NTNU – Trondheim
Norwegian University of
Science and Technology

NUTS: Ground station with GNU Radio and USRP

Karl David Vea

Master of Science in Electronics
Submission date: June 2015
Supervisor: Torbjørn Ekman, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problemstatement

A ground station is a very important part of a satellite communication system, in order to get data from a satellite in orbit. The use of Universal Software Radio Peripheral (USRP) minimizes costs for such systems and ads flexibility that can be used to simultaneously record data from several passing satellites for later decoding, or the possibility of testing different modulation schemes or link protocols.

In this work, the student should study the use of GNU Radio together with the USRP Software Defined Radio (SDR) as an alternative for a student satellite ground station. The student should describe the ground station and its place in the satellite mission and describe how GNU Radio works. The types of blocks needed to successfully demodulate a GMSK signal must be described and implementation must be discussed and analyzed. The NGHam link protocol should be implemented. Through simulations and evaluations, the student should discuss if and how GNURadio suits an university satellite project. Simulations on demodulation should consider how frequency synchronization can be achieved, as well as timing and BER analysis.

Abstract

One of the most important parts of a satellite system is the ground station. Without communication the usefulness of the satellite is small. Until now Ham radio has been the preferred choice as ground station, but due to recent years' rapid development of technology, Software Defined Radio (SDR) is now a contender.

This report will study if GNU Radio combined with USRP is a better alternative than Ham radio as ground station. What GNU Radio and USRP are and how they work is described. Modulation, demodulation and signal characteristics regarding Gaussian Minimum Shift Keying (GMSK) is analyzed. The importance of synchronization in a telecommunication receiver and different techniques to solve it is described and evaluated.

Simulations to test how much frequency offset, timing offset and AWGN noise the GMSK demodulation block withstands was performed. The results leading to the conclusion that fine tuning of frequency before the demodulator is probably not required.NGHam encoder protocol was implemented as a block in GNU Radio.

When a frequency band of interest is sampled by the USRP and shifted down to baseband, the signal of interest may not be centered around 0 Hz which is necessery before demodulation. A frequency analysis of a NGHam test signal was performed and showed 50 kHz frequency offset. Together with exploiting the syncword which the NGHam protocol use for frame synchronization, a syncword correlatorbank followed by a frequency translation FIR filter is proposed as a solution for the initial synchronization while the GMSK demodulation block handles the fine tuning.

GNU Radio is recommended as SDR platform for continuing work on ground station implementation.

Sammendrag

En av de viktigste delene av et satellitt system er bakkestasjonen. Uten kommunikasjon er nytteverdien til satellitten lav. Frem til nå har Ham radio vært det foretrukne valget for bakkestasjon utstyr, men seinere års teknologi utvikling har ført til at Software Defined Radio er nå en utfordrer.

Denne rapporten vil studere om GNU Radio kombinert med USRP er et bedre alternativ som bakkestasjon. Hva GNU Radio og USRP er og hvordan de fungerer er beskrevet. Både modulasjon, demodulasjon og signal karakteristikkene til Gaussian Minimum Shift Keying er analysert. Viktigheten av synkronisering i en mottaker innen telekommunikasjon og ulike muligheter er analysert og evaluert. Simuleringer er utført for å teste hvor mye frekvens avvik, tids avvik og støy toleranse for GMSK demodulerings blokken i GNU Radio. Resultatene leder til en konklusjon hvor fin tuning av frekvens før demodulator blokken trolig er unødvendig da denne kan hente seg inn fra et avvik. NGHam enkoder protokol ble implementer som en blokk i GNU Radio.

Når et frekvens bånd av interesse blir samplet av en USRP, så blir det automatisk flyttet ned til baseband. Signalet man er ute etter trenger ikke nødvendigvis være sentrert i midten. En frekvens analyse av et NGHam test signal ble utført og viste et frekvensavvik på 50 kHz. Sammen med å utnytte at NGHam protokollen bruker ramme synkronisering, så er en synkord korrelatorbank etterfulgt av en frekvens translasjons FIR filter blokk foreslått som løsning på synkroniserings problemet. Siden GMSK demodulasjon blokken kan ta seg av finjusteringen. GNU Radio er anbefalt som SDR platform for videre bruk til bakkestasjonen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The NTNU Test Satellite (NUTS) is a student satellite project. For some period, Ham radio was the preferred choice of equipment in a ground station receiver for micro/pico satellites. Lately, SDR and USRP have become a popular alternative. This is substantiated by several satellite companies now employing this combination Section 2.6. This report aims to examine the possibilities of GNU Radio as the SDR platform together with USRP as ground station. GNU Radio and Universal Radio Peripheral (USRP) will be introduced and detailed explained in Section 2.5 and Section 2.4.

NUTS aims to implement the newly developed NGHam link protocol which uses GMSK modulation Section 3.2. Synchronization is the most problematic part of the ground station receiver design, this is analyzed and a possible solution is proposed in the Discussion Chapter 6. An overview of the most relevant blocks in GNU Radio is briefly summarized in Section 4.1. Simulations of the GMSK demodulation block and the NGHam implementation is found in the Methods Chapter 4, while the simulation result is in Chapter 5.

Chapter 2

System description

2.1 NUTS and CubeSats

The NTNU Test Satellite (NUTS) is a student satellite project. The satellite is developed by students of the Norwegian University of Science and Technology. The project requires a wide variety of different technological areas. The aim for NUTS is to develop and launch a double cube satellite in compliance with the CubeSat Program. This program standardized the design of picosatellites and managed to reduce both cost and development time, and made launches more accessible for such satellites [16].

2.1.1 Orbital effects on satellite performance

The CubeSat will be operating in low earth orbit (LEO), and it ranges from 160-2500km altitude [18]. This satellite will operate at aproximately 600 km above Earth and in a polar orbit. One of the biggest disadvantages about this orbit is that the satellite is not seen as a fixed position seen from a point on Earth like a geostationary satellite, instead it travels around the earth at a speed of aproximately 7 km/s, in a circular orbit and is only seen from the groundstation for about 8-10 minutes each time it passes. This means that a tracking antenna is required at the groundstation. Because of the lower altitude, LEO satellites require smaller antenna systems and needs less transmit power because of lower path loss [18].

2.2 Ham radio

Icom IC 9100 is a analog amateur radio transceiver used in the High Frequency (HF), Very High Frequency (VHF) and Ultra High Frequency(UHF) bands. Using Hamlib API protocol. This is based on sound and is therefore more complicated and less flexible with regards to different modes. Because this is soundbased, the bandwidth is restricted to only 25 kHz compared to the USRP2

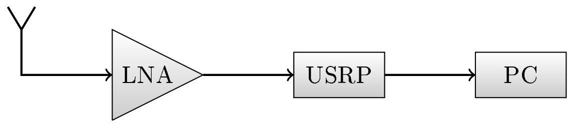


Figure 2.1: Block diagram of ground station receiver

with 40 MHz bandwidth. Amateur radio is bandwidth restricted to 18 kHz in the VHF band and 30 kHz in the UHF band [8]. By using SDR, it is a lot easier to make improvements, change and tweak the equipment compared to the analog “black box” ICOM. With the USRP it is possible to sample a wider frequency band for later analysis. It is also more flexible with regards to less equipment requirements.

2.3 Base station

A short description of what the ground station will do and how it is built is presented in this section. The satellite have the possibility to transmit data with 1200, 4800 and 9600 symbols/s using a 2 level GMSK-modulation scheme. The UHF band will be used by the main radio for transmission, and the VHF band for receiving and backup radio in default mode. The ground station should be able to log and display received data. The Yagi-Uda antenna system is connected to a Low Noise Amplifier (LNA), going in to USRP2 2.1. The transmitter system is the reversed version of the receiver except the LNA is switched to a Power Amplifier (PA).

