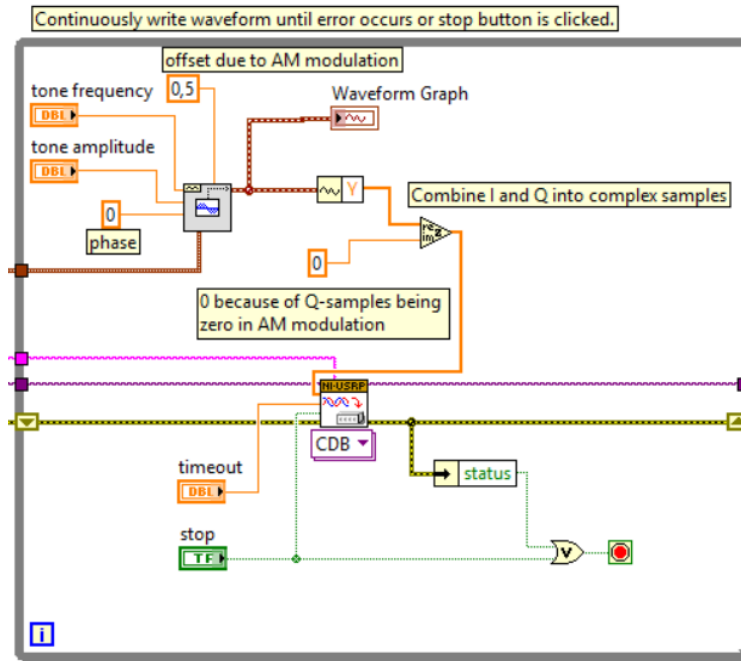


Figure 4.2: Single tone AM modulator while-loop.

According to the USRP usage flow from chapter 3.4.2, a while loop should contain the generation and writing of the I and Q samples. The generator and math blocks are therefore put into a while-loop containing the USRP 'Write Tx Data (poly)' block, shown in figure 4.2. This results in the desired single tone AM modulator.

To give a visual representation of the message signal being transmitted, a visualization panel shown in figure 4.3 is made. This panel includes all the necessary.

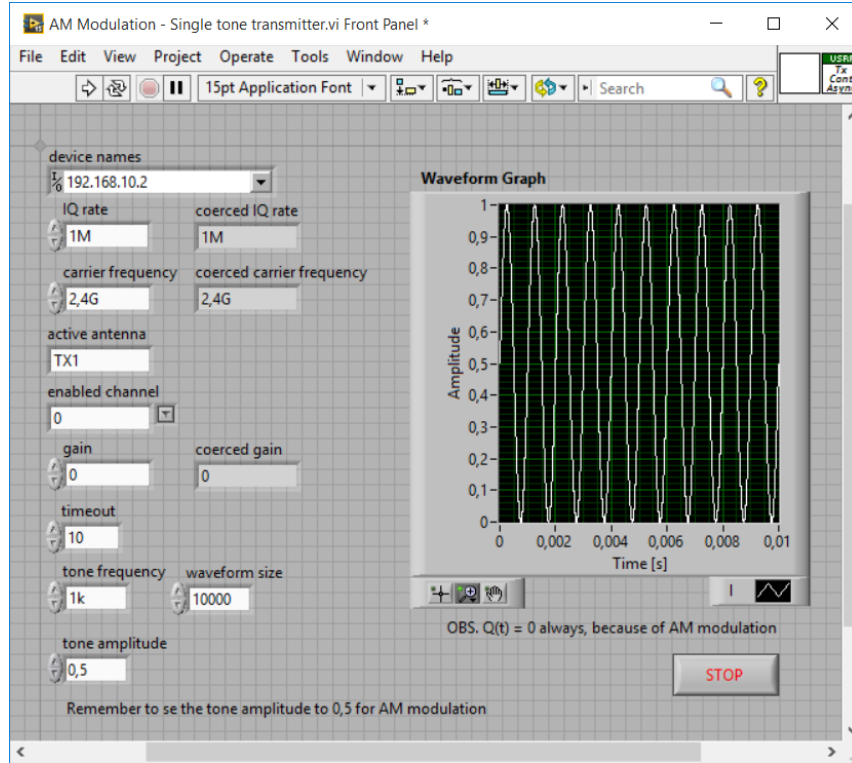


Figure 4.3: Visualization panel for single tone modulator.

