Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Project Assignment

Satellite communications

Ground Station Based on Software Defined Radio

Author:
Karl David VEA
Petter S. Storvik

Supervisor:
Torbjörn Ekman
Roger Birkeland

December 19, 2014

NTNU – Trondheim
Norwegian University of
Science and Technology

Problem Statement

A ground station is a very important part of a satellite communication system. Sending a satellite to space would not be of much use if no ground station is present. The use of Universal Software Radio Peripheral (USRP) minimizes costs for such systems and ads flexibility. A ground station based on Software Defined Radio is a popular way to go for student satellites. Student satellites often use HAM radio frequencies at VHF and UHF bands. A suited software platform for development of the radio should be decided. The choices are GNURadio and LabVIEW. Transmission protocol, including the use of error correction and ARQ, must be explored. To do this a simple link budget based on parameters from the NUTS project is to be presented. The most important propagation impairments should be presented and taken into consideration. The communication system should be simulated and some results from the simulation need to be introduced.

Abstract

The NTNU Test Satellite (NUTS) is a project where the aim is to build and launch a small satellite consistent with the CubeSat. Because no coding or protocol is present in the project, this report attempts to clarify and decide on a suited coding and package structure. To do this in a tidy and efficient manner, a link budget has been introduced, using previous link budget in this and similar projects, and some estimates of the link margin were made. A suiting bit error rate requirement was found to be 10^{-3} for the downlink and no errors in the decoded information on the uplink. This resulted in a link margin of 6.76 dB and 27.29 dB on the downlink and uplink respectively. Due to the need of error free data on the uplink, error correcting codes were compared to Hybrid ARQ. The latter was found to be more efficient regarding redundancy, hence it was chosen as the preferred transmission scheme. Reed-Solomon error correction was considered, both by itself and concatenated with convolutional coding. A complete communication system, consisting of a BFSK transmitter, AWGN channel and BFSK receiver, was successfully implemented in LabVIEW for simulation purposes. Simulation results indicate a coding gain of 4.2 dB in the case of Reed-Solomon only, and additional 2.2 dB concatenated with convolutional coding. The possible coding gains were taken into consideration and weighted against the computational needs for the coding/decoding algorithms. This resulted in the decision to omit convolutional coding as Reed-Solomon alone should be sufficient. Three possible existing protocols were introduced, AX.25, AAUSAT3 and NGHAM. As NGHAM is already implemented in the VHF radio of the satellite and is using Reed-Solomon only, which was proven sufficient, this was decided as the most appropriate protocol in regards to the NUTS project.

Contents

List of Figures

List of Tables

1 Introduction

NUTS is a student satellite project where the aim is to build a fully functional satellite. The satellite has to be in accordance with the cubesat standard, and will mostly take pictures and send sensor information back to the earth. This report will focus on communication with the satellite from a ground station perspective. The idea of the ground station is to use a Universal Software Radio Peripheral (USRP). This means that the radio is programmed in software, and is easier to change/improve. To program the USRP, two possible software platforms can be used.

- LabVIEW - National Instruments software with a broad area of usage.
- GNURadio - An open source project which aim to make software radio and signal processing available for everyone.

In Section 4.1., these two programs will be compared and the choice will be justified. A general link budget is explained in Section 4.2., along with some in depth characterizations of the satellite communication channel. The USRP specification along with an existing link budget will be used to present an improved link budget for the system. Based on the resulting link budget and NUTS project specifications, three existing alternative protocols will be explored. These are the AX.25, NGHAM and the AAUSAT3 protocols and will be explored in Section 4.4. Grounds for deciding on a protocol is going to be based on these three protocols, and advantages and disadvantages will be further discussed. All discussions follow in Section 6.

A complete communication system based on the NUTS project will be implemented in the chosen software platform. For this implementation, see Section 4.5. The code for this simulation can be found in Appendix D. This implementation includes losses, transmitter and receiver. Various coding schemes is implemented and noise resilience is tested.

2 System Description

2.1 The CubeSat Program

The NTNU Test Satellite (NUTS) is a student satellite project. The satellite is developed by students of the Norwegian University of Science and technology and requires a wide variety of different technological areas. NUTS' goal is to develop and launch a small satellite which follows the CubeSat Program. The CubeSat Program is a result of collaboration between three professors, Jordi Puig-Suari and Bob Twiggs, that started in 1999. They standardized the design of picosatellites and managed to reduce both cost and development time, and made launches more accessible for such satellites[6].

2.1.1 Overview

There are numerous design specifications which have to be followed to make a CubeSat. These are all listed in the CubeSat Design Specification[6], available on the CubeSat web page. Some important specifications are:

- Mechanical construction should be a 10 cm cube.
- A three unit CubeSat can't weigh more than 4 kg, that is 1.33 kg each.
- At launch the satellite should be inert from the launch sites perspective.
- All parts should remain attached to the satellite at all times.
- One RF inhibit and RF power output of no greater than 1.5W at the transmitting antenna's RF input.
- Documented vibration test, thermal vacuum test, shock test and visual inspection should be carried out according to launch provider.

The NUTS CubeSat will have an altitude of 600km, thus falling into the category of a geocentric low earth orbit (LEO) satellite.

2.2 The NUTS Project

2.2.1 Radio Specifications

As mentioned above the transmit power will be less than 1.5 Watts. The NUTS project have been using radio equipment designed for amateur radio, and therefore using Very High Frequency (VHF) and Ultra High Frequency (UHF) frequency bands. VHF band corresponds to 144 - 146 MHz and UHF corresponds to 432 - 438 MHz. These two frequency bands are the amateur radio bands, thus requires a HAM-radio licence. Therefore, frequency allocation is easier in this band, compared to other commercial bands. However, the use of HAM frequencies implies some restrictions in terms of sent effect and time usage. The HAM radio bandwidth is 25kHz, thus this is our limit. The amateur radio equipment are usually using the AX.25 protocol and this protocol will be explained later in the report.

The satellite is made to be fairly simple and robust, this is reflected in the onboard satellite radio. For this system the bit rate is twice the difference between the two frequencies representing ”0”s and ”1”s. The satellite is made to transmit GMSK modulated signals, which is a special case of FSK modulation.

2.2.2 Data Description

Downlink

- Telemetry - Sensor data describing the satellites' health
- Morse code - Beacon
- Payload data - Image and other sensor data

This data is sent from the satellite to the ground stations. There are different requirements when it comes to bit errors on the downlink and uplink. It is decided that a bit error rate of 10^{-3} is sufficient to use in the SNR calculations.

Uplink

- Commands - Satellite control commands

Because the uplink data are control commands it is very important that this is error free. If not, commands sent to the satellite will not be executed.

2.2.3 Other Radio Link Equipment

The ground station will consist of an USRP, explained later in this section. The receiving antenna for VHF is a Tonna 20818, 2x9 crossed Yagi-Uda with 13.1 dBi and for UHF a Tonna 20938, 2x19 crossed Yagi-Uda with 16 dBi [17]. The VHF antenna has a mast-mounted Low Noise Amplifier (LNA) of type SP-2000 while the UHF is of type SP-7000. It is connected to the antenna by a coax cable. Between the LNA and the USRP there is another coaxial cable of the type Aircell 7 with 3.5 dB cable loss. The receiver link will be further explained in Section 4.2.3.

2.3 USRP

Universal Software Radio Peripheral (USRP) is an inexpensive hardware platform for using software defined radio (SDR). The USRP does the high speed digital signal processing such

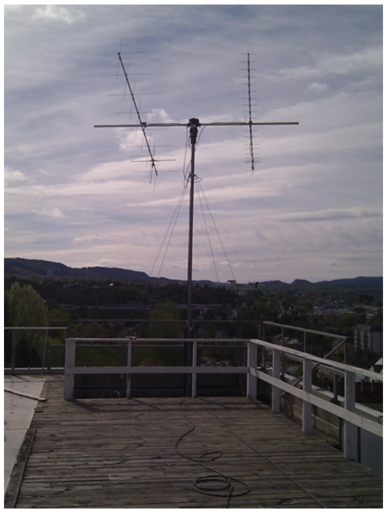


Figure 1: Yagi-Uda antenna

as ADC/DAC, up/down converting and filtering while the host computer does the low speed operations.

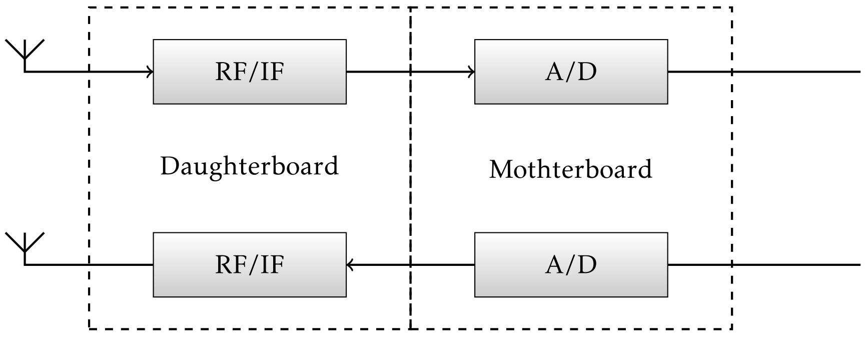


Figure 2: USRP

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/down in frequency. In total, the USRP can be used in the frequency range from DC to 6 GHz. This is done by swapping the daughterboard to a daughterboard that supports the needed frequency range.

The daughterboard used in the USRP is the WBX board, this is a transceiver with 40 MHz bandwidth, supporting bands from 50 MHz to 2.2 GHz. It provides an output power level of up to 100 mW and a noisefigure of 5 dB[18].

3 Theory

3.1 Modulation

When it comes to modulation scheme, the ground station have to demodulate the signals from the satellite. In the system description the satellite transmitter was briefly described, and the modulation form was frequency shift keying(FSK).

3.1.1 Binary Frequency Shift Keying

Binary FSK transmits 0 & 1 by assigning a frequency shift to the carrier. This way we can decide what bit was sent by looking at the received frequency. BFSK is considered as a very easy modulation scheme at the transmitter side, as it can be constructed by only a microcontroller and a voltage controlled oscillator.

A conventional FSK can be expressed as:

$$s_i(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c t \pm 2\pi \Delta f), \qquad 0 \leq t \leq T_b \tag{3.1}$$

where T_b is the symbol duration and $\pm 2\pi\Delta f$ is the sent bits[13, p.312]. To make an FSK signal more spectral efficient the phase needs to be continuous, i.e. no rapid changes in the signal. It can be shown that to make this happen the phase shifts between "1" ($s_k = 1$) and "0" ($s_k = -1$) must correspond to $\pm\pi h$.