2.4 Universal Software Radio Peripheral (USRP)

Universal Software Radio Peripheral (USRP) is an inexpensive hardware platform for using software defined radio (SDR). The field programmable gate array (FPGA) in the USRP handles the high speed analog to digital converting (ADC) and digital to analog converting (DAC), up/down converting from baseband and filtering while the host computer does the low speed operations.

The motherboard is the part of the USRP that converts the signal from analog to digital. The daughterboard, on the other hand, is moving the signal up or down in frequency. In total, the USRP can be used in the frequency range from DC to 6 GHz. This is achieved by using a daughterboard that supports the frequency range of interest. The daughterboard used in the USRP2 in the CubeSat program, is the WBX board. This is a transceiver with 40 MHz

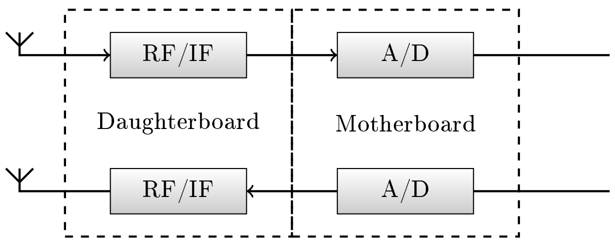


Figure 2.2: USRP

Figure 2.3: USRP2

bandwidth, supporting frequencies from 50 MHz to 2.2 GHz. It provides an output power level of up to 100 mW and a noisefigure of 5 dB [14].

The USRP2 has a fixed samplerate of 100 Mbps and then decimates the signal down to the frequency set in the USRP hardware driver (UHD) source. The signals streamed to and from the USRP have to be integer fractions of the clock rates, which is 100 Msps/N for the USRP2. For most applications, this involves resampling on the host as well. Combined with a minimum decimation of 4, the USRP2 has 25 MHz of instantaneous RF bandwidth. Decimation of the signal is carried out by using cascaded integrator–comb (CIC) filters. This is a class of FIR filters which only use addition and subtraction, and this makes them more economical than decimators who use multiplication [13].

Connection to the computer is handled by a Gigabit Ethernet cable, which entails no need for extra driver installation. Only need to set at static IP address which is usually 192.168.10.2. But it is important that both the USRP and Gnuradio has the same FPGA driver version for the UHD sinc/source or else it will not work.

2.5 GNU Radio

GNU Radio is a very popular open source project aimed to make SDR radio available to anyone. Equipped with GNU Radio and a USRP, or other simple equipment as a DVB-T dongle, it is possible to get started after just a few minutes. Even though it mainly supports Linux distros, there are Windows ports available. The designing process in GNU Radio can either be graphical or text based programming. While the graphic approach is a separate program called GNU Radio Companion (GRC), the latter can be programmed using your favorite text editor. The programming language is python, and this is used to set up the flow graph, connecting the the different blocks together. This is because python is considered to be a slow language, therefore most of the work is done in C++. It is also possible to make your own blocks in C++, thus providing almost unlimited possibilities. Another benefit is that the GNU Radio community is large and the response time on the online forums are short. One problem might be the program flow, as it seems to be difficult to monitor and control. There is also a pretty steep learning curve for beginners. As for setting up the system, it is done in a fairly simple manner by using apt-get install command in the linux terminal. A drawback of this option is that it is not the latest version that is installed and dependency problems might appear if out-of-three (OOT) modules are to be installed and used. An alternative solution is to use Python Build Overlay Managed Bundle System (PyBOMBS). This is a relatively new install management system that manages all the different dependencies when using OOT projects. By using PyBOMBS, an app store that is fairly easy to use becomes available [4].

2.5.1 Python and C++ programming

Out of tree modules: This is a block that is created outside the GNU Radio source tree. The script called "gr_modtool" creates the different the most important one includes the implementation file, header file and test code. A default constructor and destructor is created. It is marked in the code by <++> where one is supposed to change the code. There are four different blocktypes based on the relationship between input and output:

- Decimator blocks N:1
- Interpolator blocks 1:M
- Synchronous blocks 1:1
- General blocks N:M

In the implementation file there is a forecast() function, which tells the scheduler how many input items is needed to produce each output item. This is straightforward in the first three block types, but can be more tricky in blocks where the input/output relationship is not fixed.

In the general_work() function, the signal processing is done performed. Hence the implementation of the digital signal processing (DSP) functions in the block is put there.

To be able to send metadata along with the stream of bits, bytes, floats et cetera, stream tags is the solution. This is an isosynchronous data stream that runs parallel to the main data stream [3]. A stream tag is generated in the work function of a block and from there on flows downstream with a particular sample until it reaches a sink or is forced to stop propagating by another block.

Block testing When creating a new block it is important to make sure it works properly before installing it for use in GRC. To test this, the gr_modtool script makes a Quality Assurance (QA) code in python for the new block [4]. The QA code tests if the output of the block is as expected. This is done by configuring the QA python file with creating input data, expected output data, and then compare the actual output with the expected output. Python has several assertion tests, depending on how the block works it could be beneficial to use "almostequal" instead of for instance "equal" .

2.5.2 Program flow

GR top block is the main block where all other blocks are connected under, it control the running of the flowgraph. It also initiate the flowgraph.

GR block, many different types depending on functionality. All blocks have both one (or several) input and output stream, except sources and sinks. All flowgraphs require at least one source (producer of data) and one sink (consumer of data). Connections between blocks are made by the GR buffer, this is where input data and output data between blocks are stored.

The scheduler allocates computer resources. Starting at the source and looping around the flowgraph, checking if there is data available at the ouput buffer and if input buffer has sufficient data for each block by calling each block's executor, if true, the block is scheduled for processing.

Sample Rate (Hardware): There should allways be one rate control block within a flowgraph. This could be UHD sink/source, sound card or the throttle block. In general, it is important that is only one, if multiple control rate blocks is present, they will eventually cause overflows/underruns. Because they will become usynchronized.

2.6 GNU Radio and USRP for satellite communication

Skybox Imaging is a company founded in 2009. They provide commercial high-resolution Earth observation satellite imagery, high definition video and analytics services. Uses USRP and GNU Radio for satellite communication, using DBPSK modulation and USRP with WBX daughterboard, their satellites are based on the same CubeSat concept as NUTS [10]. Plans to have a fleet of 24 satellites with lifespan of 4 years. They were acquired by Google in 2014.

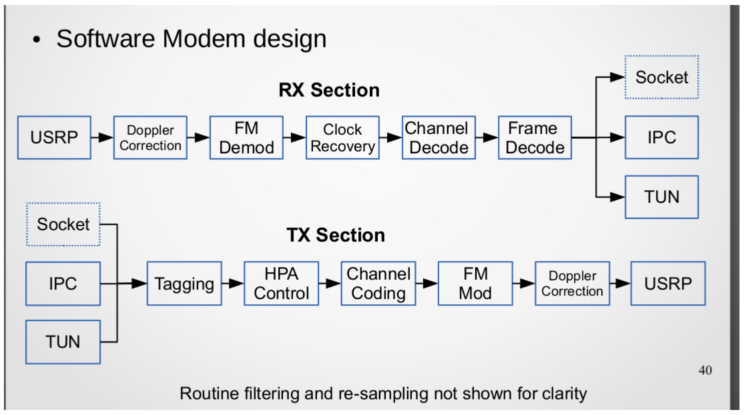


Figure 2.4: SpaceQuest ground station design[11]

SpaceQuest, Ltd Employs a fleet of LEO microsatellites used by the Automatic Identification System (AIS). This system monitors and track ships and large vessels by using the satellites as transceivers. Signals is transfered from the ships to the ground station via the satellites. The VHF band is the primary band, transmitting 9600 symbols/s [11]. GNU Radio and USRP is used as ground station and a block diagram shown in Figure 2.4.

Chapter 3

Theory

3.1 Modulation

Minimum Shift Keying (MSK) This is a special case of continous phase frequency shift keying (CPFSK). The peak frequency deviation equals 1/4 bit rate. This is a FSK modulation with index 0.5, which means the minimum frequency spacing that allows two signals to be orthogonal. Some of the important features include, constant envelope, spectral efficiency and good Bit Error Rate-performance. It is also capable of self-synchronizing and does not produce either Inter-Symbol-Interference (ISI) or Adjacent Channel Interference (ACI) [20].

