NTNU
Norwegian University of
Science and Technology

NUTS: Ground station based on software defined radio

André Løfaldli

Submission date: December 2015
Supervisor: Roger Birkeland
Responsible professor: Torbjörn Ekman

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Statement

The ground station is one of the principal components of the satellite, it is through this the satellite is controlled when in orbit.

In order to support a packet radio link between the satellite and ground station, it is desired that the ground station should be built around a software defined radio (SDR) platform, such as the Ettus Research USRP.

This will be a more flexible and complete installation compared to the classical approach, employing HAM radio equipment. This equipment is not ideal for packet radio transmission. In order to more easily be able to implement other modulations and link protocols, a set up around an SDR platform is desired.

Key tasks for the student:

- Revise top-level design of ground station based on an USRP SDR
- Simulate and evaluate the current implementation of the NGHAM protocol
- Enable modulation and demodulation of a GMSK signal
  - Frequency sync (alignment) between OWL and USRP
  - Implement GMSK demodulation
- Demonstrate end-to-end communication between the USRP and an OWL radio

In addition to the given tasks, the student is expected to participate in relevant group work. The NUTS project is a multidisciplinary project, which requires more involvement from the student than just the completion of the individual task and report.

Abstract

This report is written for a final year project at the Norwegian University of Science and Technology (NTNU). It is part of the NTNU Test Satellite project, where a double CubeSat is in development.

The purpose of the report is to implement a ground station based on software defined radio and GNU Radio. This type of ground station is believed to offer more flexibility than more traditional ones.

With the ultimate goal of achieving end-to-end communication between the ground station and the hardware that will go onto the space segment. This requires a modem capable of GMSK modulation and the ability to encode and decode data using the data link layer protocol NGHAM. For this, the build in GMSK modem is used with a specially implemented NGHAM interface.

After testing the system inside the lab, at close range, the system has proven a satisfactory performance. Transmission between the ground station and the OWL showed only a small ($< 1\%$) amount of data loss.

All source code is available in the GitHub repository `gr-nuts`[1].

Contents

Chapter 1

Introduction

The work described in this report is part of the NTNU Test Satellite project, or NUTS for short. It is a student driven project at the Norwegian University of Science and Technology (NTNU). The main goal is to develop, and eventually launch, a double CubeSat into a Low Earth Orbit. A single unit CubeSat is $10\,\text{cm} \times 10\,\text{cm} \times 10\,\text{cm}$, meaning NUTS will have twice this size. The project has been running since 2010 and is part of the national Norwegian Student Satellite Program (ANSAT)[2]. Currently, the goal is to have a complete engineering model by august 2016 and launch sometime in 2017.

The space segment of the satellite mission will consist of several subsystems, all of which are designed by students. They are:

- communication
- electrical power supply
- attitude determination and control
- antenna
- mechanical structure
- on board computer

These subsystems are vertically mounted on a common circuit board, called the backplane. Several iterations has been made over the years, and it is still under development. In a addition to holding the subsystems together, it distributes power and enables communication between them.

To allow for full duplex transmission, the communication subsystem is made up to two separate modules operating on different frequencies. The VHF subsystem is a complete design by a previous student. It is a 2 m band amateur radio transceiver called OWL[3]. The UHF subsystem is under development and will be based on a Analog Devices transceiver, for the 70 cm band[4]. The operating frequencies, allocated by the Radio Amateur Satellite Corporation (AMSAT), are 145.98 MHz and 437.305 MHz respectively[5].

To communicate with the space segment, a ground station is needed. This report will discuss the original design of the ground station, based on an ICOM amateur radio transceiver[6]. Then the current design, based on Ettus Research USRP [7][8] and GNU Radio[9] will be presented. For this, a data link layer protocol has been implemented to demonstrate communication with the OWL VHF module.

This report starts, in Chapter 2, by discussing some core concepts about the ground station and the communication system. Then, in Chapter 3, an evaluation of the work previously done, along with the new implementations made. Chapter 4 contains descriptions of the various test setups, with the results discussed in Chapter 5. Finally, Chapter 6 is the conclusion, followed by some proposed continuations of the project in Chapter 7

The tests performed demonstrate the behaviour of the system, which includes the ability to send and receive data between the ground station and the OWL VHF. The results are considered satisfactory, as end-to-end communication was successful.