3.1.2 Minimum Shift Keying

One problem with sending binary data, is that the transitions between a "0" and a "1" results in large sidebands. In a communication system it is important to not waste bandwidth, you have to stay within "your" allocated bandwidth and you do not get to interfere with other bands. Minimum Shift Keying (MSK) deals with this problem and is a special case of Continuous Phase FSK. The signal is characterized as a narrowband FM-signal. One of the reasons this signal is more bandwidth efficient is because it has a continuous phase[13, p.312].

Because MSK is a special case of CPFSK it can be explained by using the equations above. Using the result form CPFSK, we can find that the modulation index for MSK is given by:

$$h = \frac{2\Delta f}{R_b} = 0.5 \qquad \text{and} \tag{3.2}$$

$$\Delta f = \frac{1}{4T} \tag{3.3}$$

It can be shown that the signal frequencies for MSK are $f_c \pm \frac{1}{4T}$, thus the minimum distance between the two frequencies in MSK is $\Delta f = \frac{1}{2T}$. This minimum distance is the minimum distance that allows orthogonal detection of two FSK signals.

3.1.3 Gaussian Minimum Shift Keying

Gaussian Minimum Shift Keying (GMSK) is closely related to MSK modulation, but it is even more bandwidth efficient. This is achieved by using a premodulation Gaussian pulse-shaping filter. Gaussian pulse-shaping stabilizes the instantaneous frequency variations over time, hence reducing the sidelobe levels in the transmitted spectrum. The constant envelope in a GMSK signal ensures great power efficiency which is a very attractive property in wireless communications. Because the Gaussian filtering does not affect the phase trajectory, one can detect GMSK in the same way as MSK.[13]

3.2 Channel

Channel capacity is the upper limit for the rate of data possible to transmit without intersymbolinterference. Though there exists methods for exceeding this limit, it is often considered as the limit for error free communication.

$$C_{\text{awgn}} = W \log_2(1 + SNR) \qquad \text{(bits/s)} \tag{3.4}$$

where W is the bandwidth for the used channel.

3.3 Error Correcting Codes

This section will cover channel coding. Source coding will not be considered here because it is outside the scope of this report. The principle is that by adding redundant information/bits to the sent message, it is possible to discover and in some cases correct bit errors[11]. Channel coding is in general considered independent of the modulation form. There are many different techniques for doing this.
There are many types of Forward Error Correcting (FEC) codes. Most important are block codes, convolutional codes and turbo codes[13, p.356]. These do all add redundancy to recover bit errors in the received signal.

Block Codes: FEC-codes that allows recovery of bit errors. These codes are used on chunks/blocks of bits of a given length. Parity bits are added to these blocks, thus creating a coded bit stream called codewords. A (n,k) block code takes k information bits and encodes them into n code bits. This gives $n-k$ added redundant bits. The rate of a block code is defined as[13, p.395]:

$$R_c = \frac{k}{n} \tag{3.5}$$

Convolutional Codes: Codes that achieve more or less the same results as block codes, but do not divide the bit stream into blocks. Instead, the bit stream is continuously mapped to an encoded output stream. The advantage of convolutional codes is that you are able to get a larger coding gain than block codes of the same complexity. The encoded bit stream is generated by passing information bits through a shaft register. k bits at a time is shifted into the register and for each k bits it outputs n bits. The rate of a conventional code is then given by the same formula as for block codes, in Equation 3.5[13, p.407].

Turbo Codes: Turbo codes are the most modern codes. The principle is to combine convolutional codes with channel estimation theory. These codes allows for wireless communication to come close to Shannon capacity bound, and have a larger coding gain than convolutional codes and block codes[13].

Concatenated codes: These are not a separate code type, but a way to use multiple codes to code information. This is achieved by using an outer and an inner code. The outer code is applied first and the inner code is then added before the channel. This allows for using different coding types with different properties[13].

4 Methods

4.1 GNURadio vs LabVIEW

The ground station will be based on Software Defined Radio (SDR) and USRPs, as explained in Section 2.3. For this there are two obvious software choices, GNURadio and LabVIEW. The main focus when making this decision was on GNURadio, due to prior knowledge and experience with LabVIEW. This section will first consider them individually, and they will furthermore be compared followed by a justified conclusion.

GNURadio is a very popular open source project aimed to make SDR radio available to anyone. Equipped with GNURadio 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 GNURadio can both be graphical or a more traditional programming approach, i.e. text based programming. While the graphical approach is a separate program called *Gnu Radio Companion*(gRC), the latter can be programmed using your favourite text editor. The programming language is python, which is a fairly easy and powerful language. Because python is considered to be a slow language, the main blocks used in GNURadio are made in C. It is also possible to make your own blocks in C, thus providing unlimited possibilities. Even though help and support regarding GNURadio is hard to find on campus, the community is large and the response time on the forums are short. As for setting up the system, it is done in a fairly simple manner. Because the USRP uses a standard networking cable, there are no driver problems, which usually exists in the Linux world. One problem is the program flow, which seems to be difficult to monitor and control. There is also a pretty steep learning curve for beginners.

LabVIEW is a programming language for various tasks. While GNURadio focuses on SDR, LabVIEW also solves other tasks having a lot of different supported hardware. The way it differs from conventional programming is that it consists of ”blocks” and ”wires”. This is quite similar to gRC discussed above. These blocks can be thought of as functions and the wires are paths which data travels along. There are no way of writing LabVIEW-code, as it is its own programming language, but there are ways of inserting C/Matlab code. LabVIEW was not initially developed to support USRP development, but USRP support has been made later. Combining this with the rather powerful modulation toolkit, signal processing can be done in an efficient way. Without this modulation toolkit it would be a lot of work to develop in LabVEIW, and therefore it is a necessity. This limits the development time for such systems. Although the learning curve in LabVIEW also is somewhat steep, some prior knowledge to SDR in LabVIEW reduces the learning time.

Comparing these two we, can see that both approaches can reach the same goal. As mentioned, LabVIEW is a more advanced programming language that not only focuses on radio. Therefore, there are many options when it comes to data manipulation. This means that the whole ground station, from receiving the data to displaying it and saving it to a file, will be fairly simple to implement in LabVIEW. As for GNURadio, it can be a little more work due to the lack of all data manipulation tools. However, this does not mean that it is impossible, in fact there are many complex systems implemented with GNURadio. When it comes to the resources for getting help with the two, there are no big difference. While we got contact from National Instrument and forums for getting help with LabVIEW, GNURadio has a huge community of users. Therefore both programs is considered very powerful and capable to solve our problem. It all comes down to personal preference and the fact that LabVIEW simplifies the process of the overall ground station design. LabVIEW is also introduced to students at NTNU at an early stage and therefore some previous experience in LabVIEW exists.

4.2 Link Budget

Satellite communication is communication between a satellite with a given orbit and some receiver/transmitter on Earth. This type of communication is in general line of sight (LOS), and is characterized by high attenuation. Often preferred frequencies used for satellite communications are frequencies above 3 GHz due to the fewest propagation and attenuation problems caused by the ionosphere. The lower limit for satellite communication is the ionospheric penetration frequency, which is 30 MHz[10]. As mentioned the NUTS satellite uses VHF and UHF frequencies that lies far below 3 GHZ. First different propagation impairments must be mentioned and explained. The values used in this chapter are based on the link budget in appendix B and C and a previous master assignment[7]. Because a thorough analysis of the link budget parameters falls outside the scope of this report this link budget is based on an early estimate of the satellite channel, and is not guaranteed to match the real physical conditions.

4.2.1 Free Space Path Loss (FSPL)

The free space path loss (FSPL) is given by[4]:

$$\text{FSPL} = (\frac{\lambda}{4\pi R})^2 \tag{4.1}$$

where R is the distance between the two antennas. This equation describes the loss between two antennas with line of sight communication in free space. In this case, the FSPL is estimated to 134.24 dB[15]. However, this will be time dependant, as the distance between the satellite and the earth station will vary. As mentioned the distance will be 600 km. Friis transmission equation relates the transmitted and the received power in a communication system. This equation is closely related to the free space path loss between two antennas.

$$\frac{P_r}{P_t} = \text{FSPL}\ G_{0t}G_{0r} \tag{4.2}$$

where G_{0t} and G_{0r} are the two gains in the transmitter and receiver antenna.

4.2.2 Propagation Impairments

The signal from a satellite will be attenuated and distorted in the troposphere (0-20km) and the ionosphere (80-1000km). Effects from the ionosphere decreases as the frequency increases and above 3 GHz the ionosphere is more or less transparent. From 3 GHz and above, the troposphere effects is the main problem. There are many sources of attenuation and distortion, most important are gaseous absorption, cloud and rain attenuation, ionospheric attenuation, refraction and Faraday rotation in the ionosphere. We can divide propagation loss into two groups; those that are relatively constant and predictable, and those that are random and unpredictable[2]. Since the satellite is communication on the VHF/UHF frequencies, propagation impairments for frequencies below 3GHz will be explained.

Doppler shift is a frequency shift caused by relative motion between the transmitter and receiver[14, p.229]. The doppler shift is given as:

$$f_d = \frac{v}{\lambda}\cos\alpha = f_c\frac{v}{c}\cos\alpha \tag{4.3}$$

where f_c is frequency of original wave. Maximum doppler shift is therefore:

$$f_m = \frac{v}{\lambda}\cos 1 = \frac{v}{\lambda} = f_c\frac{v}{c} \tag{4.4}$$

Polarization impairments are often seen in the satellite channel. The polarization of a radio wave will rotate due to electrons in the ionosphere and the earth's magnetic field. This rotation is known as the Faraday effect[10] This effect can have serious impact on VHF band space communications when linear polarization is used. Because the average value of the Faraday rotation is fairly predictable, most of the time it can be compensated for by manually adjusting the the polarization angle on the ground station antenna[10, p.99]. Faraday rotation is only a problem for linearly polarized signals. In our case the polarization on the ground station antenna and the satellite antenna are not matched. The cubesat transmit antenna is linearly polarized, this would optimally be circular but due to the complexity it is not realizable. It should be noted that Sigvald Marholm has explored implementation of a circular polarized antenna for the VHF radio[12]. The receiver antenna on the ground station is circular polarized and this compensates for polarization mismatch by guaranteeing a maximum 3dB loss[2].

Coherence bandwidth caused by ionospheric dispersion and other atmospheric causes will not be a problem. This is due to the fact that for frequencies up to 30 GHz, the coherence bandwidth always exceeds about 1 GHz[10].