3.2 Gaussian Minimum Shift Keying (GMSK)

This is a binary modulation scheme and can be thought of as binary frequency shift keying (BFSK). It is based on minimum shift keying (MSK) and both are continuous-phase frequency-shift keying (CPFSK). This means that there are no sharp phase discontinuity in time domain, which would lead to broad frequency spectrum.

GMSK is very popular in wireless communication where bandwidth constrictions and low power consumption is vital. A rectangular bitstream is passed through a Gaussian pulse shaping filter before modulation[1]. This yields lower sidelobe levels of the powerspectrum, lower than MSK and makes GMSK very spectral efficient, see Table 3.1. This GMSK filter can be defined from the product of B, its 3dB baseband bandwidth and $\mathrm{T_b}$, the baseband symbol duration ($\mathrm{BT_b}$). For instance, GMSK with a $\mathrm{BT_b}$=0.5 have a sidelobe level 30dB lower than the mainlobe, while MSK only has 20dB lower. A smaller $\mathrm{BT_b}$ value leads to a more compact spectrum at the cost of higher ISI. GSM for instance uses

[1]Gaussian in GMSK refers to the impulse response of the pulse shaping filter the rectangular pulsetrain is passed to, not the resulting frequency response. Not MSK that is gaussian filtered, because that would destroy the constant envelope.

Table 3.1: Occupied RF bandwidth as a fraction of R_b [20]

BT	90%	99%	99.9%	99.99%
0.2 GMSK	0.52	0.79	0.99	1.22
0.25 GMSK	0.57	0.86	1.09	1.37
0.5 GMSK	0.69	1.04	1.33	2.08
MSK	0.78	1.20	2.76	6.00

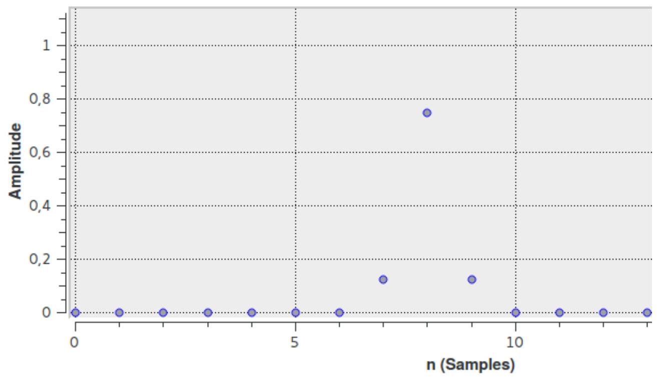


Figure 3.1: Gaussian filter response BT=0.5, 2 samp/bit

BT_b= 0.3, while the satellite radio in the CubeSat is using BT_b= 0.5. GMSK with BT = infinity equals MSK.

After the Gaussian filter, one symbol occupies now several bit periods. This is what introduces ISI in the receiver and degrades the MSK performance when coherent symbol-by-symbol detection is used. This is a tradeoff between spectral efficiency and bit error rate (BER) due to ISI. However, because this is a constant envelope modulation, it also has(property) very good power efficiency because non-linear C-class amplifiers can be used in saturation mode. This is both power efficient and cheaper than linear amplifiers.

This is used in mobile communication like GSM, and satellite communication because of the aforementioned excellent spectral and power efficiency.

For BT_b=0.3, the bits are spread over three periods, this means that ISI only comes from the adjacent bit periods before and after. For BT = 0.5 the bits are spread over two periods 3.1. The optimal BT_b value with regards to ISI is shown to be 0.5887, Tab [20]. This value is only 0.14 dB away from the case of no ISI.

3.2.1 GMSK generation

There are two main ways of generating GMSK modulated signals. The simple and intuitive way is to pass a non-return-to-zero (NRZ) bitstream to a Gaussian low pass filter and then a voltage controlled ocillator (VCO) with modulation

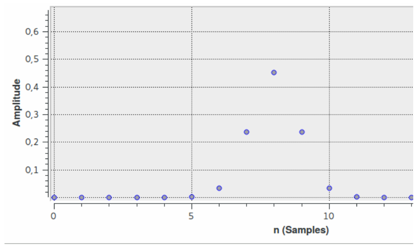


Figure 3.2: Gaussian filter response BT=0.3, 2 samp/bit

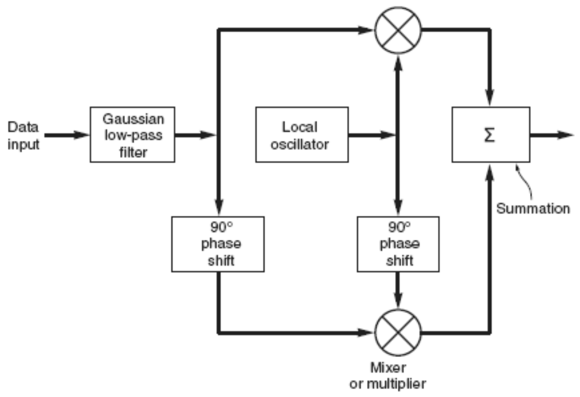


Figure 3.3: I-Q quadrature modulation block diagram

index 0.5 like the Fig 3.4. The other solution is using a I-Q modulator as illustrated in Fig 3.3

3.2.2 Gaussian pulse-shaping filter

This is a non-Nyquist technique for pulse shaping. This differs from Nyquist filters such as raised cosine (RC) and root raised cosine (RRC) by not having zero-crossings and a smooth transfer function. The filter is spectral efficient, but induces ISI. This is a trade off between adjacent channel interference (ACI) and ISI. The relationship between the 3dB bandwidth of the gaussian filter and

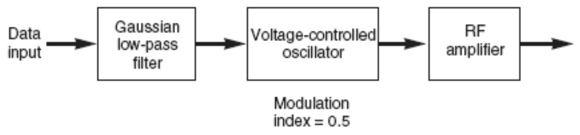


Figure 3.4: Simple GMSK modulation using Gaussian filter and VCO

$\mathrm{BT_b}$is given by:

$$BT_b = \frac{f_{-3dB}}{BITRATE} \tag{3.1}$$

This gives a cutoff frequency of 4800 Hz for $\mathrm{BT_b}$= 0.5 and 9.6 kbps.

3.2.3 Detection (Demodulation)

GMSK can be detected either non-coherently like simple FSK or coherently just like MSK. Coherent demodulation requires knowledge of carrier phase. A highly effective but suboptimal solution is to just sample the output of an FM demodulator[20]. Lower Eb/No required for same BER with coherent detection versus differential detection. The required training sequence for frequency and symbol timing synchronization is usually called preamble bits. They are packed right before a unique word which is for frame synchronization purposes. Both frequency and symbol timing synchronization have to be accomplished before the ending of a preamble.

GMSK can also be viewed as OQPSK and demodulated as such. The difference between OQPSK and regular QPSK is a time delay between I and Q channels, so by removing this delay in the receiver one can demodulate this as QPSK.

3.2.3.1 Coherent

A quadrature demodulation is dependent on carrier freqency and phase recovery and frame synchronization. The optimal performance in ISI environment can be achieved by using maximum likelihood sequence estimation (MLSE) with a Viterbi trellis for demodulation, but this is more complex [6].OQPSK demodulation could be used by delaying the Q channel by a 1/2 symbol period. With raised cosine nyquist filters. They have good frequency selection and do not induce ISI [2].

3.2.3.2 Non coherent

There are two types of non-coherent demodulation for GMSK. Non-limiter/discriminator and differential detection. Differential detection is only dependent on the relative phase difference between consecutive symbols.