The modulator is tested by typing in an IP address for one of the USRP modules and setting the carrier frequency. Please be aware that the tone amplitude has to be set to 0,5 to match the amplitude modulation scheme. The IQ sampling rate has to be set high enough to avoid aliasing with the tone frequency, eg. refer to Nyquist. In the figure the sampling rate is set to 1M and the tone frequency to 1 kHz.

The full LabVIEW circuit of the single tone AM modulator can be found in appendix A.1.

4.2.2 Audio-file modulation

Modulating a single tone is very basic and good for the understanding of the AM modulation scheme. Though LabVIEW easily allows much more complex signal processing, why the modulation of an audio-file can be done with a short circuit. LabVIEW has a few built in function blocks to load, read and resample an audio-file. The while loop for the generation and writing of the I and Q samples is shown in figure A.2.

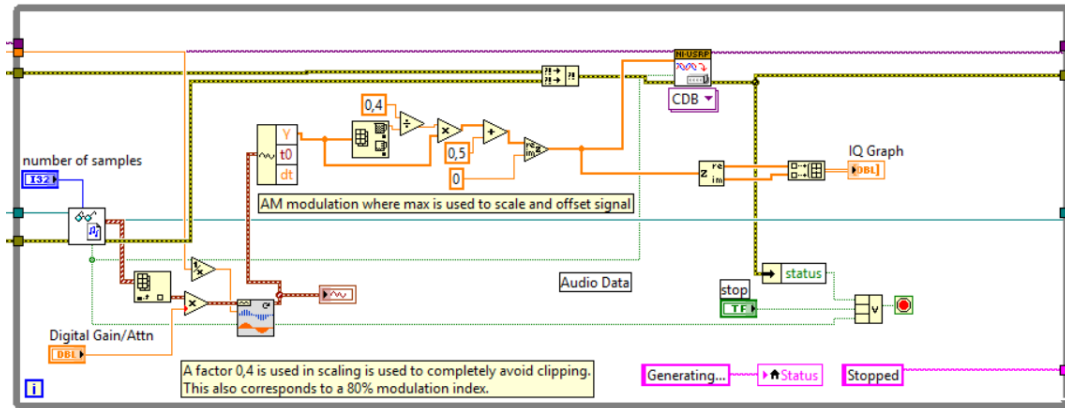


Figure 4.4: Audio-file AM modulator while-loop.

In this figure the reading and resampling is done in the left part of the circuit. The resampling matches the sample frequency of the audio signal with the sample frequency of the IQ sampling rate, set to 1M within this example. The output waveform of the audio signal is fed into the AM modulation part, seen in the top of the circuit. The max amplitude of the signal waveform is used to scale the audio waveform to an amplitude of maximum $\pm 0,4$. A factor of 0,4 is used to avoid clipping of very sudden loud signals. This scaled signal waveform is finally offset by 0,5 before being combined into the complex I and Q waveform.