Group delay is a reduction in the propagation velocity of a radio wave compared to the free space path, due to the presence of free electrons in the propagation path. At VHF/UHF frequencies the delay can be up to 10 μs under maximum electron density conditions[10].

Ionospheric Scintillation is rapid fluctuations of the amplitude and phase of a radio wave and is caused by electron density irregularities in the ionosphere. The effect is observed in the frequency range from 30 MHz to 7 GHz, but mostly in the VHF band from 30-300MHz. This can have a big impact on the limitations of reliable communication. It is most severe in the equatorial, auroral and polar regions. The result of ionospheric scintillation is diffraction and forward scattering[10].

4.2.3 Link Budget

A link budget describes the SNR in the satellite channel. It includes all gains and losses in the system and is a very important tool when designing a radio link. In general the link budget can be expressed in its general form as[2]:

$$P_R = P_T - L_T + G_T - L_P - L_A + G_R - L_R \tag{4.5}$$

Where:

P_R is the power received

T_T is the power transmitted

L_T is loss in the waveguide between transmitter and antenna

G_T is the transmitter antenna gain

L_P is the free space path loss described in the chapter above

L_A is additional losses, i.e attenuation due to rain, clouds, etc

G_R is the receiving antenna gain

All gains and losses are expressed in dB. This general equation operates with an additional loss factor. This factor is described as various attenuations caused by the ionosphere and troposphere. Some of these were explained above. In this case additional losses can be expressed as:

$$L_A = L_{pol} + L_a + L_i + L_{point} \tag{4.6}$$

Where:

L_{pol} is polarization loss

L_a is atmospheric loss

L_i is ionospheric loss

L_{point} is pointing loss

By inserting the values provided by the link budget estimate the received power can be calculated. But since this link budget does not consider the receiver to be a USRP, the receiver part of the budget is wrong.

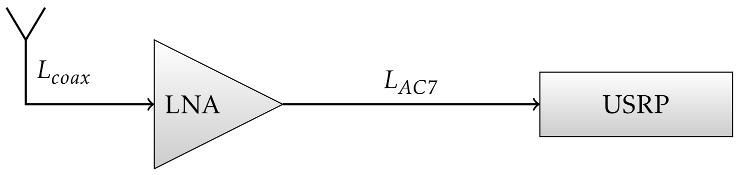


Figure 3: Receiver structure

As seen in Figure 3, the receiver consists of a cable between the antenna and the LNA, and one cable between the LNA and the USRP. This antenna type, as explained in Section 2.2.3, is a Yagi-Uda antenna. Antenna gains are specified in the antenna specification[17]. The loss for the Aircell 7 coaxial cable is given in the datasheet[3] as 7.6 dB/100m for 144 Mhz and 13.6 dB/100m for 432 Mhz. This gives the following losses for a 40 m cable:

$$L_{AC7,UHF} = \frac{40m}{100m} \cdot 13.6\text{dB} = 5.44\text{dB} \tag{4.7}$$

$$L_{AC7,VHF} = \frac{40m}{100m} \cdot 7.6\text{dB} = 3.04\text{dB} \tag{4.8}$$

Assuming the same cable is used both pre- and post-LNA(i.e $L_{coax} = L_{AC7}$), loss in the cable before LNA can be computed equivalently. Table 1 consists of some key parameters for link budget calculations. While some values are updated according to new information others are found from existing link budgets in the project.

Parameter	**UHF (up)**	**UHF (down)**	**VHF (up)**	**VHF (down)**
Frequency	437 Mhz	437 MHz	145 MHz	145 MHz
G_T	16 dBi	2.15 dBi	13.1 dBi	2.15 dBi
P_T		500 mW		500 mW
G_R	2.15 dBi	16dBi	2.15 dBi	13.1 dBi
L_{pol}	3.00 dB	3.00 dB	3.00 dB	3.00 dB
L_a	0.30 dB	0.30 dB	0.30 dB	0.30 dB
L_i	1.00 dB	1.00 dB	1.00 dB	1.00 dB
L_{point}	0.50 dB	0.50 dB	0.50 dB	0.50 dB
$L_{AC7,40m}$	5.44 dB	5.44 dB	3.04 dB	3.04 dB
$L_{AC7,6m}$	0.82 dB	0.82 dB	0.46 dB	0.46 dB

Table 1: Link Budget Parameters

Because the satellite communication will be based on the UHF radio while the VHF radio is meant as a backup, the following link budget will only be calculated for the UHF. Noise is generated in the receiver structure. The resulting link budget is derived from previous link budgets[7]. Since the complete link budget calculations falls outside the scope of this project, these values are meant as a rough estimate for real values.

Downlink:
Link budget for the downlink is given as[7]:

$$\frac{E_b}{N_0} = 16.56\text{dB} \tag{4.9}$$

As explained above, this is only a rough estimate, and not the exact value. The consequence of this will be taken into considerations in the discussion part.

Uplink:
For the uplink the link budget is[7]:

$$\frac{E_b}{N_0} = 37.09\text{dB} \tag{4.10}$$

Both the downlink and uplink link budget will be discussed later.

4.2.4 BFSK Requirements

The bit error rate for coherent BFSK detection is derived from Equation 3.1 and given by[8]:

$$P_{e,BFSK} = \frac{1}{2}\text{erfc}(\sqrt{\frac{E_b}{2N_0}}) \tag{4.11}$$

As explained in Section 2.2.2 10^{-3} is used as maximum allowed bit error rate for the system. To obtain a bit error rate of 10^{-3} this is plotted using the matlab script in Appendix A. This value will indicate which link margin is needed for the system to be able to decode BFSK-symbols.

4.2.5 USRP Requirements

Receiver sensitivity is the minimum input SNR that can be detected of a radio receiver and provide usable information. This could be done by connecting the USRP to a high quality signal generator and measure the lower limit that can be detected. Due to time and equipment limitations, this was not done. Instead, the value was found from a similar measurement[15]. This measurement showed a result of -130 dBm.

It should be noted that the accuracy of the USRP timing can be enhanced using a GPS clock. This would improve the link budget by some decibels, due to correct timing. An external clock is fairly cheap and can be connected to an input on the front of the USRP. Alternatively, a crystal oscillator can be used in the USRP. While this provides more accurate results, it is a more complicated and expensive approach.

4.3 Coding

4.3.1 Automatic Repeat-request

ARQ is applied to ensure error free transmission of packets. The principle of ARQ is to retransmit received packets with bit errors, and it therefore relies on an error detecting code. This way a small number of bits are used when the channel condition is good, and more bits are used if channel conditions are bad. This is due to the retransmission of packets. There are several types of ARQ. The most trivial one is ”stop and wait ARQ”, which transmits a package and waits for a response from the receiver. These responses are called ”ACK”, positive acknowledgement for correct reception, and ”NAK”, negative acknowledgement for incorrect reception. One problem with the ”stop and wait ARQ” is its efficiency, which suffers from the fact that the transmitter must wait for the response before transmitting the next package. To cope with this problem a more efficient ARQ can be used, the ”go back n ARQ”. Instead of waiting for the ”ACK”/”NAK”, the transmitter transmits n frames before waiting for acknowledgements. However, this form of ARQ requires retransmission of several packages, not just the one containing the error. Therefore ”selective repeat ARQ” is the most efficient transmission scheme. This form of ARQ allows for retransmission of the package containing the error, thus keeping its efficiency high and retransmitted bits low.

As stated above, ARQ relies on an error detecting code. Compared to an error correcting code, a error detecting code needs fewer redundant bits added to information bits. In Hybrid ARQ (HARQ) a FEC-code is applied and tries to correct a given number of errors. The parity bits can either be sent in the package, or it can be sent upon a request. If there are more errors than the FEC-code can correct, the receiver uses ARQ as fallback to ensure error free communication. This is resulting in HARQ being better than normal ARQ in bad conditions, but more demanding and unnecessary in good conditions. Type 1 HARQ is the simplest form for ARQ and adds error detection and error correction prior to transmission. Errors remaining after the error correction will be detected, and the receiver asks for retransmission. Type 2 HARQ is slightly more efficient. Here the parity bits are only sent if the receiver needs them, thus minimize the number of bits sent.

Coding is a task that often is computational hard to decode, but fairly easy to encode. Due to this the coding used for the downlink will be less dependant on the microcontroller and the coding for the uplink must be easy to decode. HARQ makes it possible to send less bit when a channel is good, but requires a approximately error free uplink. This will be a problem, especially when we are limited to fairly easy coding schemes. The time it will take to decode the message will therefore impact the efficiency of the HARQ. Another problem is the complexity of HARQ. As opposed to just coding the bits and sending them, it also have to store previously sent bits and wait for responses from the ground station.

4.3.2 Error Correction

To avoid depending on the uplink, a widely used method was considered. As mentioned in Section 3.3 on page 11 concatenated coding are often used in similar types of communication. This is due to the convolutional code interfacing natural to the real world, and the Reed-Solomon correcting errors within a block. The resulting block diagram for this operations are viewed in Figure 4.

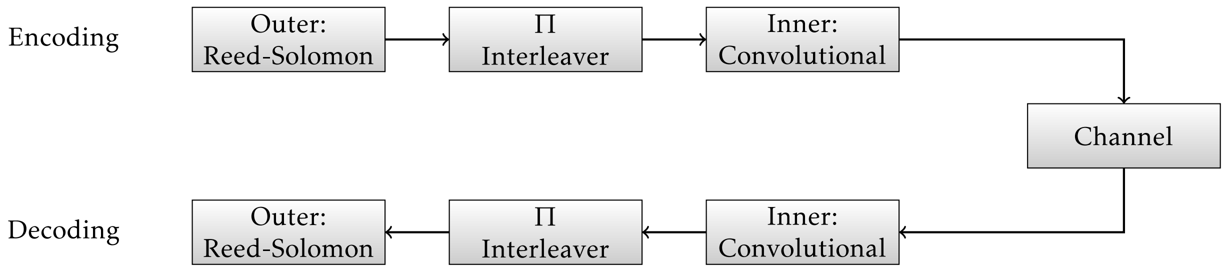


Figure 4: Concatenated coding

Reed-Solomon (RS) codes are symbol based error correcting codes(nonbinary codes). This means that bits are combined into symbols before the encoding is done. It is most effective against errors that appears in bursts. Reed-Solomon codes are often denoted (n,k)RS, where n is number of symbols per bloc and is defined as $n = 2^m - 1$, and $k < n$ denotes data symbols in each block[13]. The microcontroller used in NUTS are 32 bit controllers, but as it is more convenient to work with 8 bits / 1 byte, this will be used in the calculations. This gives:

$$n = 2^8 - 1 = 255 \text{ symbols per block} \tag{4.12}$$

The number of errors correctable for a RS code is given by:

$$\begin{aligned} e &= \frac{n-k}{2} \\ 2e &= n-k \end{aligned} \tag{4.13}$$

If the RS code should be able to correct 16 errors per block, we get:

$$\begin{aligned} 2(16) &= 255 - k \\ k &= 255 - 32 = 223 \end{aligned} \tag{4.14}$$

This (255,223)RS code is a very commonly used code that can be found in applications from compact discs to deep space communications[10]. Using this code will result in adding 32 redundant bits per 223'th data bit, i.e. adding 32 bits per block. Given the channel capacity and other channel conditions explained in the previous sections, this should be sufficient outer code in our system.