In differential detection, as a substitute of using absolute carrier phase, information is encoded using the carrier phase differences. At the receiver, it is then recovered by attaining the difference in phase of received signal at current time and at past time, usually at multiples of the symbol period. Because only the phase difference of the carrier is used in the transmission and detection, the requirement for carrier recovery is eliminated in non-coherent receiver and it is simpler to implement.

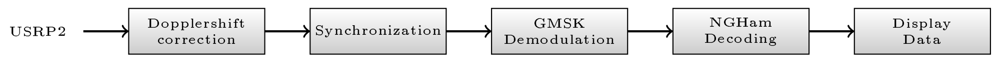


Figure 3.5: Block diagram of a receiver

3.3 Receiver

Implementation of GMSK receiver is more complex than linear receivers, as GMSK signal is non-linear in nature, which is why a matched filter will not work in the same way as for instance in QPSK. Instead Murota and Hirade use a Gaussian band pass filter at the receiver [19]. Because this is burst transmission, a preamble and syncword for each packet is required. This is implemented in the NGHam protocol. The USRP does AD converting of the signal and decimation. Sample rate requirements minimum 2 x bandwidth to avoid aliasing and this is the minimum sample rate for lossless communication. Demodulation requires baseband signal. A block diagram Figure 3.5.

3.3.1 Synchronization

There are three important steps in synchronization of a radio signal for

1. Carrier recovery
2. Symbol timing recovery
3. Frame synchronization

For coherent detection in a digital communication system, the receiver and transmitter need to be synchronous. This is because it needs to know when the symbol starts and ends for each symbol to be able to sample at the right time. The estimation of carrier phase and frequency is called carrier recovery and symbol synchronization. There are two possible ways, data-aided or non-data-aided synchronization.

Data-aided synchronization is when a preamble that is known to the receiver, is transmitted with each frame of data. This contains information about the carrier and symbol timing. This approach is often used in wireless and satellite communications to minimize the required synchronization time [17]. The downside of data-aided synchcronization is reduced effective datarate and power efficiency because a fraction of the total transmit power is used on the preamble. The preamble is usually for frequency and symboltiming sync, while the syncword is for frame synchronization [17].

3.3.1.1 Carrier recovery

In a wireless communication system, the transmitter and receiver use independent oscillators, which are not 100% accurate. Because the satellite is a moving object, the signal will experience doppler shift which leads to a shift in frequency and phase. To get the correct information data out of the receiver it is important to correct for this. There are many ways to do this, in this thesis only a few is considered [12].

Multiply-filter-divide In this method of non-data-aided carrier recovery a non-linear operation is applied to the modulated signal to create harmonics of the carrier frequency with the modulation detached. The carrier harmonic is then filtered by band pass filter and frequency divide d to recover the carrier frequency [12]. The example of open - loop carrier recovery is Multiply-filter-divide , which is favored in burst transactions since the acquisition time is typically shorter than for closed - loop synchronizers.

Costas loop is a phase-locked loop based circuit which is used for carrier phase recovery from suppressed-carrier modulation signals, such as from double-sideband suppressed carrier signals. This translates to double the sensitivity and also makes the Costas loop uniquely suited for tracking doppler-shifted carriers.

Carrier frequency and phase recovery as well as demodulation can be established using a Costas loop of the suitable order. A Costas loop which is similar to a PLL that uses coherent quadrature signals to measure phase error which is used to discipline the loop's oscillator at the receiver [12].

de Buda In his method, the reference carrier is recovered by dividing the sum of the two discrete frequencies contained in the frequency doubler output by four, and the bit clock is directly recovered by their difference [20]. This method is similar to Costas loop and both are often used in GMSK demodulation.

3.3.1.2 Symbol timing recovery

There are several timing recovery algorithms out there. This is a few of the most popular.

Gardner timing recovery algorithm requires two samples per symbol and knowledge of the previous symbol timing to estimate the timing error for current symbol [12].

Late-early timing recovery is one of the simplest methods and is widely used in digital communications. This method takes three samples spaced by Ts (sampling duration), with the duration T . The early samples are sampled at nT - Ts and late samples are sampled at nT + Ts . The difference between the late and early samples is the timing error. Based on the timing error , the next symbol timing sampling time is either advanced or delayed until the timing error is reduced [12].

Mueller-Muller timing recovery The Mueller-Muller timing recovery algorithm requires only one sample per symbol and knowledge of the previous symbol to estimate the timing error [12]. Timing error calculated for either I or Q as follows: Where is the decision symbol of the sample? There are 3 cases: first case, if error equals 0, no adjustment needed. If e<0, timing is advanced. If e>0, timing is delayed .

3.3.1.3 Frame synchronization

In wireless communication, the frame synchronization is defined as the process in which , the incoming frame alignment signals i.e., a distinctive bit sequences or sync words are detected while the stream of framed data is being received. Thereby, passing the data bits within the frame to be extracted for decoding or retransmission. A syncword is used to convey end of header information and start of data.

Syncword detector is a syncword correlator, it operates by correlating the incoming signal with a known syncword. If the value of the correlation is above a given threshold parameter, a frame is detected. The start of datapacket is then.

3.3.2 Automatic Gain Control

This a closed-loop regulating circuit, the purpose of which is to provide a controlled signal amplitude at its output, despite variation of the amplitude in the input signal. The average or peak output signal level is used to dynamically adjust the input-to-output gain to a suitable value, enabling the circuit to work satisfactorily with a greater range of input signal levels [1].

3.4 NGHam link protocol

The NGHam protocol was developed by a member of the NUTS volunteer group. This protocol is meant to be an alternative of the rather inefficient AX.25. Because it was developed for the Owl VHF radio it is already implemented in the satellite radio.

As seen in Figure 3.6, frames can have 7 possible lengths. This will reduce redundant padding bits if less information is sent. The size tag says which of the seven possible lengths the received frame has. Note that this tag is not coded with error correction/detection, but constructed so that the hamming distance between all code words are maximized. Thus, it allows the decoder to tell which length is sent even with quite high amounts of bit errors [7].

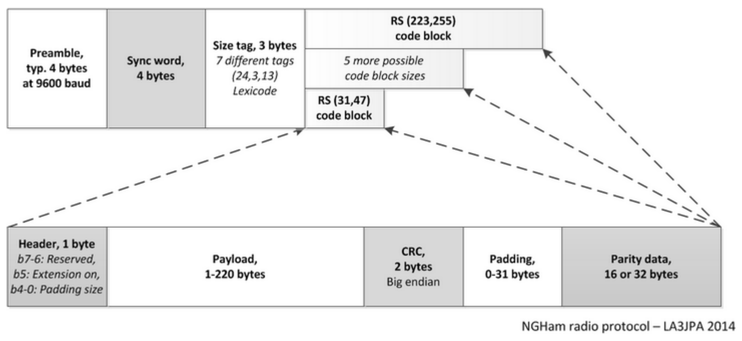


Figure 3.6: NGHam block diagram

When it comes to redundancy, this protocol excels, as it keeps its header information to a minimum. Even though this protocol does not include the convolutional code, this does not seem to be necessary. One big advantage of this protocol is that it has seven different package sizes. This will keep the number of redundant dummy data to a minimum when the number of information bits is less than 223. The Reed-Solomon(RS) code is adjusted accordingly when the amount of bits are changed. All these properties are in general an advantage.

Cyclic redundancy check (CRC): This is an error detecting code used to detect accidental changes to raw data. It is easy to implement in binary hardware and particularly good at detecting common errors caused by noise in transmission channels. This is a part of the NGHam encoder as seen in Figure 3.6.

3.4.1 Reed-Solomon Code

The Reed-Solomon (RS) code is a block-based forward error correcting (FEC) code. Block length of RS codes is $n = 2^m - 1$ and the number of parity bits needed to correct e error is $k = n - 2e$. It is often used in combination with convolutional coding due to its burst error correction capabilities, but in this protocol it is used alone to save data rate and complexity in the receiver. The minimum distance of RS code is the largest of all linear codes and is given by $d_{min} = 2e + 1$ [20, 400].