Chapter 2

Background Theory

2.1 The Ground Station

The ground station is required for communicating with the satellite once it is in orbit. Its main tasks include sending commands and receiving telemetry and payload data.

The original design was based around an ICOM 9100 transceiver[6]. It included streaming demodulated data to the audio card on the computer for further processing. In certain cases, it is useful to have the ability to store the raw data for later. One example is if there is a mismatch between the frequencies of the local oscillator and the satellite.

The current design is based on a *Universal Software Radio Peripheral (USRP)* developed by Ettus Research[8]. This is a software defined radio (SDR) with a motherboard built on an FPGA, with ADCs and DACs, which does the conversion between real analog signals and digital baseband representations. It has a modular frontend, known as a daughterboard, which performs operations such as filtering and up/down conversion. The device used in this report is a USRP2[1] with a WBX[10] daughterboard capable of serving frequencies between 50 MHz and 2.2 GHz. It is connected to a computer using Gigabit ethernet, transferring sample data as complex 32-bit floating point numbers.

To be able to receive signals from orbit, a pair of crossed yagi antennas are used, one for each frequency band. They each have corresponding power amplifiers (PA) and low noise amplifiers (LNA). The antenna array have motors, enabling it to point in various directions and track a satellite during a pass[11][12].

[1] The USRP2 is no longer for sale, and has been replaced by the NI B200[8].

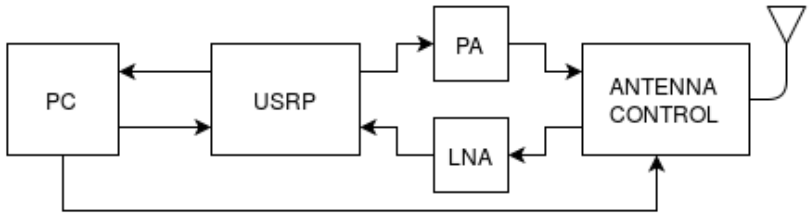

Figure 2.1: Block diagram of the ground station

This report only describes communication in the lab at close range, so a simple linear antenna is used with no amplifiers.

2.2 Signals

In the NUTS communication system, data is transferred using a link layer protocol developed by a student. The protocol is called Next Generation HAM (NGHAM), and is based on the popular amateur radio protocol AX.25. It defines an 11 bytes long header and Forward Error Correction (FEC) codeword which contains the payload and may be of 7 different sizes. Table 2.1 shows the relation between the payload length and the resulting codeword length. If the payload length does not match any in the table it is padded with zeroes until a valid length is reached[13].

PAYLOAD LENGTH	TOTAL PACKET LENGTH
28	47
60	79
92	111
124	159
156	191
188	223
220	255

Table 2.1: Relation between payload lengths and total packet length

The header consists of a 4 bytes long preamble with many transitions, which improves the clock recovery performance. Following is a 4 bytes long sync word with good auto correlation properties (see Appendix B, improving the data alignment. The 7 different packet lengths are indicated by a 3 byte long size tag, separated by a large hamming distance.

The codeword includes Cyclic Redundancy Check[14], Reed Solomon encoding[15]

is whitened using a scrambler[16]. The structure of an NGHAM packet is shown in Figure 2.2

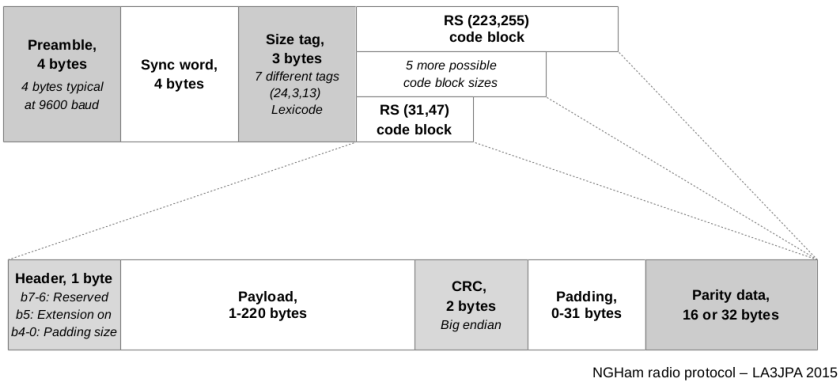

Figure 2.2: The structure of an NGHAM packet[13]

The modulation used is *Gaussian Minimum Shift Keying (GMSK)*, which is also used by the Global System for Mobile communication (GSM) network and the Automatic Identification Service (AIS) for maritime navigation[17].