Convolutional codes are briefly explained in Section 3.3 on page 11. As explained above the reason for using a convolutional code as an inner code is that it interfaces natural to the real world. A much used convolutional code is a code with a constraint length $k = 7$ and the rate given by:

$$R_c = \frac{1}{2} \tag{4.15}$$

Constraint length can be described as a measurement of how powerful the convolutional code is. The downside with a high constraint length is the complexity of the decoder increases exponentially. Therefore, a constraint length above 7 is not worth it in respect to gain and complexity. Most used for decoding a convolutional code is the Viterbi algorithm[13]. This is a widely used algorithm which aims to find the most likely path through a trellis that results in a given sequence.

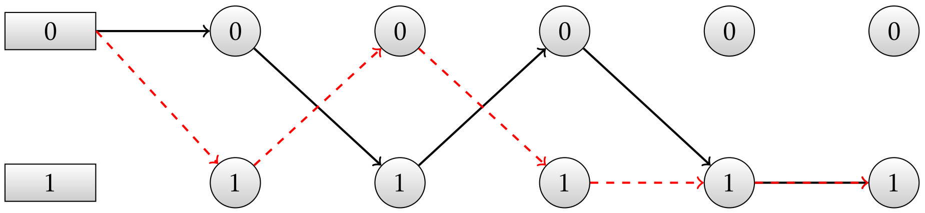


Figure 5: Example trellis. Sequence "001011"

As illustrated in Figure 5, the decoders' goal is to follow the right path. In this case the black path is the right bit sequence. Often errors will come in bursts[2]. For instance, the red arrows in the figure is a burst error making the decoder take the wrong path. Therefore the decoder will follow the right path until a burst occurs, then taking the wrong path from this point. Because a burst error usually does not last for that long, the decoder eventually gets into the right path. If the bit transmitted bit sequence is fairly large, these bursts will harm a small amount of bits. As explained above, the Reed-Solomon code operates on blocks. The RS decoder will make an error if a large amount of bit errors occur in one block. Thus using interleaving between the RS and the convolutional code will dampen the effects of a burst error.

Interleaving is the process of shuffling a bit sequence. This technique will spread the bits in time making it more robust against burst errors. As mentioned, RS can correct up to 16 bits per block. This means that a burst error can result in one block containing errors. But by spreading the bits within each block to random other blocks, a burst error will not be that dangerous as the bit sequence is reordered before RS decoding[13]. Interleaving is not discussed later in this report due to computational limitations at the satellite.

4.4 Protocol

To choose a protocol, some existing options was explored and to some extent compared. There are three possible existing protocols that matches the usage in this case, AX.25, NGHAM and the protocol used in AAUSAT3. Even though none of these fit the goal of the NUTS project directly, they can be used if some minor modifications are applied.

4.4.1 NUTS Requirements

The NUTS satellite is going to transmit, as described in Section 2.2.2, mainly image and sensor data. For internal package transport it is decided to use CSP. This is a transport protocol developed by Aalborg University and is tied to the AAUSAT3 protocol described in Section 4.4.4. The CSP must be inserted into the encoded information bits. Most relevant existing protocols will be further explained.

4.4.2 AX.25 Protocol

The NUTS project has until now based its work on the AX.25 codec and amateur radio equipment. This is a data *link layer protocol* based on previous X.25 protocol and are mostly used in amateur radio[16]. Data link layer is a layer in the Open Systems Interconnection model (OSI), which is a conceptual model that aims to standardize different internal functions in a communication system. It is divided into two sublayers, the Media Access Control (MAC) and Logical Link Control (LLC) layers. MAC layer is controlling access to the different parts of the network and LLC layer is doing error detection and synchronization. Therefore, the AX.25 is for transporting data between nodes, ensuring that errors will be detected. AX.25 operates with three different frame types:

1. Information frame (I frame)
2. Supervisory frame (S frame)
3. Unnumebered frame (U frame)

Flag	Address	Control	PID	Info	FCS	Flag
1 byte	14/28 bytes	1/2 bytes	1 byte	256 bytes	2 bytes	1 byte

Figure 6: AX.25 Information Frame

Figure 6 shows that the protocol has a rather extensive header, this means that the data rate will be unnecessary high. In general the AX.25 is not optimized for operating under a noisy channel. This is due to the lack of error correcting, i.e. it only have error detection. It also lacks some data compression technique, even though this is not considered as a problem as it can be applied pre-AX.25. Frames that does not consist of $N \cdot 8$ bits and/or starts and ends with the flag byte, will be considered as an invalid frame.

4.4.3 NGHam

Another newer protocol for this is the NGHam protocol, 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, and will therefore be the least time consuming approach.

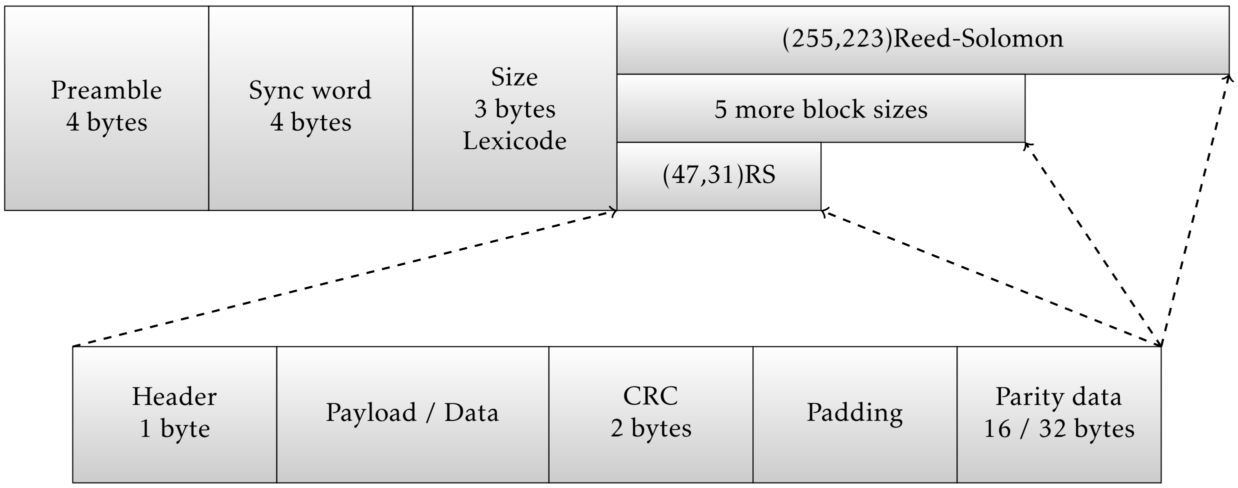


Figure 7: NGHam Protocol

As seen in Figure 7, frames can have 7 possible lengths. This will reduce redundant padding bits if less information is sent. Therefore, it will be more efficient than the AX.25 in terms of bit rates. 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.

4.4.4 AAUSAT3

Another much respected protocol for this usage, is the AAUSAT3 developed protocol. The AAUSAT3 is the third CubeSat from Aalborg university, and they therefore have some experience with satellite link protocol design.

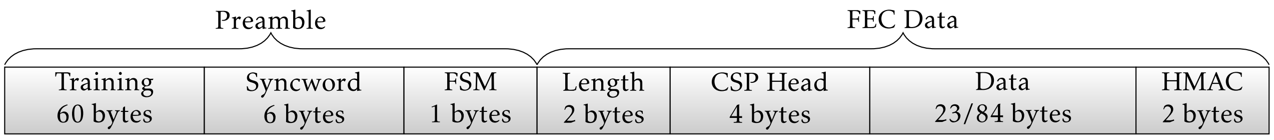


Figure 8: AAUSAT3 frames

As seen in Figure 8, the protocol starts with a variable length training sequence which defaults to 60 bytes. The syncword is a synchronization sequence of 6 bytes, followed by a Frame Size Marker which tells if the received package is long or short. CSP Header is a TCP/IP replacement, developed at the Aalborg University by the CubeSat team. The data can be either 23 or 84 bytes long, depending on the FSM information. AAUSAT3 are using a convolutional inner code and a Reed-Solomon outer code. They are not using FEC on the preamble[1].

4.5 LabVIEW Simulations

To test and verify ideas and system specifications, a complete communication system was implemented in LabVIEW. Not only does this simulation give important insights into the system itself, but it also verifies if the ground station is possible to implement on such a platform. This system can be downloaded using the link in appendix[INSERT HERE]. The simulation consists of four parts:

1. Transmitter
2. Channel
3. Receiver
4. Plots and calculations

4.5.1 Transmitter

The transmitter part can be seen in the upper part of the LabVIEW code. A representative simplified block diagram is shown below.

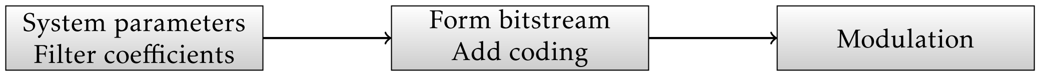


Figure 9: Transmitter Block Diagram

First the system parameters for FSK modulation along, with the pulse shaping filter coefficients, must be constructed. In LabVIEW this is done with a ”Create system parameters” and a ”Create filter coefficients” block. While the system parameters set parameters for the modulation scheme, such as type of modulation and samples per symbol, the filter coefficients will define the filter used for both pulse shaping at the transmitter and matched filter at the receiver.

One import aspect to the communication between the satellite and the ground station was coding. Therefore the LabVIEW simulation aimed to simulate three different coding schemes, letting the user choose between them. This way the decision can be based on the simulation as well as the properties of the coding and the link budget. The three simulated bitstreams are almost identical, but the amount of information bits differ.

No coding is the general test case. Here nothing is done to the information bits. When this mode is selected, the length of the information bitstream is 1000.