3.4.2 Scrambler

A scrambler is commonly used in satellite and radio relay communications. This device manipulates the data stream by whitening the data before modulation. It can be placed either before or after the FEC. This is usually performed by linear feedback shift registers (LFSR). The process is then reversed on the receiver side by a descrambler [9].

The channel Basically the channel can be closely modeled as an additive white Gaussian noise (AWGN) channel [17, s.536]. Detailed description channel impairments [18]. The channel capacity is describe by the Shannon Hartley theorem [15]:

$$C = Blog_2[1 + (S/N)] \tag{3.2}$$

C is the channel capacity in bps, B is the channel bandwidth in Hz and S/N is the signal-to-noise receiver input. But calculating the link budget for the satellite channel is out of scope of this report.

Doppler shift Doppler shift is a frequency shift caused by relative motion between the transmitter and receiver [15]. [5]

$$(\frac{f_R - f_T}{f_T}) = (\frac{\triangle f}{f_T}) = \frac{v_T}{v_P} \tag{3.3}$$

where $\mathrm{v_T}$ is the component of the satellite transmitter velocity vector directed towards the ground station and $\mathrm{v_P}$ is the phase velocity of light in free space (3 × 10 8 m/s). Instead of calculating this by hand, there are several applications available which predict satellite orbit and tracks satellites. Gpredict is one of them and is the one used with the previous Ham radio ground station setup [5].

Chapter 4

Methods

4.1 GNU Radio blocks

This is a short description of the most relevant blocks available in GNU Radio.

Correlate and sync This block is designed to search for a preamble by correlation and uses the results of the correlation to get a time and phase offset estimate. These estimates are passed downstream as stream tags for use by follow-on synchronization blocks. The preamble is provided as a set of symbols along with a baseband matched filter which we use to create the filtered and upsampled symbol that we will receive over-the-air[3]. The phase_est tag is used to adjust the phase estimation of any downstream synchronization blocks and is currently used by the costas_loop_cc block. The time_est tag is used to adjust the sampling timing estimation of any downstream synchronization blocks and is currently used by the pfb_clock_sync_ccf block.

Costas loop The Costas loop can have two output streams: stream 1 is the baseband I and Q; stream 2 is the normalized frequency of the loop[3]. The Costas loop locks to the center frequency of a signal and downconverts it to baseband. Uses coherent quadrature signals to measure phase error. This is then used to change the oscillator.

Clock recovery MM implements the Mueller and Muller (M&M) discrete-time error-tracking synchronizer. A short explanation is found in 3.3.1.2 and this is used as a part of the GMSK demodulator.

Frequency Xlating FIIR filter frequency translating, channel selection and decimation filter block which can shift a frequency band specified by the center frequency and bandwidth up or down in frequency, usually used for moving a signal down to baseband before demodulation [3]. After shifting the signal down, a bandpass and low pass filter with the desired parameters remove unwanted

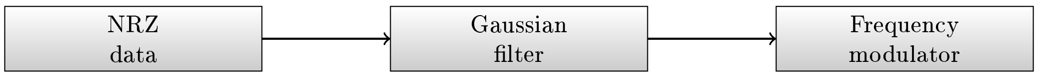


Figure 4.1: GNU Radio GMSK modulator block

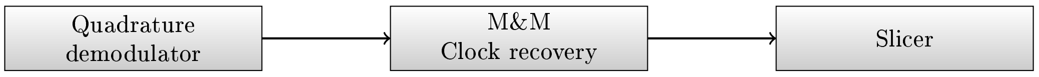


Figure 4.2: GNU Radio GMSK demodulation block

noise. This can be done in GNU Radio using firdes.low_pass(gain,sample rate, cutoff frequency,transition width). After multiplication, the result is lowpass filtered to obtain only the baseband signal. Decimation is performed to minimize the number of samples needed for processing in the later blocks. Parameters:

- Decimation
- Taps
- Center frequency
- Sample rate

GMSK modulator Turn datastream into NRZ data Figure 4.1. Form Gaussian filter. Generate Gaussian response (Needs to be convolved with window below), and then FM modulation [3].

GMSK demod The GMSK demod block is built up of three parts [3]. First the FM demodulation of the incoming signal. Then the clock recovery MM block tracks the symbol clock and resamples as needed by using a polyphase filterbank. The stream of floats is then sliced at 0, outputting 1 bit (the LSB of the output byte) per sample 4.2.

Fll band edge The frequency lock loop derives a band-edge filter that covers the upper and lower bandwidths of a digitally-modulated signal[3]. The bandwidth range is determined by the excess bandwidth (e.g., rolloff factor) of the modulated signal. The placement in frequency of the band-edges is determined by the oversampling ratio (number of samples per symbol) and the excess bandwidth. The size of the filters should be fairly large so as to average over a number of symbols.

Filter design tool The Filter Design Tool is a GUI that allows you to interactively design different types of filters [3]. This is a python script that can be opened from inside the GRC GUI. It is a GUI with options for simulating and plotting different characteristics of filters and make it possible to visualize and observe the effect of altering parameters have on the filterdesign. Afterwords coefficients can be imported directly to the GRC.For example, when placing an interpolating or decimating FIR filter block into your GRC flowgraph the filter taps could be found using the filter coefficients output by the design tool.

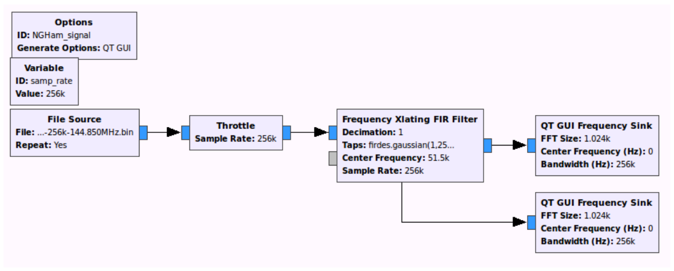
Channel Model The basic channel_model block is essentially an additive white Gaussian noise (AWGN) channel with a few extra additions. This block simulates AWGN as well as frequency and timing offsets between the transmit and receiver and a simple static multipath environment.

- noise_voltage: The AWGN noise level as a voltage
- frequency_offset: The normalized frequency offset. 0 is no offset; 0.5 equals half the symbol rate.
- epsilon: The sample timing offset to emulate the different rates between the sample clocks of the transmitter and receiver. 1.0 is no difference.
- taps: Taps of a FIR filter to emulate a multipath delay profile.

Error rate takes two input streams and calculates how often errors occur and return a value $0 \leq x \leq 1$ percent. The error rate block awaits for data on both input channels before evaluating. This block is not a part of the source tree, it is downloaded from the app store.

4.2 NGHam signal

A test sample from the VHF radio was obtained indoor with the USRP2 and GNU Radio. An UHD source with sample rate = 256 kHz, center frequency = 144.850 MHz and sent to a file sink for later use 4.5. To inspect the frequency content of the sample a frequency sink was used with the obtained signal as source Fig 4.3. It is clear from the upper graph in Fig 5.4 that the transmitted signal is centered about 50 kHz. Before demodulation, the signal must be shifted down to baseband and centered around zero Hz. By using the frequency xlating FIR filter block, the signal can be shifted down and filter out noise. A gaussian filter was implemented using the embedded firdes.gaussian(gain, samples per symbol (sps), BT, number of taps) function in GNU Radio with the following parameters (1, 256000/9600, 0.5, 30). Because 256 kHz was the sample rate and symbol rate of the NUTS radio is 9600 bits/s. Used filter design tool to see the magnitude response for different number of taps, 30 is the default number and was the best candidate due to the



Options
ID: NGHam_signal
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 256k