GMSK a special case of Minimum Shift Keying (MSK) where the symbols are shaped using a Gaussian filter. This gives the signal the following properties[18]:

- Constant envelope
- Relatively narrow bandwidth
- Coherent detection performance equivalent to that of QPSK

And because of the filtering of the symbols, the bandwidth is reduced further[18].

It should be noted that NGHAM supports other forms of modulation, which are already implemented in the OWL[13].

2.3 GNU Radio

The software suite GNU Radio is used to process the data transferred between the computer and the USRP. It is an open source toolkit with hardware drivers and signal processing tools, and has been under constant development since 2001. The framework it provides can be used to create runnable applications, known as a *flow graph*[19].

The flow graphs are usually written in Python and consist of separate modules, known as *blocks*, processing the signals. To optimize the throughput of the flow-graph a runtime scheduler is used to control the rate at which data is passed between blocks. This is to avoid bottlenecks, and input or output buffers filling up with data[20].

Data may be passed between blocks in two forms, either as a continous stream or as an asynchronous message. The streams may contain various data types, including integers, floating point numbers and complex numbers[21]. For a complete list see Appendix A

Each block tells the scheduler the ratio of its number of input items N versus its number of output items M. While a block may have any values for M and N there are three common types[22]:

- synchronous blocks, $M = N$
- interpolating blocks, $M < N$
- decimating blocks, $M > N$

In addition there are source and sink blocks, having either M or N set to 0. Examples of these are the USRP Hardware Driver (UHD) blocks, used for interfacing with the USRP.

To process the data flowing through a block the `work()` function is called. The amount of data it will process for each call is set by the scheduler, but the block may set a required amount in the `forecast()` function. This is useful eg. in a filter.

This means that when designing a packet based communication system, there is no way of knowing the boundaries of a packet. To deal with this the tagged stream block was added. It uses meta data attached to the data stream to indicate when a packet starts and when it ends. Its `forecast()` changes the size of the input buffer so that it only contains the samples from a single packet. In Figure 2.3 an example of both cases is shown.

The tagged stream block may only be used when the packets are back to back or when there is a fixed time slot between them. This is because they require all data passing through to be accounted for by a tag. This makes them inappropriate for use in a communication system with no set interval for when a packet arrives.

For the packet encoder block the tagged stream block is a good choice because it can be connected to a correctly tagged stream. In the case of the decoder another approach had to be used.

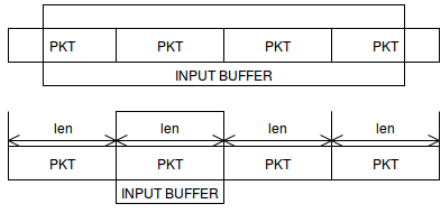

Figure 2.3: An illustration of how the general block and the tagged stream block loads data into its input buffer

Chapter 3

Methods

3.1 Previous work

In the past, some attempts has been made to implement the NGHAM encoder and decoder blocks in GNU Radio. This was started by a master student[23] and continued by another student. In this section the most complete version of the code will be evaluated[1].

It features an implementation of the NGHAM encoder block and the NGHAM decoder block. They are both implemented as synchronous blocks, meaning there is an equal amount of data flowing in and out. This causes problems for the scheduler when running because the overhead added by NGHAM means that by encoding the payload data, redundancy is added, and thus increasing the amount of data flowing.

Another problem is that because they are not using tagged streams, there are no ways of knowing the size of the payload.