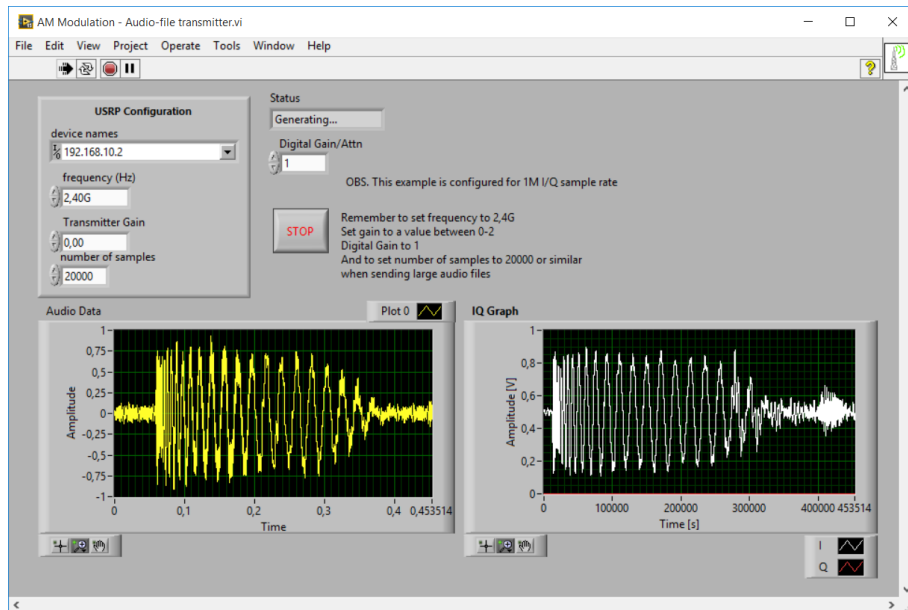


Figure 4.5: Visualization panel for audio-file modulator.

A visualization panel is designed to allow the visualization of the transmitted samples, see figure 4.5.

The panel includes the settings of the USRP module and the number of samples, representing the waveform buffer size. The larger a buffer, the more samples of the waveform is included at each write to the USRP module. Too few samples and the buffer inside the USRP will underrun. For a 1M IQ sampling rate, a buffer size of 20.000 is a good size that allow a smooth audio without buffer underrun.

The full LabVIEW circuit of the audio-file modulator can be found in appendix A.2.

4.3 USRP as AM demodulator

The demodulation of the AM modulated message signal is simply the reverse of the modulation, shown in figure A.2. According to equation 4.5 to 4.8 the message signal is extracted from the received I and Q signal by doing the opposite scaling and offsetting, as when the signal was modulated. According to the USRP receiver flow from chapter 3.4.1, the reading and processing of the I and Q samples should happen in a while-loop. The while-loop, shown in figure A.3, contains the USRP 'Fetch Rx Data (poly)' block to get the I and Q samples, which are delayed for one buffer period by using a shift register. This delay is added to match the processing time to the buffer period, as the demodulation and resampling of the received signal takes time.

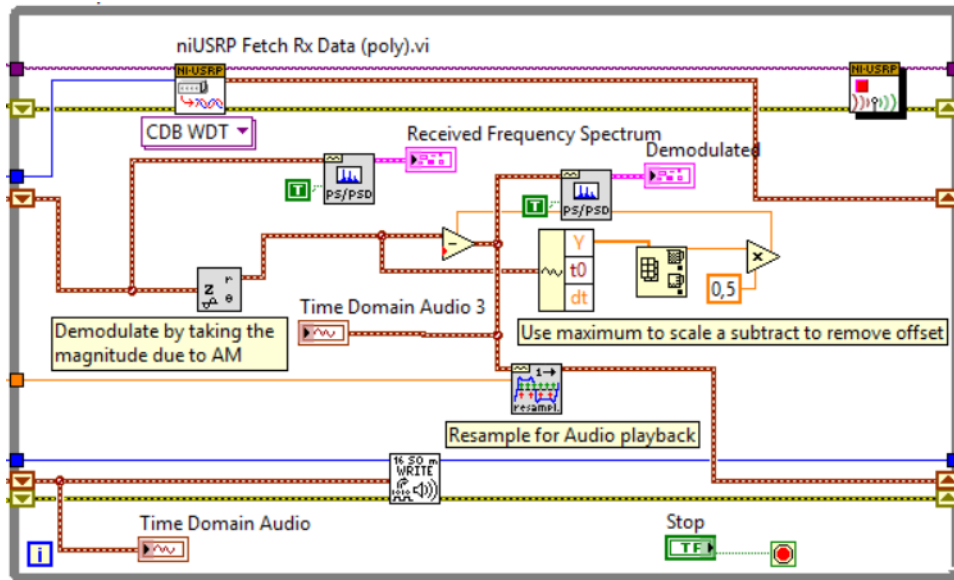


Figure 4.6: AM demodulator while-loop with audio generator.

The AM demodulation part of the circuit is located in the middle. The received I and Q samples are split into the magnitude and phase part, according to equation 4.5. When received, the I and Q samples will have an offset due to the selected AM modulation scheme. This offset is removed by offsetting the magnitude with the half of the maximum signal amplitude.

The resulting waveform is resampled with the sampling frequency of the audio card, eg. 44,1 kHz, so it can be played out thru the speakers of the PC. Furthermore the waveform and spectrum of the received signal is displayed in the two waveform graphs on the visualization panel, see figure 4.7.