Reed Solomon error correcting coding. Here a normal (255, 223)Reed-Solomon code is applied according to Section 4.3.2. To do this in LabVIEW the bitstream must be converted to a symbol stream. The symbol stream a RS encoder/decoder block expects is an integer stream consisting of integer values formed by $log_2(n+1)$ bits. Consequently, the resulting number of information bits will be $255 \cdot log_2(n+1) = 2080$. This can be achieved by using the ”MT Pack Bits” which packs bits into integers of a given length. It is important to ”Unpack Bits” after Reed-Solomon encoding, converting the integer stream to bitstream again. This to make the modulator able to modulate the bitstream. Note that this is done with a user created SubVI called ”UserEncoder.vi”.

Concatenated coding is described in Section 4.3.2. This does Reed-Solomon encoding according to the paragraph above, but adds a convolutional code on top of that. As explained, the rate of the convolutional code is 1/2. This gives twice as many information bits as the Reed-Solomon case, $1040 \cdot 2 = 4160$. The convolutional encoder/decoder in LabVIEW is in this case set to operate with the rate 1/2 which results in a predefined generator matrix. Its important to note that the convolutional decoder in LabVIEW loses some information bits. To cope with this, there has to be added padding bits in before decoding. The amount of padding is given as $D \cdot n$, where D is the number of trellis stages the Viterbi algorithm uses. In this case this corresponds to adding $15 \cdot 2 = 30$ bits in the end of the bitstream. This because LabVIEW uses $D = 15$ by default.

Guard bits 4 bits	Sync bits 20 bits	Information bits 1000/2080/4160 bits	Back padding 50 bits

Figure 10: Simulation package

The created package is illustrated above. Guard and padding bits are added to the package to deal with unwanted end initial and end effects of the FIR filter used for pulse shaping in the modulator. To avoid data loss the back padding length must be of a certain size. The formula for this is explained in LabVIEW help files. Total bits to be added after information bits is given by:

$$N = (L-1)\frac{log2M}{m} + \frac{L-1}{2} \tag{4.16}$$

It is important to design the system so that there are enough back padding bits to cope with these filter effects.

4.5.2 Channel

The channel part of the LabVIEW code can be seen in the middle of the code.

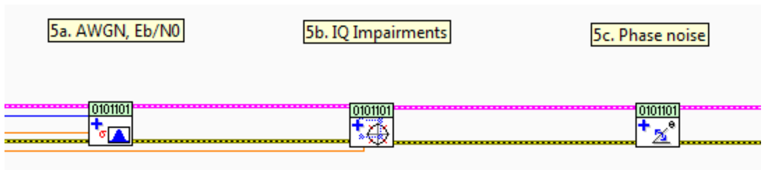


Figure 11: LabVIEW Channel

There are three different channel impairments added to the simulation.

1. AWGN - Takes a value for E_b/N_0 (controlled by a slider) and adds corresponding white Gaussian noise to the signal. This is one major factor in satellite communication since the satellite channel can be thought of as an AWGN channel.

2. IQ Impairments - Can simulate a lot of different IQ impairments, but mainly used to add a frequency offset. This offset can be controlled using a slider and is simulating the Doppler shift(see Section 4.2.2).

3. Phase noise - Adds phase noise to the signal. This is not used in this report, but added to allow for future testing.

4.5.3 Receiver

The receiver is the bottom of the LabVIEW code.

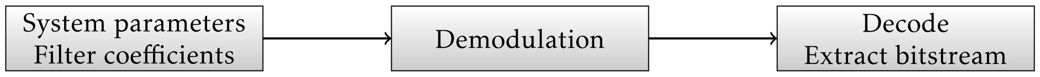


Figure 12: Receiver block diagram

For the receiver to demodulate the received signal, the same system parameters as in the transmitter must be applied. It also uses the matched filter coefficients to do the matched filtering. In this case the receiver demodulator consists of two different blocks, a ”demodulate” block and a ”detect FSK” block. The reason for this approach is that the ”demodulate” block returns some needed information that the ”detect FSK” does not, whereas the ”detect FSK” is the block best suited for the code. This returns the demodulated bitstream starting with the symbol where peak correlation to the sync bits were found. Therefore the bitstream will consist of sync bits, information bits and some back padding. The ”sync index” returns the index of the synchronization bits and using this the sync sequence can be removed and data extracted. Due to the different information bitstream sizes, the removing/extracting is done in the UserDecode block. This is a user defined SubVI that removes the sync and decodes the information bits if necessary.

4.5.4 Plots and Calculations

Several different plots and calculations were added to the LabVIEW code. These can all be viewed in the ”plots and calculations” part of code(top right). This part includes:

- Properties such as Rx and Tx length and BER.
- Eye diagram of the received signal.
- Constellation plot of the modulated signal after noise.
- Visual representation of information bits and decoded bits.
- Power spectrum of both transmitted and received signal.

LabVIEW modulation toolkit have many different tools for doing these, but the data into these blocks must be converted and configured to give correct input.

5 Results

5.1 Link Budget Calculations

5.1.1 Link Margin For BFSK

The matlab plot of BER is plotted in Figure 13 according to Equation 4.11.

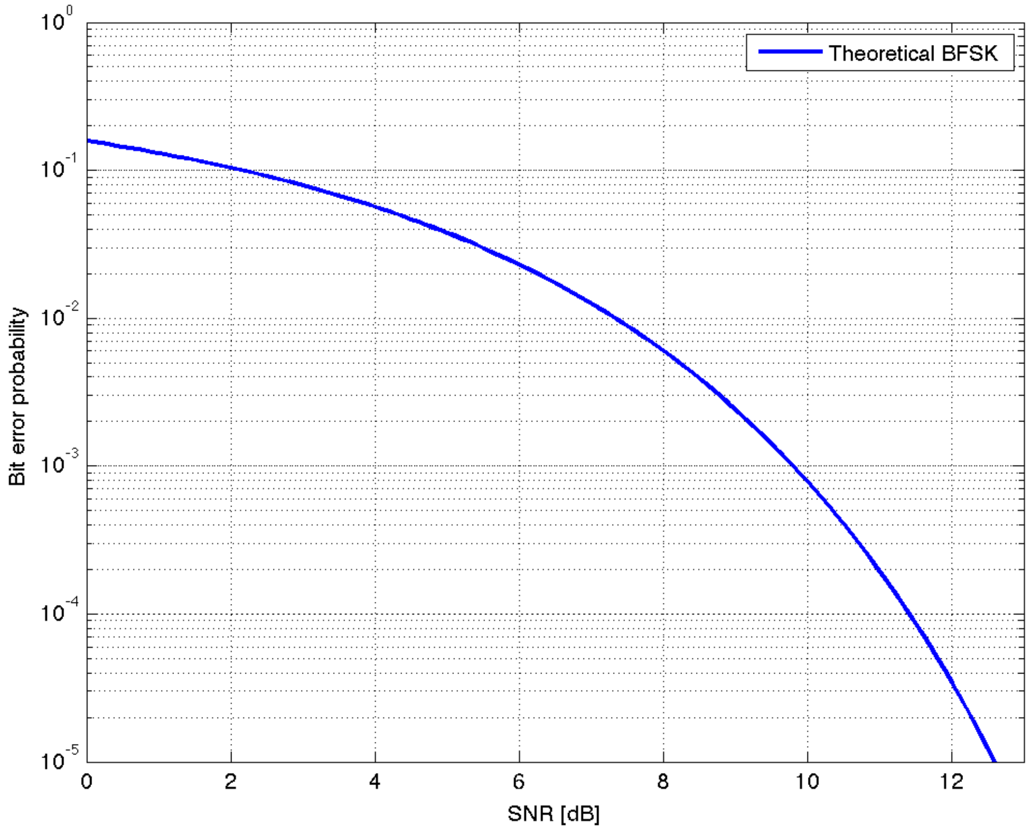


Figure 13: Theoretical BER for BFSK

As one can see, the bit error probability of 10^{-3} corresponds to a SNR of 9.8 dB. I.e. the needed link margin for decoding BFSK symbols is 9.8 dB.

5.2 Coding and Protocol

5.2.1 LabVIEW Simulation Results

The LabVIEW system implemented in this project can be used to simulate the gain acquired from coding. As explained, the simulation system consists of three different transmission systems, no coding, Reed-Solomon coding and concatenated coding. Simulation results are presented by varying the SNR to the limit where bit errors occur, then doing the same for the coding examples. Coding gain will then be the difference in dB between the cases.

No coding	**Reed-Solomon**	**Concatenated**
15.00 dB	10.80 dB	8.50 dB

Table 2: E_b/N_0 before bit errors

5.2.2 Channel Capacity

Using Equation 3.4 the channel capacity can be calculated. In this case the SNR is given in the link budget[7] as 12.40 dB on the downlink and the bandwidth for the channel is 25 kHz. Thus the channel capacity is:

$$\begin{aligned} C_{\text{awgn}} &= 25\text{ kHz} \cdot \log_2(1 + 10^{\frac{12.40\text{ dB}}{10}}) \\ &= 105\text{ kbits/s} \end{aligned} \tag{5.1}$$

This can also be expressed as capacity per hertz:

$$\begin{aligned} C_{\text{awgn}} &= \log_2(1 + 10^{\frac{12.40\text{ dB}}{10}}) \\ &= 4.2\text{ bits/s/Hz} \end{aligned} \tag{5.2}$$

Consequently, the total number of bits per second must be less than 10.19 bits/Hz and this is considered as the upper limit for our system.

6 Discussion

6.1 Bit Error Rate Requirements

As explained earlier, the requirements for the downlink bit error rate is 10^{-3}. In other words, there will be one error for every thousand transmitted bit. This limit can be justified as the information sent down to the ground station is not that important, hence some errors can be tolerated. This limit can be justified since the impact of some errors on the down link data is minimal.

6.2 Link Margin

To be able to decode with an error probability of 10^{-3}, a minimum of 9.8 dB SNR is needed according to Figure 13. The link margin is defined as the difference between SNR received and SNR required to demodulate. In this project it results in;

$$LM_{\text{down}} = 16.56\text{dB} - 9.8\text{dB} = 6.76\text{dB} \tag{6.1}$$

$$LM_{\text{up}} = 37.09\text{dB} - 9.8\text{dB} = 27.29\text{dB} \tag{6.2}$$

As seen in the following results, this corresponds to a link margin of 6.76 dB on the downlink and 27.29 dB on the uplink. This is very similar to link margins seen in other CubeSat projects. It should be noted that this is still calculated by using rough estimates and unfinished link budgets, and is therefore not entirely correct. Due to this, impacts of link budget variations will be explained when needed. The uplink seems to have a positive link budget where it can be argued that coding is unnecessary. This is, however, quite a risk due to the impacts of eventual faulty data on the uplink. On the downlink it is more obvious that some kind of error correcting is required. This is due to the relatively low SNR. Coding will be further discussed later.