Although the implementation is not usable for its intended purpose, it is able to correctly encode and decode the data, under certain circumstances. If the flow graph is only set to run with a limited amount of data flowing into the blocks, they produce the correct output. However, the flow graph is required to run contiunously with an unknown amount of data flowing through the block, so a new implementation is necessary. This will be discussed in Section 3.3

[1] No documentation available, but a software repository exists

3.2 GMSK modem

To communicate successfully with the OWL VHF module, a GMSK modem is required. GNU Radio provides two blocks for this, a modulator and a demodulator. These are implemented with Python using a number of other blocks. A flow graph version of these are shown in Figures 3.1 and 3.2, all values used are the default parameters defined by GNU Radio.

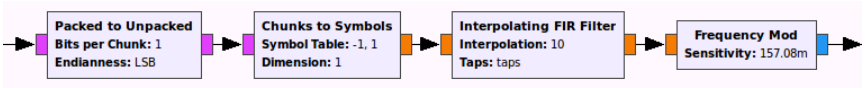

Figure 3.1: The internals of the GMSK Mod block

According to [24] there are two methods to generate a GMSK signal, one is through Frequency Shift Keying (FSK) and the other is Quadrature Phase Shift Keying (QPSK). The GMSK modulator block is based on the former. It takes packed bytes as input, which means all 8 bits are used. These are unpacked by placing the bits at LSB position of 8 consecutive bytes. The unpacked bytes are then converted to NRZ, and then complex GMSK symbols using a Gaussian filter and an FM modulator block.

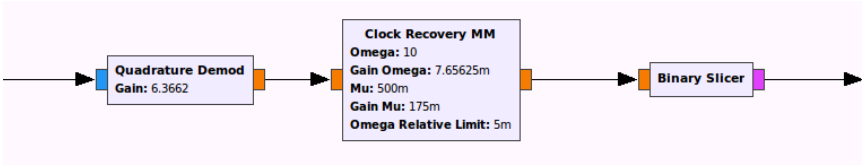

Figure 3.2: The internals of the GMSK Demod block

The demodulator takes in complex GMSK symbols and converts them to real symbols using a quadrature demodulator block. To synchronize the sampling times, the clock recovery block is used. It is implemented based on an optimized version of the Mueller and Müller algorithm for timing error detection[25]. The resulting NRZ samples are sliced at 0 to produce a stream of unpacked bytes.

A simple demonstration will be given in Section 4.1.

3.3 Implementing NGHAM blocks

When implementing the NGHAM encoder and decoder as GNU Radio blocks, two different methods are used. All the internal code in the blocks is based on the original NGHAM code[13]. The complete source can be seen in [1]

Encoder The encoder is implemented as a tagged stream block, where the length of the payload is attached to the incoming data stream. This lets the block determine the length of the resulting packet length. Table 2.1 show the values packet length that various payload lengths will produce. The numbers are taken from the NGHAM specification[13].

First the encoder calculates how many zeroes of padding is needed to match the smallest valid payload length. The padding length is stored in the first byte of the codeword. The next bytes are the payload data, which is then used to calculate two bytes of CRC[14]. The CRC bytes are then added before the padding. The resulting vector is used to calculate parity data using an open source FEC library[26]. The parity data may be of length 16 or 32 bytes, depending on the payload length, and is added after the padding.

The complete codeword is then scrambled to reduce the likelihood of long sequences of `1`'s or `0`'s appearing, and thus increasing the timing recovery properties[16]. The codeword then is passed on, preceded by the header containing the appropriate size tag.

Decoder When implementing the decoder an attempt was made to use the same approach as with the encoder. As discussed in Section 2.3 there is no way of knowing how much time will pass between each received packet. Instead it is implemented as a sink, meaning it has no stream on its output, which posts the data of the decoded packets to an asynchronous message queue.

To detect an incoming packet, the decoder loads a single bit at a time into a register. Every new bit is put at the LSB position, while the others are shifted to the left. The register is then correlated against the NGHAM sync word using an xor operation, and then counting the number of different bits using a function from the VOLK library[27].

```
d_data_reg = (d_data_reg << 1) | ((in[count++]) & 0x01);
wrong_bits = d_data_reg ^ sync_word;
volk_32u_popcnt(&nwrong, wrong_bits);
```

If the number of wrong bits are less than a set threshold, the next three bytes are loaded to search for a matching size tag using the same method.

Once the packet size is known the reverse of the procedure in the encoder is performed to decode the message. A state machine representation is included in Figure 3.3.