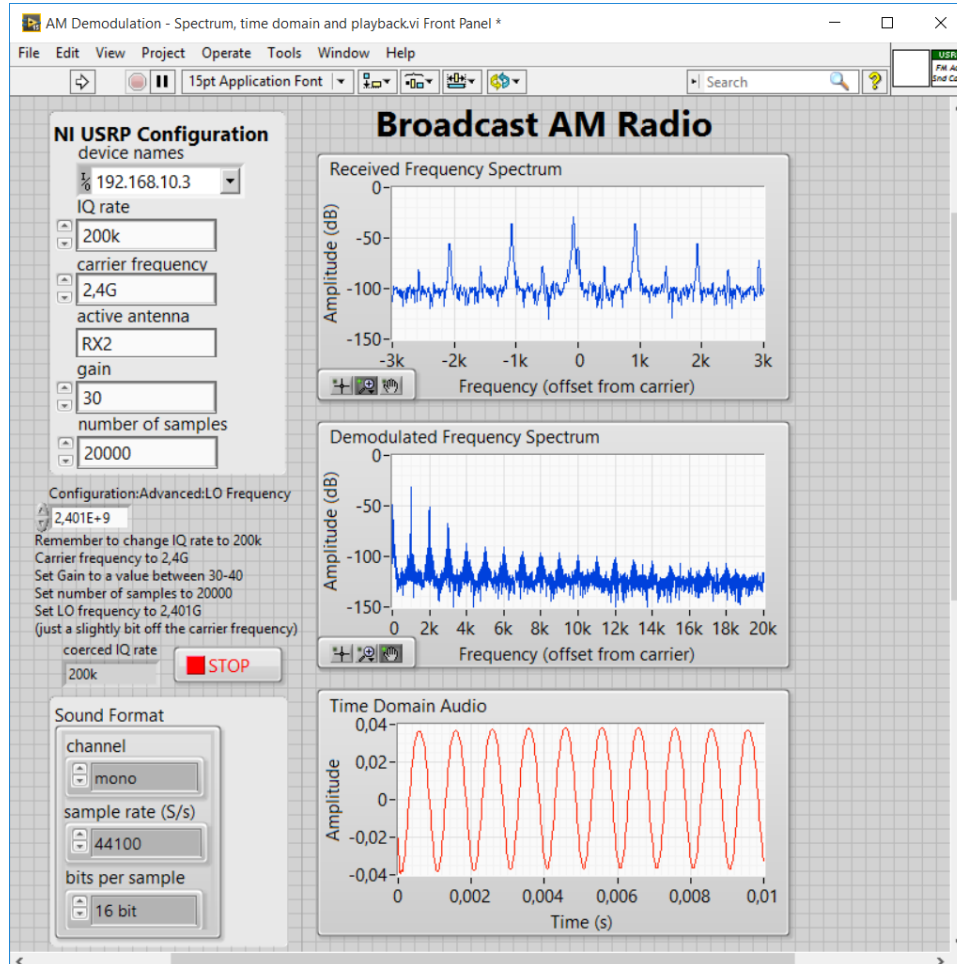


Figure 4.7: Visualization panel of AM demodulator with a single tone transmission.

In the visualization panel in figure 4.7 the IQ sampling rate is set to 200 kHz which is fine for audio applications with a message bandwidth of approximately 20 kHz. The received signal and displayed waveform in 4.7 comes from the single tone AM modulator from chapter 4.2.1, with the transmission of a 1 kHz sine wave. This is clearly seen in the demodulated frequency spectrum, where the largest magnitude of frequency content is located at 1 kHz.

The visualization panel in figure 4.8 shows the reception of the transmitted audio signal from chapter 4.2.2 and figure A.2. At reception the signal is demodulated, visualized and played thru the PC speakers in real-time, so changes in the waveform displays of both the transmitter and receiver applications happens simultaneously.