File Source
File: ...-256k-144.850MHz.bin
Repeat: Yes

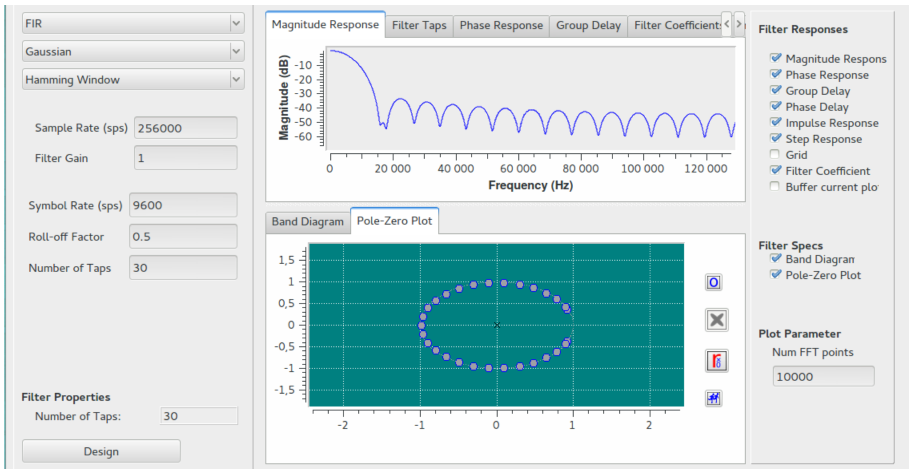
Throttle
Sample Rate: 256k

Frequency Xlating FIR Filter
Decimation: 1
Taps: firdes.gaussian(1,25...
Center Frequency: 51.5k
Sample Rate: 256k

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 256k

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 256k

Figure 4.3: Flowgraph of Frequency Xlating FIR filter



FIR
Gaussian
Hamming Window

Sample Rate (sps) 256000
Filter Gain 1

Symbol Rate (sps) 9600
Roll-off Factor 0.5
Number of Taps 30

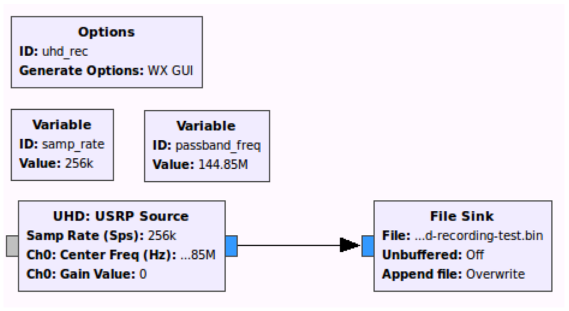
Filter Properties
Number of Taps: 30

Filter Responses
✓ Magnitude Respons
✓ Phase Response
✓ Group Delay
✓ Phase Delay
✓ Impulse Response
✓ Step Response
☐ Grid
✓ Filter Coefficient
☐ Buffer current plo

Filter Specs
✓ Band Diagran
✓ Pole-Zero Plot

Plot Parameter
Num FFT points 10000

Figure 4.4: Gaussian filter with filter design tool



Options
ID: uhd_rec
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 256k

Variable
ID: passband_freq
Value: 144.85M

UHD: USRP Source
Samp Rate (Sps): 256k
Ch0: Center Freq (Hz): ...85M
Ch0: Gain Value: 0

File Sink
File: ...d-recording-test.bin
Unbuffered: Off
Append file: Overwrite

Figure 4.5: Satellite radio indoor test transmission

Local oscillator frequency offset: Oscillators have offset given in parts per million (ppm). This is dependent of quality, temperature etc. For example, ± 20 ppm is a common value. That means that the max frequency offset would be $\pm 145x20 = 2900$ Hz for the amateur radio VHF (144146 MHz) and $\pm 435 * 20 = 8700$ Hz for UHF band (432-438 MHz). And this is only on either the transmitter or receiver side. If both oscillators have the same offset but opposite sign, then it would double the total offset.

4.3 NGHam encoder block

Porting the NGHam encoder block that Jon Petter Skagmo has implemented in C to GNU Radio. To create a new module gr_modtool was used as mentioned in 2.5.1. The general block type was chosen because the relationship between input and ouput is not fixed, hence this was the only alternative. Input and output signatures were set to char because the encoder operates on bytes. In the general_work() function, the main code was implemented. While predefined constants, arrays, scrambler code and cic code were placed in the header file. Some of the functions was modified to more compact syntax.

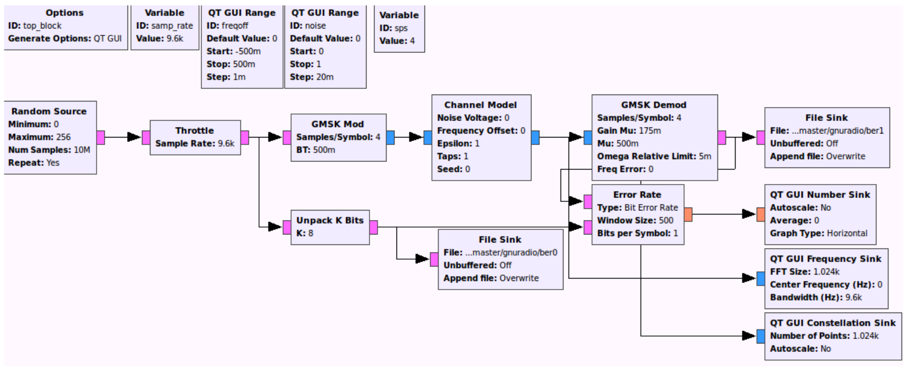
The RS encoder which is found in gnuradio/fec/api did not work, this due to the FEC API does not yet fully support it at this time. So instead a less elegant solution was implemented by including the RS encode implementation by Phil Karn [7], directly into the c++ header file. This runs now and output 58 bytes as expected when given 28 bytes as payload.

QA code The test was performed step by step to check that each part of the block was added correctly. Starting in the front with the preamble as seen in Figure 3.6 and moving backwards. Performed QA code testing for preamble, syncword, sizetag, cic, payload and the ccsds scrambler. The QA code can be seen in appendix.

4.4 GMSK demodulation block simulations

To decide what kind of synchronization is required, the GMSK demodulation block implemented in GNU Radio was tested against frequency offset, AWGN noise and timing offset.

BER test for GMSK demod block: A bytestream with value between 0 and 256 is generated by the Random Source block Fig 4.6, the throttle block is used to avoid the PCU to work 100 %. GMSK Modulation block takes packed bits as input and output complex float stream, while demodulation block input complex float and outputs unpacked bits with 1 bit per byte. The bytestream is sampled by 4 samples per symbol (sps) and BT is set to 0.5 as touched upon in Section to 3.2. To compare input versus output, because input is packed bits, they need to be the same type before entering the Error Rate block. This is done



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 9.6k

QT GUI Range
ID: freqoff
Default Value: 0
Start: -500m
Stop: 500m
Step: 1m

QT GUI Range
ID: noise
Default Value: 0
Start: 0
Stop: 1
Step: 20m

Variable
ID: sps
Value: 4

Random Source
Minimum: 0
Maximum: 256
Num Samples: 10M
Repeat: Yes

Throttle
Sample Rate: 9.6k

GMSK Mod
Samples/Symbol: 4
BT: 500m

Channel Model
Noise Voltage: 0
Frequency Offset: 0
Epsilon: 1
Taps: 1
Seed: 0

GMSK Demod
Samples/Symbol: 4
Gain Mu: 175m
Mu: 500m
Omega Relative Limit: 5m
Freq Error: 0

File Sink
File: ...master/gnuradio/ber1
Unbuffered: Off
Append file: Overwrite

Unpack K Bits
K: 8

File Sink
File: ...master/gnuradio/ber0
Unbuffered: Off
Append file: Overwrite

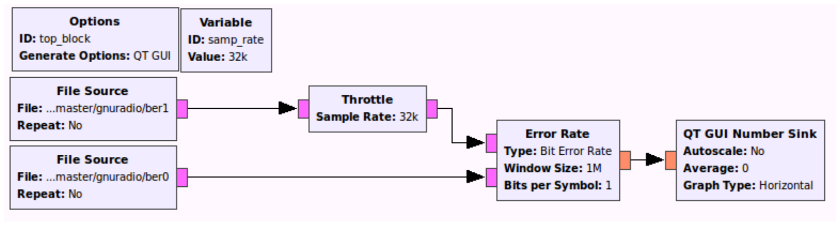
Error Rate
Type: Bit Error Rate
Window Size: 500
Bits per Symbol: 1

QT GUI Number Sink
Autoscale: No
Average: 0
Graph Type: Horizontal

QT GUI Frequency Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 9.6k

QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

Figure 4.6: Setup BER test



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

File Source
File: ...master/gnuradio/ber1
Repeat: No

File Source
File: ...master/gnuradio/ber0
Repeat: No

Throttle
Sample Rate: 32k

Error Rate
Type: Bit Error Rate
Window Size: 1M
Bits per Symbol: 1

QT GUI Number Sink
Autoscale: No
Average: 0
Graph Type: Horizontal

Figure 4.7: Error rate from file sink

by a Unpack K bits block, in this case K = 8. The result is then displayed by the QT GUI Number Sink. Using a constellation sink to see how the constellation changes with the impairments. QT GUI Range variable was used to enable adjustments of the channel parameters while the flowgraph is running.