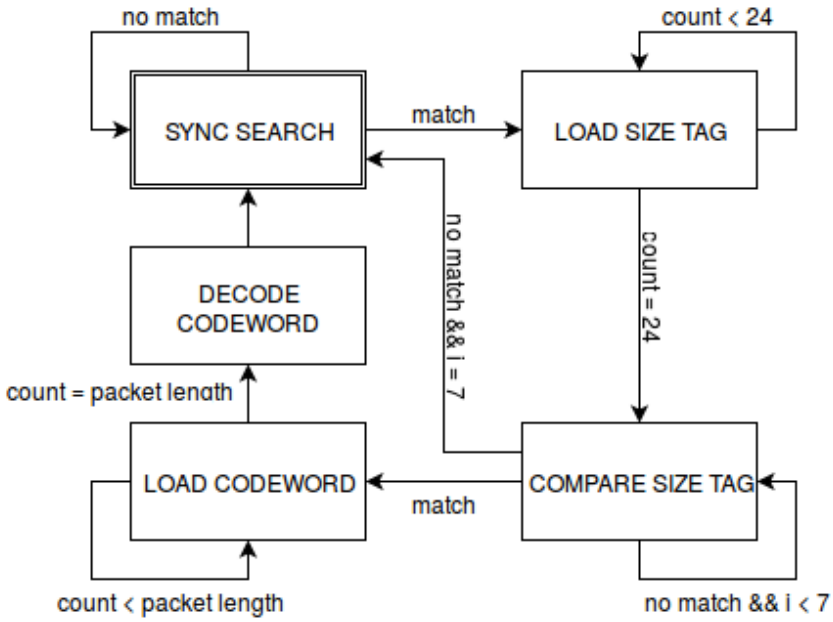

Figure 3.3: A state machine representation of the NGHAM decoder

Chapter 4

Testing and verification

This chapter will discuss the process of verifying the functionality of the implementations previously presented.

First a test of the GMSK modem in GNU Radio is described, where the requirements are that the modulated data may be demodulated correctly. Then the new NGHAM blocks are tested to see if the encoded packets can be decoded and the containing data successfully extracted.

4.1 GMSK modulation and demodulation

To verify the function of the GNU Radio GMSK modem a simple test is set up. In Figure 4.1 is a flow graph that modulates a vector with an easily recognisable pattern.

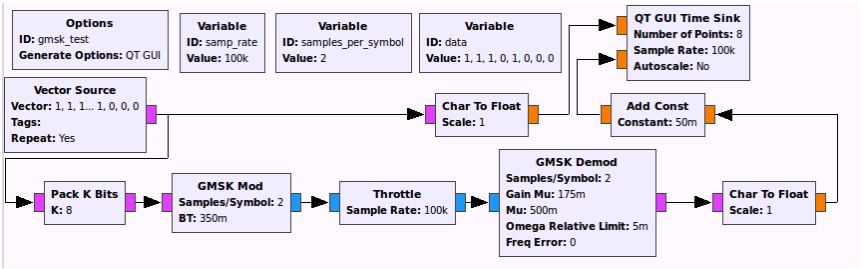

Figure 4.1: GRC flow graph for verifying GMSK

The vector source block outputs a repeating series of unpacked bytes. These are

are packed in the Pack K bits block with $K = 8$, before it is modulated in the GMSK Mod block. The throttle block is used to limit the rate at which the data is flowing[1]. To be able to plot the original vector and the demodulated version, they are converted into floating point values. The demodulated data have a small value added so that the lines don't cover each other when plotted. These are then plotted using the time sink block.

To verify the behaviour of the clock recovery block isolated, another flow graph is created. A random source creates a stream of packed bytes which are modulated with GMSK. Then a little bit of noise is added to create a more realistic signal. After passing through the throttle block, two versions of the constellation is plotted using the constellation sink. The flow graph used is shown in Figure 4.2