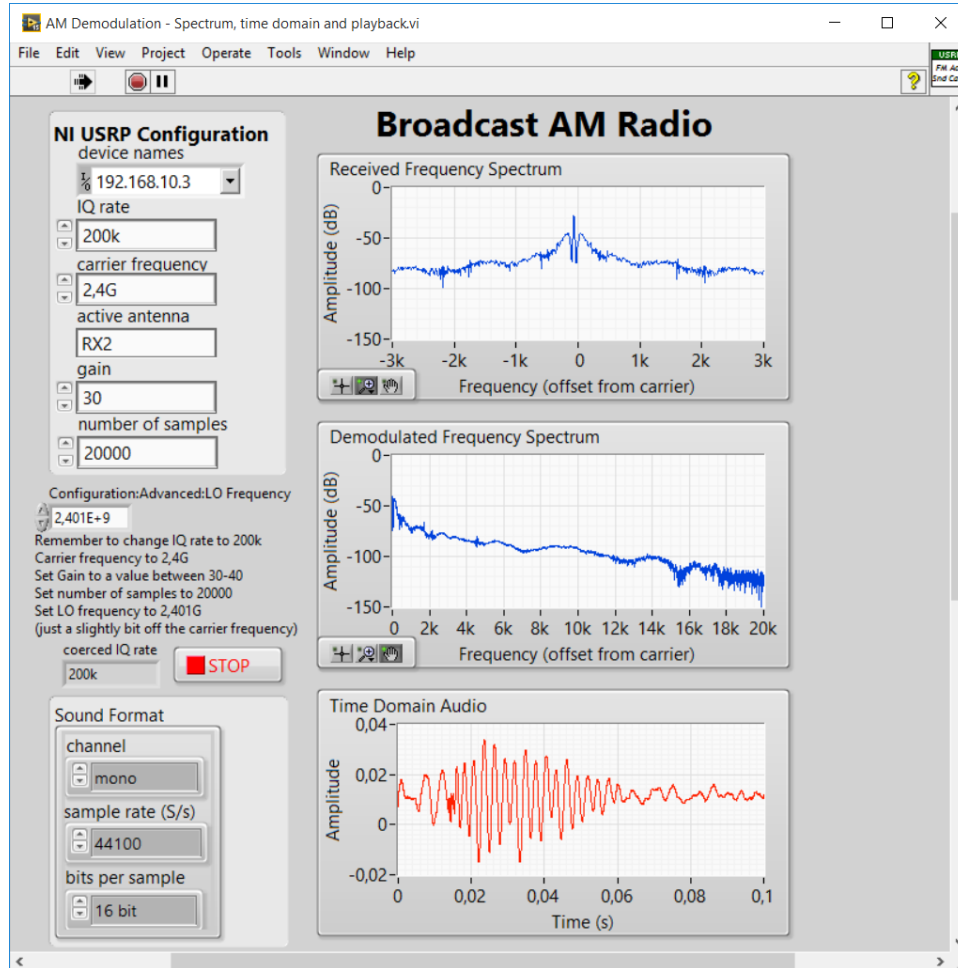


Figure 4.8: Visualization panel of AM demodulator with an audio-signal transmission.

The full LabVIEW circuit of the AM demodulator can be found in appendix A.3.

Notice that the AM demodulator contains a block to change the LO frequency of the USRP module. AM modulation is in general a very sensitive modulation scheme, as the modulation scheme itself contains an important DC offset. This DC offset might be filtered away due to the LO circuit as shown in figure 2.3. It is not recommended to run an AM modulation scheme on the LO frequency, which is by default set to the carrier frequency. Instead the LO frequency should be set just slightly higher or less than the carrier frequency. More information about the LO frequency compared to the carrier frequency can be found in reference [9].

Chapter 5

Conclusion

Initially an introduction to software-defined radios was given with respect to the details of the theory behind I and Q samples, used to transmit and receive baseband signals from a software-defined radio. To get started using software-defined radios, the USRP modules manufactured by Ettus provides a great starting point. Two USRP N200 modules was connected together thru a MIMO cable to allow the sharing of the Ethernet connection. The two modules was configured to be used within a LabVIEW environment on a Windows PC, even though Linux could be used as well with eg. GNURadio.

The theory of AM modulation was applied to the USRP modules, where one of the modules was programmed as a transmitter and the other as the receiver. Thru the use of the graphical programming environment within LabVIEW, both a single tone and an audio file was modulated and transmitted. At the receiver the modulated AM signal was demodulated and shown within a frequency spectrum and time domain graph and finally resampled into the sampling frequency of the audio card for playback.