Encountered a problem with the fact that 4 was the only possible number of sps that did not result in guaranteed 50 % error rate. Did not find any solution to this problem.

Measurements was performed by:

- Start flowgraph with freqoff = 0, increase offset gradually until errors occured.
- Timing offset= 1, increasing timing offset until error.
- AWGN noise voltage = 0, increasing until error.
- AWGN noise and freqoff together.

To check if delay due to due to for instance FIR filters , was the cause of 50 % error rate, the error rate was measured from file sources Figure 4.7. The error rate block seems to be waiting for the second before evaluating.

Chapter 5

Results

5.1 GMSK demodulation simulation in GRC.

Noise and frequency offset simulation of the GMSK demodulator. Graph over freqoffset with and without noise.

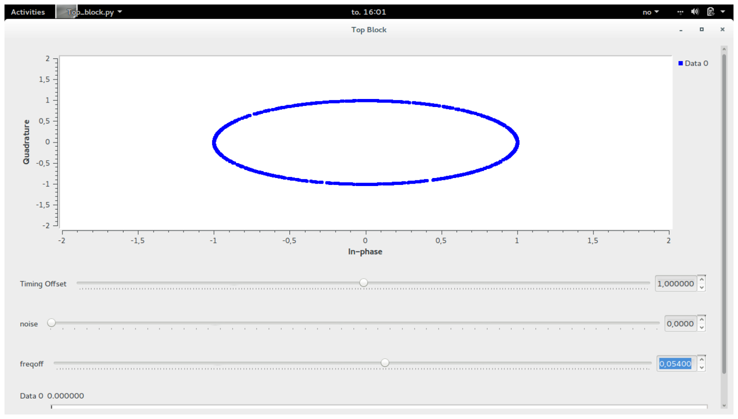
Results: Freqoff only tolerance limit = 0 0.054
Timing offset only limit= 1.014
Noise tolerance limit= 0.014
With 4 samples per symbol and a symbolrate = 9600 for the NUTS radio. Max frequency offset tolerance is in ideal : $4 * 9600 * 0.54 = 2074$ Hz.

5.2 NGHam sample

Plot of frequency spectrum before and after moving down to center Figure 5.4.

Table 5.1: Frequency limit depending on AWGN noise

AWGN noise[V]	0.03	0.05	0.07	0.09	0.13
Freqoff [relative Hz]	0.043	0.035	0.029	0.022	0.01



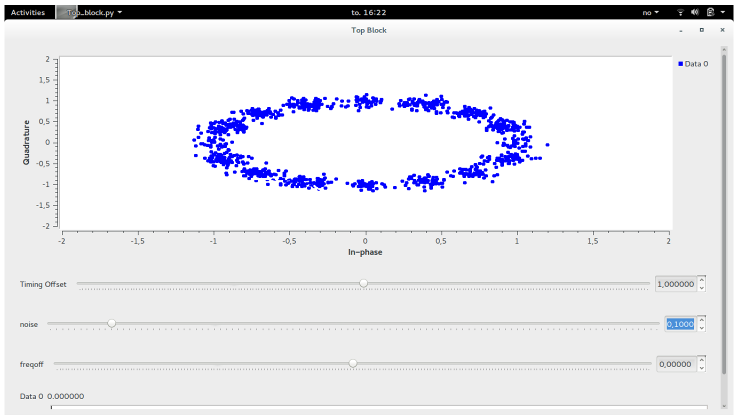
Timing Offset 1,000000

noise 0,0000

freqoff 0,05400

Data 0 0.000000

Figure 5.1: GMSK demod: frequency offset only



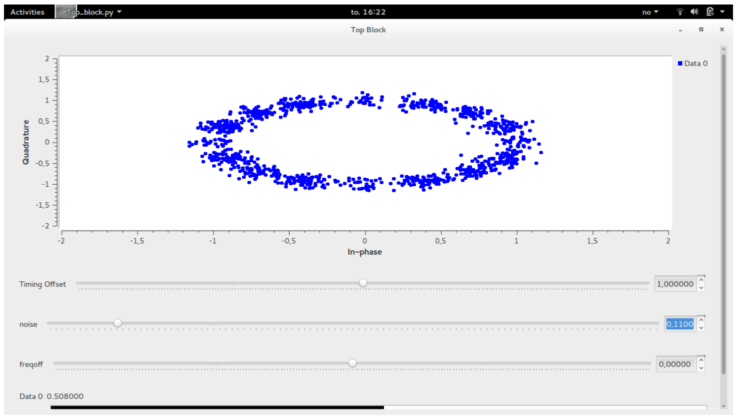
Timing Offset 1,000000

noise 0,1000

freqoff 0,00000

Data 0 0.000000

Figure 5.2: GMSK demod: with AWGN noise



Timing Offset 1,000000

noise 0,1100

freqoff 0,00000

Data 0 0.508000

Figure 5.3: Setup with AWGN noise, 50% error

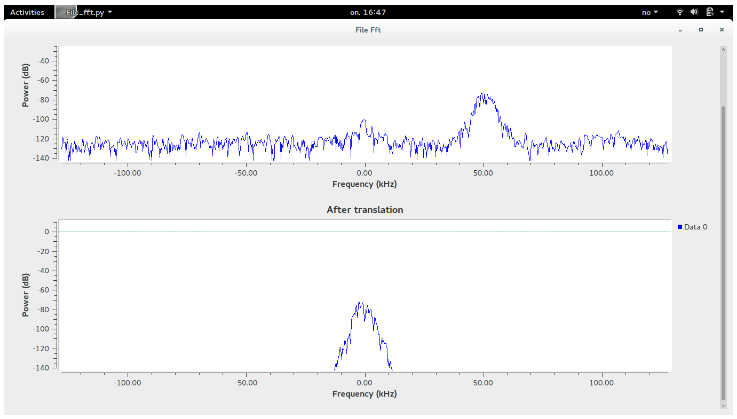


Figure 5.4: Frequency spectrum of NGHam sample before and after frequency translation

Chapter 6

Discussion

6.1 GNU Radio and USRP

The SDR platform GNU Radio offers a wide array of possibilities. Detailed explanation and documentation of individual blocks is not always as good as one could want. So if a certein block do not work as expected it can be difficult to find the problem. But often someone else have encountered the same problem so a search on the Internet could potentially be useful. The flowgraphs are easy to modify in python with simple code to add wanted functionality. It takes time to understand how everything works when implementing new blocks in C++, and there are several different files that need attention. By using gr_modtool together with online tutorials, the threshold to start implementing own simple blocks is low. QA code is a good and intuitive method for testing the implementation. By printing out the actual output and expected output and compare the two, this makes finding errors easier. It is important to use the same samp/sym in the different blocks in the same flowgraph. The NGHam sample in Figure 5.4 shows why the possibility to sample a much wider band than the 25 kHz the Ham radio is able to is sample is important.