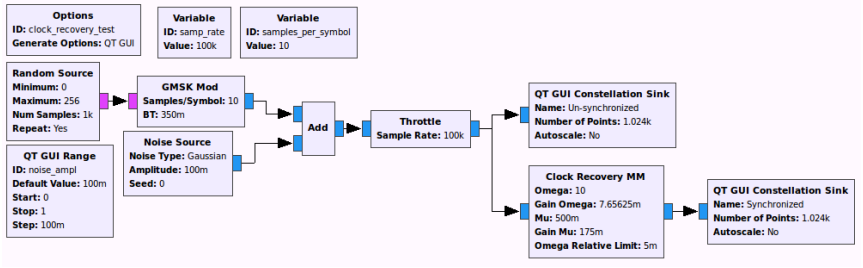

Figure 4.2: GRC flow graph for verifying clock recovery

4.2 NGHAM encoding and decoding

To perform a self test of the GNU Radio NGHAM blocks, a loopback flow graph is set up. The encoder receives a test message in the form of a tagged stream, which it encodes sends to the decoder. An example of how this may be tested is shown in Figure 4.3

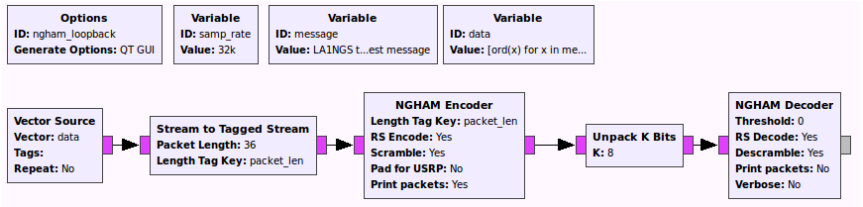

Figure 4.3: A flow graph showing the NGHAM loopback setup

[1] Without any hardware blocks controlling the data rate, the GNU Radio scheduler will try to run the flow graph as fast as possible. This may take up close to 100% of CPU power and might not be desirable.

If everyting is done correctly the decoded data should have the same content as the original. This is checked by comparing the output printed to `stdout` when both the encoder and decoder have the "Print packets" option enabled.

4.3 Communication between OWL and USRP

To communicate with the OWL VHF transceiver two flow graphs are set up. One for transmitting NGHAM packets using the encoder, and another for receiving using the NGHAM decoder.

In the transmitter flow graph, shown in Figure 4.4, the variable `message` is defined to contain the call sign of the ground station, LA1NGS, along with a test message. The message is converted from ASCII characters to their byte representation and stored in the `data` variable, which is used as parameter for the vector source block. The vector source block also attaches a tag containing the packet length before passing it on to the NGHAM encoder block. After the data is modulated, the signal is up-converted to a higher sampling rate using the rational resampler block. Finally the signal is sent to the UHD: USRP sink block, which is controlling the USRP and transmitting the signal over the air.

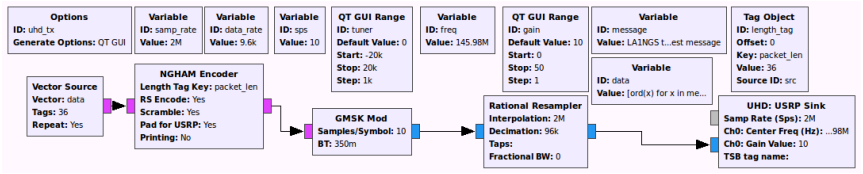

Figure 4.4: GRC flow graph for transmitting NGHAM packets

The RX flow graph works in the opposite way, it is shown in Figure 4.5. The UHD: USRP source block receives the incoming signal from the USRP, which are filtered using the frequency translating filter block. In addition to moving the center frequency of the signal, it reduces the sampling rate and removes out-of-band noise. The shape of the filter is set by the `taps` variable, making it a low pass filter.

After the signal is filtered, it is re-sampled further to get the correct symbol rate before demodulation. The demodulated signal is sent to the NGHAM decoder which may print the decoded data to `stdout` or post it to a message queue.