Another important element worth considering, is that the received signal is greater than the minimum signal needed for the USRP and the WBX daughterboard. According to the given link margin that would not be a problem, as the margin on the downlink is greater than -130 dBm.

6.3 Automatic Repeat Requests

As previously mentioned, the satellite will only be visible from the ground station for about 10 minutes at a time. This means that the chosen transmission scheme should be able to transmit the data to the earth station in that time. Due to this time constraint on the downlink, ARQ could be the bottleneck causing insufficient data transfer. The implementation of ARQ in this system would also lead to other error sources. If the transmitter, a.k.a. the satellite, have to wait for a ”ack/nack” it would also be uplink dependant. I.e. if the satellite does not get the

”ack/nack” the ARQ would not matter. Therefore it would be wise to not use ARQ in the downlink as it increases the risks of not receiving all the data.

As for the uplink, there are other requirements for the reliability of the data transfer. Since this is a set with commands the satellite will execute, it must be decoded correctly. This can be done by either encoding the bitstream heavily or by implementing some type of ARQ. As mentioned earlier, the processing power on the satellite is limited. Therefore an error correcting code is limited in terms of processing. This makes implementing the much used concatenated coding scheme, described in Section 4.3.2 on page 20, problematic. As explained the Viterbi decoding algorithm will probably be too demanding to run on the MCU alongside all the other tasks. Because of this, an ARQ scheme should be implemented on the uplink, ensuring all the control command bits to be correct. It should be noted that the amount of data sent on the uplink is limited and a lot less than the data sent on the downlink. Hence, decoding of data on the uplink could possibly be somewhat time consuming. Even though the complexity of a Hybrid-ARQ is somewhat larger, it is worth considering. In this case, a coding with low rate should be used, and improved coding should be added for retransmission. This will reduce the needed retransmissions if the conditions are bad, improving the overall performance.

6.4 Coding

To decide on whether to use error correcting codes or not, the link budget along with the simulation results should be analyzed. Although this report focus on the use of Reed-Solomon coding and convolutional coding, there are other possibilities. For example the use of turbo codes is increasing in popularity. The reason for not going into details regarding turbo codes is complexity and time limitations. Even though the complexity of the encoder may be simple, the work of implementing this in the satellite radio would be very time consuming. To analyze the needed coding, this section will be divided into two parts, downlink and uplink.

Downlink:
As stated above, the link margin for the downlink is 6.76 dB. There are several ways this can be improved. This report will not focus on how i link budget can be improved in general, but how coding can have an impact. Two ways of coding have been explained, Reed-Solomon and concatenated Reed-Solomon and convolutional code. The main focus is (255,223) Reed-Solomon and is described in Section 4.3.2. This Reed-Solomon coding contains 22 redundant bits per 255 bits of information. Its block nature ensures up to 16 error corrections, even with burst errors in the system. The upsides of this coding scheme are many. Apart from the fact that it is already implemented in one of the satellites radio, it is used in many systems and are relatively easy to implement in LabVIEW. A possible LabVIEW implementation is shown in the LabVIEW code for this project. The simulation results indicate that the use of Reed-Solomon code will give a coding gain of around 4.2 dB. This is a rather helpful improvement of the link

budget and the equivalent link margin will be:

$$LM_{\text{down}} = 6.76\text{dB} + 4.2\text{dB} = 10.96\text{dB} \tag{6.3}$$

As mentioned, the link budget is rather rough. If the reality is worse than calculated, the need for error correcting coding is even more significant. In a worst case scenario, the Reed-Solomon will not be sufficient to provide the needed bit error rate. Then there is the possibility to add convolutional code on top of the Reed-Solomon.

Convolutional coding is explained in Section 4.3.2. This is a very common way to construct a concatenated code due to the Reed-Solomons performance and the convolutional codes way of interfacing with the real world. Even though it is a great way to ensure error correction in the system, it also includes a big downside. A convolutional code with a rate of 0.5 will double the bit length. This mean that twice as many bits as the Reed-Solomon only bitstream must be transmitted. Naturally, this means twice the time, which is not optimal in the NUTS project. Because the satellite will be in reach for about 10 minutes at a time, it is desirable to receive as much data as possible. That is of course if the link budget does not need the extra coding gain. From the results it is apparent that using convolutional coding with Reed-Solomon results in:

$$LM_{\text{down}} = 10.96\text{dB} + 2.2\text{dB} = 13.16\text{dB} \tag{6.4}$$

In other words, the coding gain from adding convolutional coding are not that significant. With the given link budget it is apparent the additional coding gain from the convolutional code is not needed. With complexity and time of development in mind, it therefore would be advisable to skip the convolution code since it is obsolete.

Uplink:
The uplink, on the other hand, has a very promising link budget. It must be noted that even a promising link budget does not guarantee error free transmission. Especially in space communication noise errors often come in burst and are caused by random events, such as solar storms. It has been many reported problems in CubeSats where the lack of error correcting coding on the uplink resulted in problems reaching out to the satellite. Due to this it would be advisable to put some sort of error correcting code on the uplink too. Convolutional code is, as explained in Section 4.3.2, not an option due to the Viterbi-algorithm in the decoder. Therefore the best option is a simple Reed-Solomon. If the Reed-Solomon is used in a Hybrid-ARQ scheme, the communication from the ground station to the satellite would be error free. The Reed-Solomon ensures correct information bits even if some amount of bit error is present, while ARQ requests retransmission if more bit errors are detected. One problem still remains. That is if the signal is so weak that the receiver does not detect it at all. This problem is harder to deal with, but according to the link budget estimations this should not cause any difficulties.

6.5 Comparing Protocols

Some different existing protocols have been described in the previous sections. To choose a protocol for NUTS, these different protocols will be compared. Important aspects of comparing protocols are:

- Redundancy - How many redundant bits/bytes is there if it is used in our system.
- Rate - What is the rate of the coding used.
- Implementation - Is it possible to implement on a MCU and ground station.
- Link budget - Will the given protocol be sufficient with the given link budget.

These aspects will be taken into considerations when the three different protocols are being compared. The protocol used for the uplink will not be discussed in detail since the data for the uplink is not known in detail at the moment. But coding for the uplink is already discussed and therefore the protocol for the uplink should be based on the coding discussion.

AX.25
When using protocols developed for other purposes, there will almost always be some redundancies. The AX.25 has a lot header information which the NUTS project does not need. This was the main reason for wanting another more efficient protocol. Higher redundancies implies less information bits sent. AX.25 is being used by radio amateurs all around the world, but it is sadly an old protocol more aimed to analog equipment. As explained it does not have any form of error correction and is therefore not optimal in this case. Although the implementation of this system is present in the current ground station, it is not well suited for this purpose. I.e. the use of AX.25 protocol is implemented by using amateur radio equipment, which is suboptimal. The protocol has been implemented in GNURadio this semester in a separate project, hence a implementation in the satellite is possible. There are numerous advantages by having the ground station implemented on digital hardware/software as the USRP and LabVIEW:

- Simple interfacing between input and data display.
- Save the whole spectrum to file so that it is possible to analyze later.
- More control over data packaging and more flexible system.

AAUSAT3
This protocol, see Section 4.4.4, is developed by "trial and error" by Aalbord University. Because this is the third CubeSat they have launched, their radio protocol has been improved along the way. AAUSAT3 has a rather extensive preamble, used for potential channel estimation. Obviously this results in high redundancy. The error correction applied corresponds

to the concatenated coding simulated in this project. This ensures a minimum number of errors, as discussed in previous sections, but results in a long bitstream. For our case it seems like these added bits are not needed, hence a waste of resources. The implementation of this protocol is somehow closer to the protocol implemented already, NGHAM, but it still would require some time consuming work. AAUSAT3 protocol would improve the link budget the most compared to the other two protocols due to the coding gain, which will correspond to the simulated coding gain in section 6.4. CSP bytes are included, which is a big advantage as this transport protocol will be implemented for internal communication within the satellite.

NGHAM
The relatively new protocol, NGHAM, is developed at NTNU. The protocol is explained in Section 4.4.3. 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 code is adjusted accordingly when the amount of bits are changed. All these properties are in general an advantage, but the main advantage for this protocol is that it is already implemented in the satellites' radio. This will minimize the implementation cost and this is very important for the NUTS project.

Of these three protocols, the AAUSAT3 and NGHAM protocols are the most promising ones. The redundancy is kept relatively low and the bits are protected by a error correcting protocol.

6.6 NUTS Protocol

In previous sections some existing protocols have been explored along with advantages and disadvantages. The protocol chosen for NUTS should satisfy the following:

- Keeping the redundancy low assuring fastest possible transmission time.
- Coping with channel induced errors, i.e. error correcting code or similar technique.
- Implementation time and complexity have a great impact on the choice of the protocol, as the satellite launch is getting closer.

Based on the previous discussion its clear that the NGHAM protocol fits the best. Especially because it is already implemented in the satellite VHF radio.

Preamble 4 bytes	Syncword 4 bytes	Size 3 bytes	Length 2 bytes	Encoded bits

Figure 14: NUTS Protocol

The given protocol is in accordance with the NGHAM protocol described in Section 4.4.3. Because the NUTS project is basing its internal transport on the CSP system, a CSP header must be added to the information bits. In the end this will reduce the amount of information bits available for data, but that should not be a problem. As argued earlier in the report, the error correcting Reed-Solomon code should be sufficient, given the link margin. Therefore the additional complexity for adding convolutional code will not pay off sufficiently. It should be noted that if the link margin is worse, the convolutional code might be a necessity.

7 Conclusion

LabVIEW was chosen as the preferred software platform. A bit error rate requirement of 10^{-3} was set on the downlink which leads to a link margin of 6.76 dB, while the uplink margin is 27.29 dB. However, it should be noted that the link budget calculations may have some deviations and therefore a reevaluation is most likely necessary as more link parameters become available. It was decided to use Hybrid ARQ combined with Reed-Solomon on the uplink, this due to the importance of receiving the commands from the ground station without errors. The downlink, on the other hand, will only have Reed-Solomon coding as additional coding seems unnecessary regarding the error rate requirements set for this project. A communication system was simulated in LabVIEW with BFSK modulation, an AWGN channel and the options of no coding, Reed-Solomon coding or Reed-Solomon concatenated with convolutional coding. There was a coding gain of about 4.2 dB for the Reed-Solomon alone and an additional 2.2 dB when convolutional coding was added. After evaluating three different protocols, NGHAM, AX.25 and AAUSAT, the best option for this project with the available information is the NGHAM protocol. This is due to the protocol using Reed-Solomon code, which was proven sufficient, and it is already implemented in the VHF radio on the satellite, which saves time.