6.2 Receiver

As explained in section 3.3, synchronization is the main problem with the receiver in wireless telecommunication. There are several possibilities for this, for carrier frequency and phase recovery one of them is to use the fll_band_edge block for the inital recovery and then costas loop afterwords for fine tuning. This is because the fll band edge block locks to a wider bandwidth compared to the Costas loop. A drawback of the fll_band_edge block is that it uses a lot of CPU, due to calculating two FIR filters.

Another way is to use the preamble or syncword correlation for initial frequency recovery and then the costas loop or de Buda carrier frequency recovery and the Mueller & Muller for timing recovery block. Considering that the GMSK

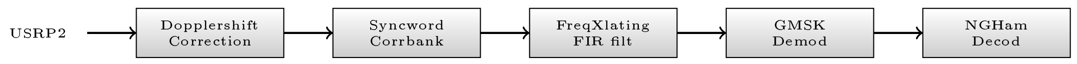


Figure 6.1: Blockdiagram proposed receiver with correlatorbank

demod block withstands up to 2 kHz frequency offset in ideal conditions Section 5.1. However, the frequency offset tolerance decreases for increasing AWGN noise, but this is expected. that the costas loop is a fine tuning loop, it would be unnecessary. Also the M&M block is already a part of the demodulator Fig4.2.

As Fig 5.4 shows, even though the USRP2 shifts the sampled band down to baseband, there is no guarantee that the information signal is centered in the middle when the band is 256 kHz wide as in the figure, since the NGHam protocol incorporates data-aided synchronization option, it is natural to exploit this. A solution would be to implement the syncword correlator as a correlator bank with equally spaced center frequency, then the center frequency where the highest correlation would be used as input to the following frequency xlation block Fig 6.1. Simultaneously a stream tag will mark the start of the data packet. When the signal reaches the demod block the stream tag enable demodulation. For this to work, the GMSK demod block implementation must be modified to read the stream tag.

- Use freq input port in Frequency Xlating FIR Filter block for setting new center frequency.
- Streamtag for marking the start of datapackets
- Correlatorbank with equally spaced correlator bands

In a worst case scenario where real time demodulation failed due to frequency offset or other things, there still is a backup solution with offline demod. Use a frequency sink to manually see where the signal is. Then frequency xlating block for bandpass filtering (remove unwanted noise) and shifting the signal down to baseband.

Dopplershift correction, import estimated center frequency after doppler shift from Gpredict[5] and also get estimations for antenna adjustment.

GMSK demodulator Alternative solution if prototype testing of demodulator block fails.

- OQPSK demodulation by delaying the Q-channel with 1/2 symbol period
- Viterbi GMSK demod implementation

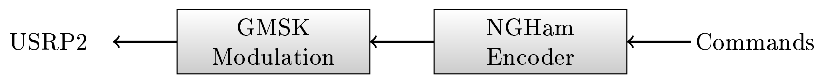


Figure 6.2: Transmitter

6.3 Transmitter

The ground station transmitter as illustrated in Figure 6.2 is straight forward with the chosen commands sent to the NGHam encoder as byte packets. The encoded bytes is then passed along to the GMSK modulation block before sent to the UHD sink for transmission.

Chapter 7

Conclusion

In satellite communication the ground station is a very important part, without a communication link the satellite becomes rather useless. GNU Radio together with a USRP is an powerful option to the Ham radio ground station.

Synchronization is the main problem of a communication receiver, due to doppler shift and the fact that locale oscillators will never be 100 % accurate. Frequency analysis of the NGHam signal sample illustrates the impact on frequency offset by the locale oscillators.

A ground station with GNU Radio and USRP provides a flexible, low cost and open source solution. Also a wide range of possibilities regarding implementations of functionality and the important option of sampling a wide frequency band is present.

NGHam protocol encoder was implemented in GNU Radio. Whil. A syncword correlator bank is proposed as the solution to the synchronization problem. This should be employed after Gpredict is used to predict the incoming doppler shift and set the center frequency in the UHD source.

GMSK demodulator block in GNU Radio is capable of handling up to a 0.055 nomalized frequency offset, translating to 2 kHz in this simulation with perfect conditions. By adding timing offset and AWGN noise this also simulated and shows the frequency offset toleration. A fine tuning frequency block, as for instance costas loop, may therefore be unnecessary. GNU Radio is recommended as SDR platform for continuing work on ground station implementation.

Future work would be to port the NGHam decoder to GNU Radio. Implementation of syncword correlator bank and add streamtag function to GMSK demod block. Setup doppler correction with Gpredict and test the system.

Bibliography

[1] Automatic gain control. https://en.wikipedia.org/wiki/Automatic_gain_control. [Online; accessed: june-2015].

[2] A brief examination of cqpsk for cpe phy modulation. http://www.ieee802.org/16/tg1/phy/contrib/802161pc-00_11.pdf. [Online; accessed: june-2015].

[3] Doxygen - GNU Radio Manual and C++ API Reference. https://gnuradio.org/doc/doxygen/index.html. [Online; accessed: june-2015].

[4] GNU Radio WIKI. http://gnuradio.org/redmine/projects/gnuradio/wiki. [Online; accessed: june-2015].

[5] Gpredict. http://gpredict.oz9aec.net/index.php. [Online; accessed: june-2015].

[6] A high-performance reduced-complexity gmsk demodulator. http://www.ittc.ku.edu/~prescott/kcp/HPRC-GMSK-Demod.pdf. [Online; accessed: june-2015].

[7] NGHam. https://github.com/skagmo/ngham. [Online; accessed: june-2015].

[8] Nrrl - Available frequency amateur radio. https://nrrl.no/tjenester/frekvens-og-lisens/bandplan. [Online; accessed: june-2015].

[9] Scrambler. https://en.wikipedia.org/?title=Scrambler. [Online; accessed: june-2015].

[10] Skybox Imaging - 2012 GNU Radio Conference September 25, 2012. http://static1.1.sqspcdn.com/static/f/679473/20514128/1349304449653/otomo-grc2012.pdf?token=mtUjRly3fo4Fx4tllkMW9StOizo%3D. [Online; accessed: june-2015].

[11] SpaceQuest ltd - 2014 GNURadio Conference September 17, 2014. http://static1.1.sqspcdn.com/static/f/679473/25461597/1411158933573/Sep17_08_CaJacob_SpaceQuest.pdf?token=ypF%2FNGZ2GRqjXgHfZKUlWFhW6Cc%3D. [Online; accessed: june-2015].

[12] A survey on coherent and non coherent receiver for gmsk signal. `http://pnrsolution.org/Datacenter/Vol3/Issue1/142.pdf`. [Online; accessed: june-2015].

[13] USRP2 datasheet. Ettus Research.

[14] Wbx ettus research. `http://www.ettus.com/product/details/WBX`. [Online; accessed: june-2015].

[15] Anil K. Maini & Varsha Agrawal. *Satellite Technology: Principles and Applications.* John Wiley & Sons, Inc, 2 edition, 2011.

[16] The CubeSatProgram. Cubesat design specification. `http://www.cubesat.org/images/developers/cds_rev13_final2.pdf`, 2015. [Online; accessed june-2015].

[17] Simon Haykin. *Communication Systems.* John Wiley & Sons, Inc, 4 edition, 2000.

[18] Louis J. Ippolito. *Satellite Communicaiton Systems Engineering.* John Wiley & Sons, Inc, 2 edition, 2008.

[19] K. Hirade K. MUROTA. *GMSK Modulation for Digital Mobile Radio Telephone.* IEEE, 1 edition, 1981.

[20] Theodore S. Rappaport. *Wireless Communications: Principles and practice.* Prentice Hall PTR, 2 edition, 2002.

Appendix

Files added electronically:

- NGHam encoder implentation c++ file.
- NGHam encoder implementation c++ header file.
- NGHam encoder implentation python QA code.