For other modulation schemes, such as FM modulation, many guides, descriptions and literature exist about how to implement these within LabVIEW using an USRP module. Two good National Instrument resources on FM modulation can be found at the following two links: <http://www.ni.com/white-paper/3361/en/> and <http://www.ni.com/white-paper/13193/en/>.

Bibliography

- [1] DesignSpark Andrew Back. *10 Things You Can Do with Software-Defined Radio*. <http://www.rs-online.com/designspark/electronics/blog/10-things-you-can-do-with-software-defined-radio>. Sept. 2013.
- [2] Ettus Research Balint Seeber. *GNU Radio Tutorials*. http://files.ettus.com/tutorials/labs/Lab_1-5.pdf. Apr. 2014.
- [3] *Ettus Research*. <http://www.ettus.com/>.
- [4] Matthias Fahnle. *Software-Defined Radio with GNU Radio and USRP/2 Hardware Frontend*. Tech. rep. http://www.hs-ulm.de/opus/volltexte/2010/27/pdf/sdr_gnuradio_usrp_feb2010.pdf: Hochschule Ulm, University of Applied Sciences, 2010.
- [5] GomSpace. *GomSpace Presentation*. <http://www.itu.int/en/ITU-R/space/workshops/2015-prague-small-sat/Presentations/GomSpace.pdf>. Mar. 2015.
- [6] Radio-Electronics Ian Poole. *Amplitude Modulation Index & Depth*. <http://www.radio-electronics.com/info/rf-technology-design/am-amplitude-modulation/modulation-index-depth.php>.
- [7] National Instruments Ingo Foldvari. *Introduction to Digital Communication*. <http://papenlab1.ucsd.edu/~wes/downloads/Matlab:LabView%20Tutorials/Introduction%20to%20Digital%20Communication,%20Ingo%20Foldvari.pdf>.
- [8] National Instruments. *2 x 2 MIMO With NI USRP*. <http://www.ni.com/white-paper/13878/en/>. 8.
- [9] National Instruments. *LO Frequency Property*. http://zone.ni.com/reference/en-XX/help/373380B-01/usrppropref/pniusrp_lofrequency/. Apr. 2012.
- [10] National Instruments. *Spectrum Monitoring With NI USRP*. <http://www.ni.com/white-paper/13882/en/>. Apr. 2015.
- [11] National Instruments. *USRP*. <http://www.ni.com/sdr/usrp/>.
- [12] MathWorks. *USRP Support from Communications System Toolbox*. <http://www.mathworks.com/hardware-support/usrp.html>.
- [13] Whiteboard Web Mikael Q. Kuisma. *I/Q Data for Dummies*. <http://whiteboard.ping.se/SDR/IQ>. Oct. 2015.

- [14] USRP N200. *Ettus Research*. <http://www.ettus.com/product/details/UN200-KIT>.
- [15] NCSU. *USRP/GNU Radio Tutorial*. <http://www.krnet.or.kr/board/data/dprogram/1779/D1-1-KRnet2013.pdf>. June 2013.
- [16] Ettus Research. *USRP Hardware Driver and USRP Manual*. <http://files.ettus.com/manual/>.
- [17] ELEC University of Victoria. *Signals and Modulation*. http://www.ece.uvic.ca/~elec350/lab_manual/data/35015-IQ-AM-SSB-FM-PSK16.pdf. 2015.

AM modulation in LabVIEW

The screenshot displays the NI-Matlab-Simulink interface for a software-defined radio (SDR) system. The top panel, titled 'Waveform Graph', shows the configuration for the transmitted waveform. It includes a 'tone frequency' of 0.3, a 'tone amplitude' of 0.01, and a 'phase' of 0. A 'Waveform' block is connected to a 'Combine (and Q into complex samples)' block. The bottom panel shows a 'CDR' block with a 'status' indicator. A 'Stop' button is visible. The interface includes various control elements like 'device names', 'enabled channel', 'waveform size', 'Q rate', 'carrier frequency', 'gain', and 'active antenna'.

Figure A.1: Single tone AM modulator with USRP blocks within LabVIEW.