8 Future Work

This report have studied different protocols to be used in the NUTS CubeSat program. What could be carried out next is exploring the frequency offset impacts of the system. This can be done by the developed LabVIEW communication system found in this report. There should also be implemented frequency tracking, due to the varying Doppler shift and its inconsistent nature. For the possibility to implement coding on the uplink, the decoding algorithms should be explored on the microcontroller. This is typically done with a benchmark test to test the overall performance of the decoding algorithm. Hybrid-ARQ should be further explored, especially regarding the impacts of decoding time. The simulation should be moved from software only to a hardware platform, i.e. LabVIEW and a USRP. Both the transmitter and the receiver should be implemented on this platform, allowing for further testing. These tests could initially be conducted in a controlled environment, but it would be interesting to test it in the air, for example on Andøya with a flying balloon. The system could also be tested using the developed radio, as this is closer to the end system. As the initial communication will be some sort of morse code/ beacon, this should also be implemented on an USRP and sufficiently tested.

References

[1] AAUSAT3. *AAUSAT3 HAM Radio Info*. http://www.space.aau.dk/aausat3/index.php?n=Ham.HAMAndRadioInfo. [Online; accessed 28-october-2014].

[2] Anil K. Maini & Varsha Agrawal. *Satellite Technolody: Principles and Applications*. 2nd ed. John Wiley & Sons, Inc, 2011.

[3] *Aircell 7 Datasheet*. http://www.ssb.de/pdfs/6070_Aircell%207_en.pdf. [Online; accessed: 09-december-2014].

[4] Constantine A. Balanis. *Antennea Theory: Analysis and design*. 3rd ed. John Wiley & Sons, Inc, 2005.

[5] Herbert J. Kramer & Arthur P. Cracknell. *International Journal of Remote Sensing: An overview of small satellites in remote sensing*. 1st ed. Taylor & Francis, 2008.

[6] The CubeSatProgram. *CubeSat Design Specification*. http://333.cubesat.org/images/developers/cds_rev13_final.pdf. [Online; accessed 14-september-2014]. 2014.

[7] Brenda Lidia Escobar. *Link Budget for NTNU Test Satellite*. http://daim.idi.ntnu.no/masteroppgaver/010/10191/masteroppgave.pdf. [Online, accessed: 09-december-2014]. 2014.

[8] Simon Haykin. *Communication Systems*. 4th ed. John Wiley & Sons, Inc, 2000.

[9] A. Houghton. *Error Coding for Engineers*. 1st ed. Springer Science+Business Media, 2001.

[10] Louis J. Ippolito. *Satellite Communicaiton Systems Engineering*. 2nd ed. John Wiley & Sons, Inc, 2008.

[11] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. 11th ed. Cambridge University Press, 2012.

[12] Sigvald Marholm. *Antenna systems for NUTS*. http://daim.idi.ntnu.no/masteroppgaver/008/8238/masteroppgave.pdf. [Online, accessed: 25-november-2014]. 2014.

[13] Theodore S. Rappaport. *Wireless Communications: Principles and practice*. 2nd ed. Prentice Hall PTR, 2002.

[14] Alejandro A. N-Zavala & Simon R. Saunders. *Antennas and Propagation for Wireless Communication*. 2nd ed. John Wiley & Sons, Inc, 2007.

[15] *Sensitivity measurements*. http://wiki.oz9aec.net/index.php/USRP_Reference. [Online; accessed: 01-december-2014].

[16] William A. Beech & Douglas E. Nielsen & Jack Taylor. *AX.25 Link Access Protocol for Amateur Packet Radio*. https://www.tapr.org/pdf/AX25.2.2.pdf. [Online; accessed 20-september-2014].

[17] *VHF Baseantenner - Permo*. http://www.permo.no/A/VHF%20baseantenner.htm. [Online; accessed: 09-december-2014].

[18] *WBX Ettus Research*. http://www.ettus.com/product/details/WBX. [Online; accessed: 09-december-2014].

A BER plot for BFSK

```
% Plots theoretical BER for BFSK
clear all;
close all;

SNRdB=0:0.1:13;
SNR=10.^(SNRdB/10);

BER_th=(1/2)*erfc(sqrt(SNR/2));

semilogy(SNRdB,BER_th, 'LineWidth', 2)
legend('Theoretical BFSK')
axis([min(SNRdB) max(SNRdB) 10^(-5) 1]);
grid on
ylabel('Bit error probability')
xlabel('SNR [dB]')
```

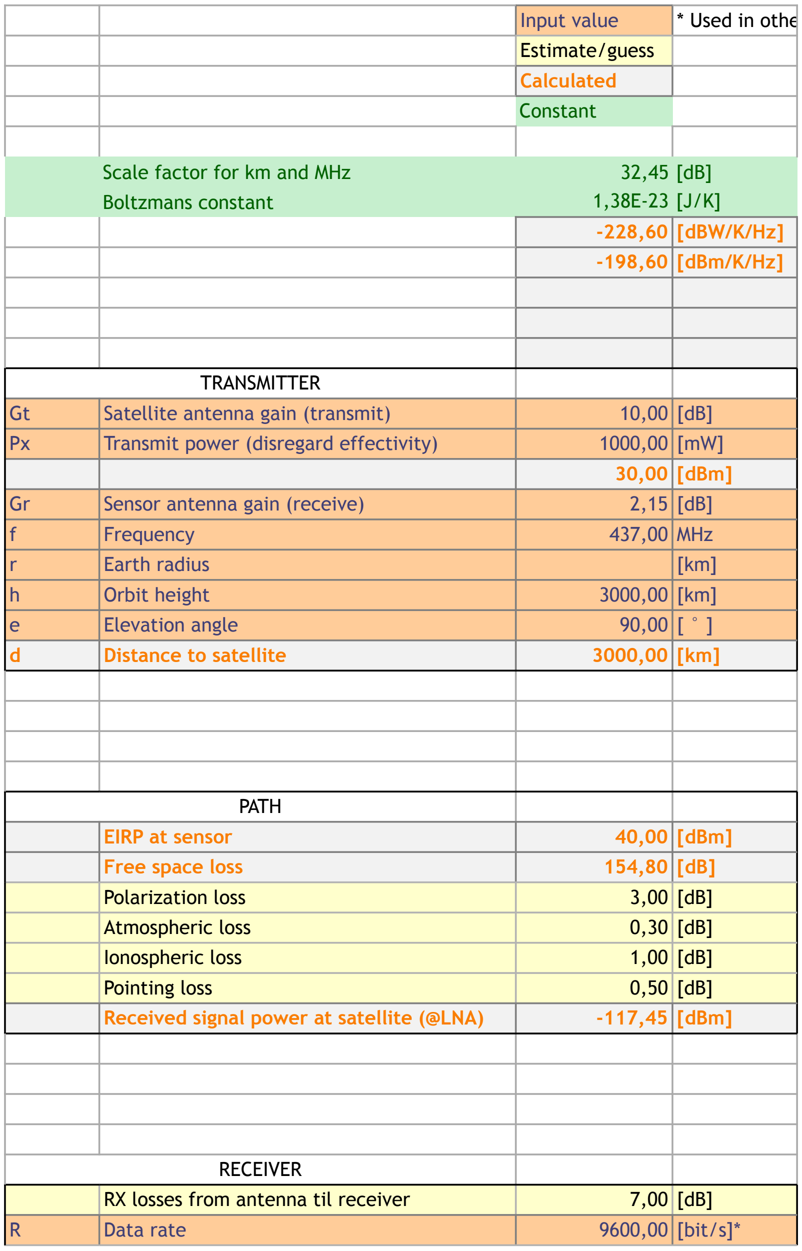
../matlab/bfsk_ber.m

B Link Budget 1

		Input value	* Used in othe
		Estimate/guess	
		Calculated	
		Constant	
	Scale factor for km and MHz	32,45	[dB]
	Boltzmans constant	1,38E-23	[J/K]
		-228,60	[dBW/K/Hz]
		-198,60	[dBm/K/Hz]
TRANSMITTER			
Gt	Sensor antenna gain (transmit)	2,15	[dB]
Px	Transmit power (disregard effectivity)	500,00	[mW]
		26,99	[dBm]
Gr	Satellite antenna gain (receive)	12,00	[dB]
f	Frequency	145,00	MHz
r	Earth radius		[km]
h	Orbit height	600,00	[km]
e	Elevation angle	90,00	[°]
d	Distance to satellite	600,00	[km]
PATH			
	EIRP at sensor	29,14	[dBm]
	Free space loss	131,24	[dB]
	Polarization loss	3,00	[dB]
	Atmospheric loss	0,30	[dB]
	Ionospheric loss	1,00	[dB]
	Pointing loss	0,50	[dB]
	Received signal power at satellite (@LNA)	-94,90	[dBm]
RECEIVER			
	RX losses from antenna til receiver	7,00	[dB]
R	Data rate	500,00	[bit/s]*
T	System temperature (at receiver)	400,00	[K]

B	Bandwidth	10000,00	[Hz]
N	Noise power	-132,58	[dBm]
S	Power at receiver	-101,90	[dBm]
	LINK QUALITY		
	Signal to noise ratio	30,68	[dB]
	Eb/N0 for link	43,69	[dB]
	Needed Eb/N0 for BPSK	7,00	[dB]
	LINK MARGIN	36,69	[dB]