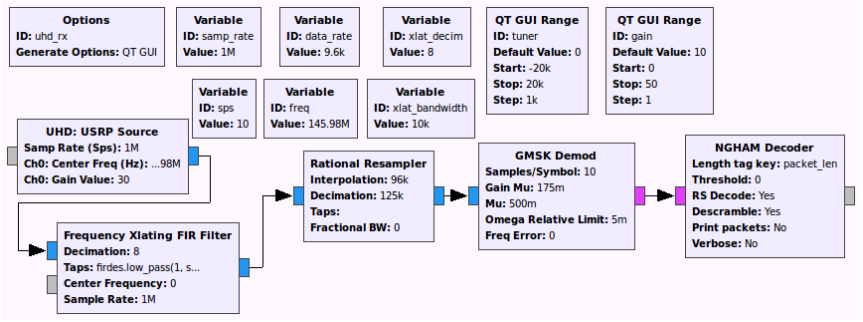

Figure 4.5: GRC flow graph for receiving NGHAM packets

Chapter 5

Results and discussion

In this chapter the results of the testing in Chapter 4 will be discussed. Unless otherwise mentioned, all plots are from the built in GNU Radio blocks.

The first test was performed to verify the capabilities of the GMSK modulator and demodulator blocks (Section 4.1). The plot in Figure 5.1 shows the constellation of the received signal before and after synchronization. In the left plot the points are distributed on a circle, because the sampling timing does not match the actual symbol times. In the right plot the samples form a quadrature of discrete points, indicating that the sampling timing is synchronized to the signal.

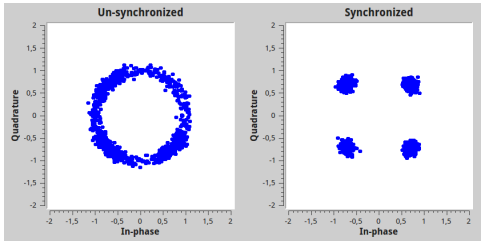

Figure 5.1: Constellation plots of the incoming signal before and after synchronization

In Figure 5.2 the original signal is shown with the demodulated signal. They both show the same bit pattern of `[1,1,1,0,1,0,0,0]`, indicating correct modulation and demodulation.

The next test was to see if the encoder and decoder was compatible with each other (Section 4.2). With both having the "Print packets" option enabled the following were printed by both of them. In Figure 5.3 the output from the encoder is shown, it includes both the hexadecimal and ASCII representation of the encoded bytes.

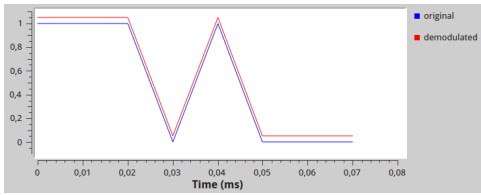

Figure 5.2: Plot of demodulated data over the original

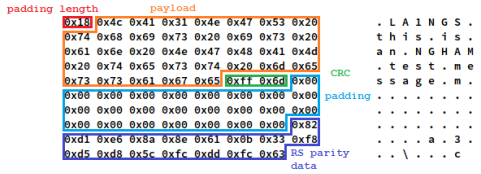

Figure 5.3: Output from the encoder with thevarious parts of the packet highlighted

The last test was to verify end-to-end communication between the USRP and OWL (Section 4.3). First the OWL was set to send 1000 packets with the USRP receiving, and then the other way around. In both cases only a small number of packets were lost ($< 1\%$). This proves that the USRP and GNU Radio combination is capable of end-to-end communication.

Figure 5.4 shows a waterfall plot of the transmission, where separate packets can be seen. It cas captured using the spectrum analyzer tool Baudline[28]

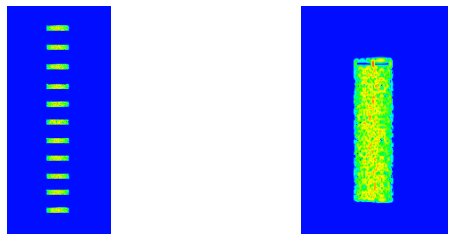

Figure 5.4: Waterfall plot showing a stream of NGHAM packets and a zoomed version

Chapter 6

Conclusion

The work described in this report has shown a way of implementing the data link layer protocol NGHAM for a ground station based on USRP and GNU Radio. It has been proven to be able to correctly handle GMSK modulation and demodulation. The encoder and decoder implemented are correctly translating the transmitted and received data.

In the end, end-to-end communication has been demonstrated, with satisfactory results.

The next chapter will discuss some unsolved problems which require more work in the future.

Chapter 7

Future Work

With the link layer protocol implemented, a part of the ground station is complete. Although its ability to communicate with the OWL VHF module is verified, it requires some more testing. Its ability to handle low signal-to-noise ratio and how this affects loss of data should be measured.