A.2 Audio file modulation with USRP

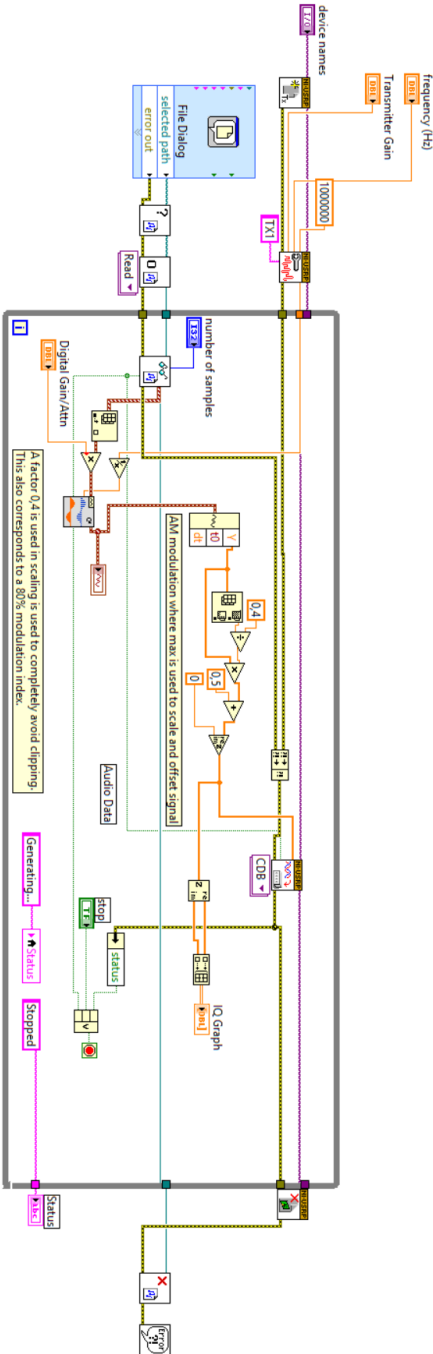


Figure A.2: AM audio-file modulator with USRP blocks within LabVIEW.

A.3 Audio and spectrum demodulator with USRP

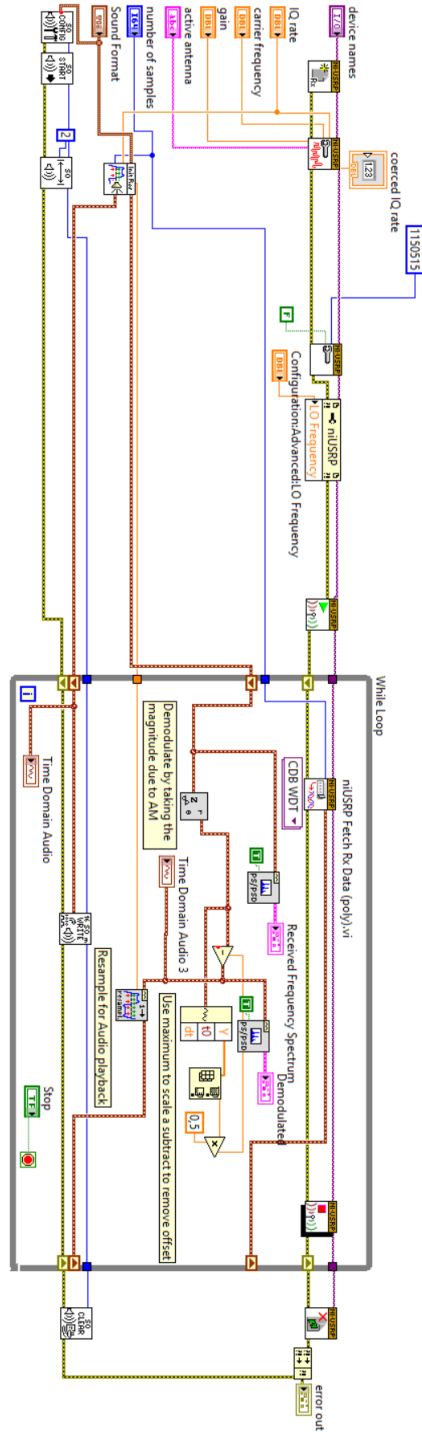


Figure A.3: AM demodulator with USRP blocks within LabVIEW.