C Link Budget 2



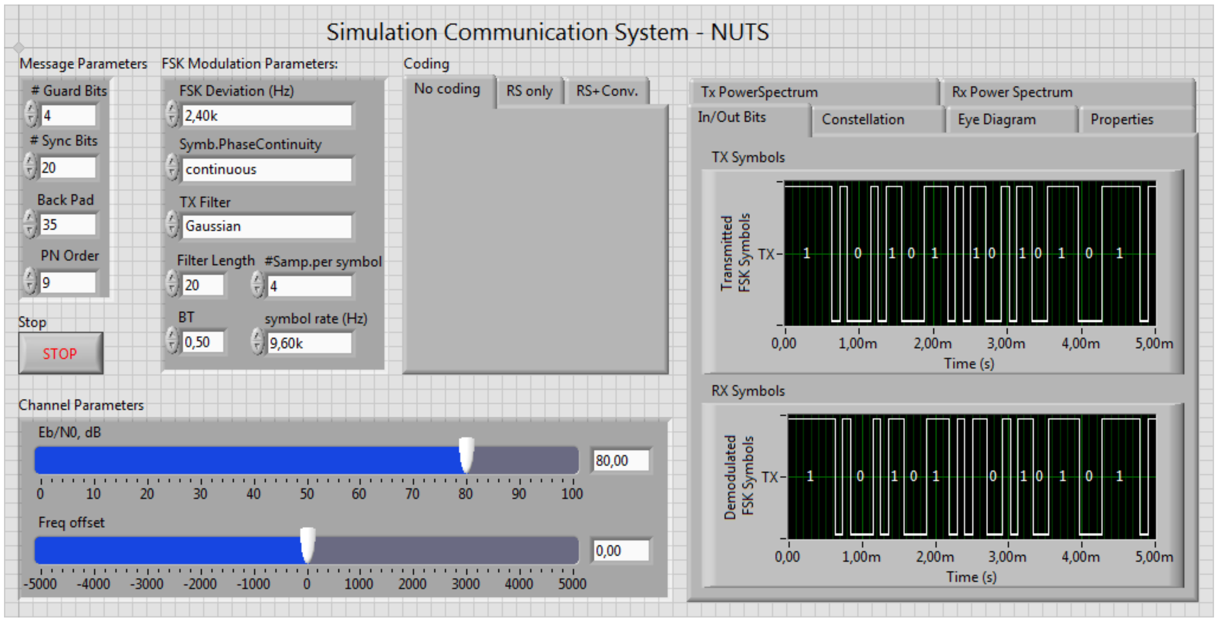
		Input value	* Used in othe
		Estimate/guess	
		Calculated	
		Constant	
	Scale factor for km and MHz	32,45	[dB]
	Boltzmans constant	1,38E-23	[J/K]
		-228,60	[dBW/K/Hz]
		-198,60	[dBm/K/Hz]
TRANSMITTER			
Gt	Satellite antenna gain (transmit)	10,00	[dB]
Px	Transmit power (disregard effectivity)	1000,00	[mW]
		30,00	[dBm]
Gr	Sensor antenna gain (receive)	2,15	[dB]
f	Frequency	437,00	MHz
r	Earth radius		[km]
h	Orbit height	3000,00	[km]
e	Elevation angle	90,00	[°]
d	Distance to satellite	3000,00	[km]
PATH			
	EIRP at sensor	40,00	[dBm]
	Free space loss	154,80	[dB]
	Polarization loss	3,00	[dB]
	Atmospheric loss	0,30	[dB]
	Ionospheric loss	1,00	[dB]
	Pointing loss	0,50	[dB]
	Received signal power at satellite (@LNA)	-117,45	[dBm]
RECEIVER			
	RX losses from antenna til receiver	7,00	[dB]
R	Data rate	9600,00	[bit/s]*

T	System temperature (at receiver)	273,00	[K]
B	Bandwidth	25000,00	[Hz]
N	Noise power	-130,26	[dBm]
S	Power at receiver	-124,45	[dBm]
	LINK QUALITY		
	Signal to noise ratio	5,81	[dB]
	Eb/N0 for link	9,96	[dB]
	Needed Eb/N0 for BPSK	7,00	[dB]
	LINK MARGIN	2,96	[dB]

D LabVIEW Simulation Code

The LabVIEW code for this project can be found at this github repo:
`https://github.com/storsnik/groundstation-labview`

E LabVIEW front panel



Simulation Communication System - NUTS

Message Parameters

Guard Bits 4

Sync Bits 20

Back Pad 35

PN Order 9

Stop STOP

FSK Modulation Parameters:

FSK Deviation (Hz) 2,40k

Symb.PhaseContinuity continuous

TX Filter Gaussian

Filter Length 20

#Samp.per symbol 4

BT 0,50

symbol rate (Hz) 9,60k

Coding: No coding | RS only | RS+Conv.

Channel Parameters

Eb/N0, dB 80,00

Freq offset 0,00

Figure 15: LabVIEW front panel

F LabVIEW Transmitter

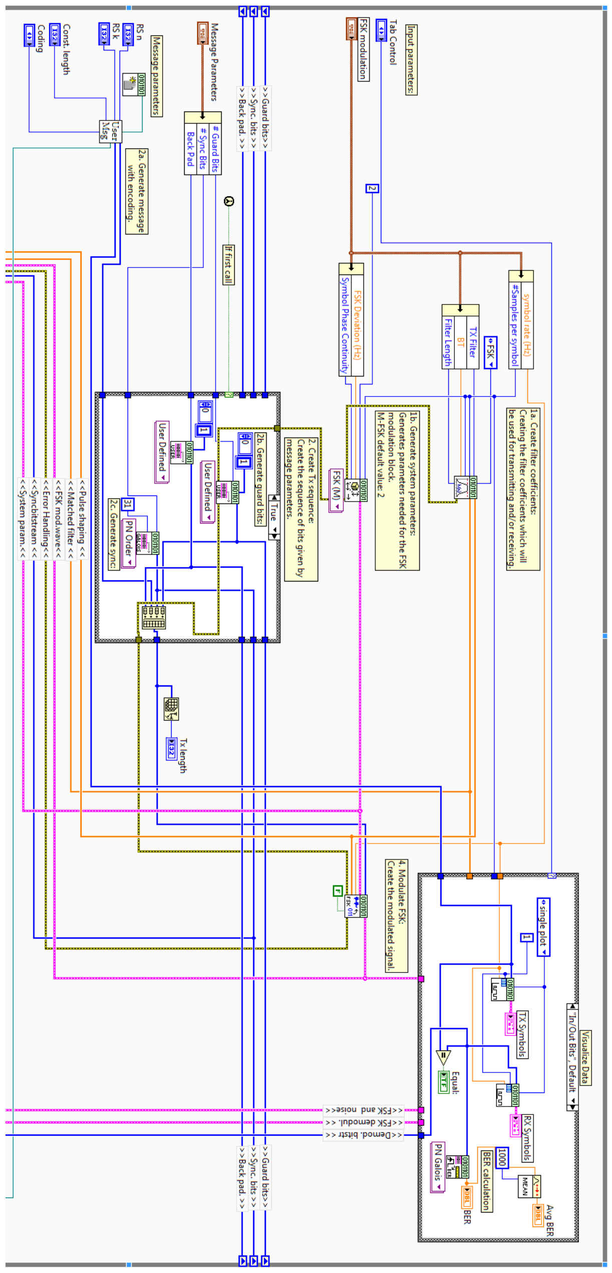


Figure 16: LabVIEW Transmitter Code

G LabVIEW Receiver

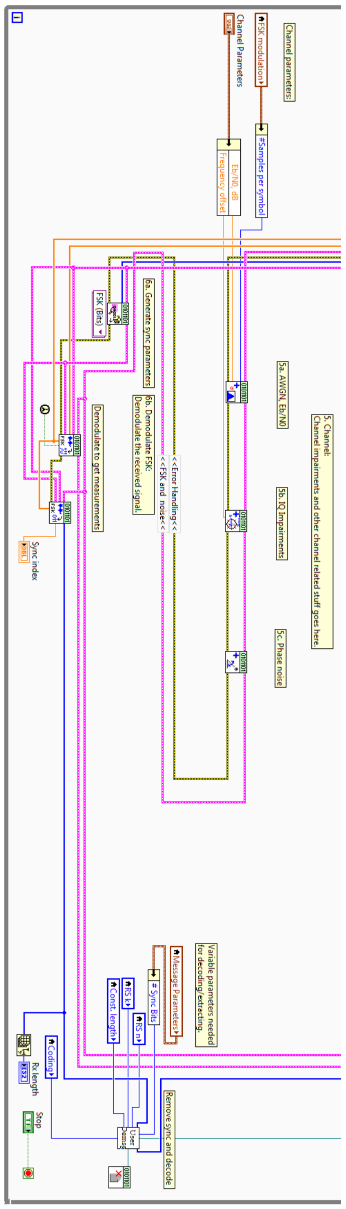


Figure 17: LabVIEW Receiver Code

H Reed-Solomon matlab simulation

```
close all;
clear all;

BER_rs = [];
BER_hdd = [];
snr_db = 0:0.5:4;
snr_lin = 10.^(snr_db/10);
N = 10000;
cnt = 1;

n = 255;
k = 223;

m = log2(n+1);

disp('Starting simulation - Wait for time estimate..')
for SNR = snr_db
    tic;

    N0 = 10.^(-SNR/10);

    % Generate row data
    data = randi(2^m-1,[N,k]);

    % Create message Galois field
    msg = gf(data, m);

    % Encode with Reed-Solomon
    enc = rsenc(msg, n, k);

    % Convert to bitstream
    encoded = double(enc.x);
    for i=1:N
        %prod(size(encoded))
        %size(encoded)
        %transpose(de2bi(encoded(i,:),'left-msb',m))
        b(i,:) = reshape(transpose(de2bi(encoded(i,:),'left-msb',m)),[1,m*n]);
    end

    % Create sent bitstream
    s = (2*b-1);

    % Add AWGN
    y = awgn(s,SNR);
    %noise = kron(ones(N,n*m),sqrt(N0/2)).*(randn(N,n*m));
    %y = s + noise;

    % Decission making
    bhat = 0.5*(sign(real(y))+ 1);

    % BER after HDD
    bit_errors = double(N*k*m - sum(sum(bhat(:,[1:k*m]) == b(:,[1:k*m]))));
    BER_hdd = [BER_hdd bit_errors/(N*k*m)];

    % Convert to integers
    for i=1:N
        enchat(i,:) = transpose(bi2de(transpose(reshape(bhat(i,:), [m,n])),'left-msb
            '));
    end
    % Decode the Reed-Solomon
    enchat_gf = gf(enchat,m);
    dec = rsdec(enchat_gf, n, k);
    decoded = double(dec.x);
    for i=1:N
        b_dec(i,:) = reshape(transpose(de2bi(decoded(i,:),'left-msb',m)),[1,m*k]);
    end

    % Count errors after decoding and calculate BER
    bit_errors = double(N*k*m - sum(sum(b_dec == b(:,[1:k*m]))));
    BER_rs = [BER_rs bit_errors/(N*k*m)];

    time = toc*(length(snr_db)-cnt);
    cnt = cnt + 1;
    disp(sprintf('Estimated %5.4f seconds remaining..',time))
end

disp('Simulation complete! Plotting data..')

% Plot results
figure (1);
semilogy(snr_db,qfunc(sqrt(snr_lin)), 'b--', 'LineWidth',1.5);
hold on
semilogy(snr_db, BER_hdd, 'g-o', 'LineWidth', 1.5);
semilogy(snr_db, BER_rs, 'r-o', 'LineWidth', 1.5);
title('BER Coding');
xlabel('SNR [dB]');
ylabel('BER');
legend('Theoretical AWGN','After HDD','RS simulation')
grid
axis tight
```

../matlab/reedsolomon.m