After it is thoroughly tested there remains to implement a the higher level protocols. The network layer CubeSat Space Protocol (CSP), developed at Aalborg University[29], will be used internally on the satellite and may also be used on the ground station.

To do this, current plan is to use the GNU Radio UDP source and sink blocks to communicate with another process. This will hopefully allow for an easily implemented application to handle incoming data and sending packets. The other blocks built in to GNU Radio.

Another unsolved problem is to integrate doppler prediction for correctly setting the carrier frequency of the ground station. There is already some functionality within GNU Radio, but this will have to be explored further.

Appendix A

GNU Radio types

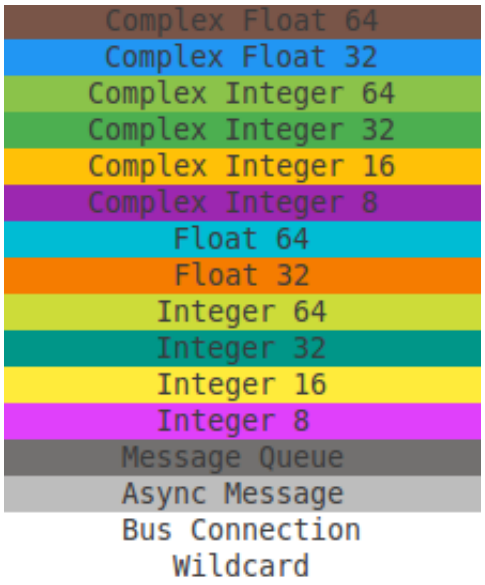

Figure A.1: Color codes for available types in GNU Radio

Appendix B

NGHAM Sync Word

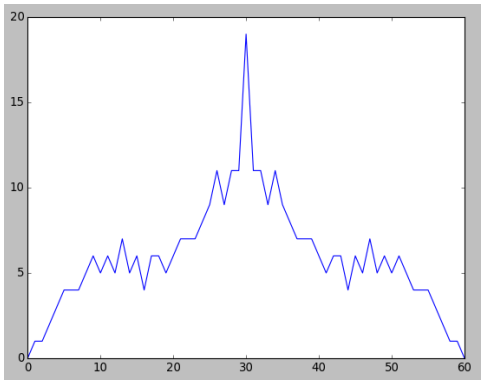

Figure B.1: NGHAM sync word auto correlation, plotted using Python

List of Figures

Bibliography

[1] André Løfaldli. GitHub: gr-nuts repository. https://github.com/lofaldli/gr-nuts. (accessed 15.12.2015).

[2] Nuts homepage. http://nuts.cubesat.no/om. (accessed 15.12.2015).

[3] Roger Birkeland. NUTS wiki, OWL VHF. https://www.ntnu.no/wiki/display/nuts/Owl+VHF. (accessed 15.12.2015).

[4] Mathias Tømmer. NUTS wiki, UHF Radio. https://www.ntnu.no/wiki/display/nuts/UHF+Radio. (accessed 15.12.2015).

[5] Roger Birkeland. NUTS wiki, Communication System. https://www.ntnu.no/wiki/display/nuts/Communication+System. (accessed 15.12.2015).

[6] ICOM homepage. http://www.icomamerica.com/en/products/. (accessed 15.12.2015).

[7] Ettus Research online store. http://www.ettus.com/product. (accessed 15.12.2015).

[8] Wikipedia page on USRP. https://en.wikipedia.org/wiki/Universal_Software_Radio_Peripheral. (accessed 15.12.2015).

[9] GNU Radio homepage. http://gnuradio.org/redmine/projects/gnuradio/wiki. (accessed 15.12.2015).

[10] Ettus Research online store. http://www.ettus.com/product/details/WBX. (accessed 15.12.2015).

[11] Timo A. Stein. NUTS wiki, Ground Station. https://www.ntnu.no/wiki/display/nuts/Ground+station. (accessed 15.12.2015).

[12] Beathe Hagen Stenhaug. Antenna system for a ground station communicating with th NTNU Test Satellite (NUTS). Master's thesis, Norwegian University of Science and Technology, July 2011. (accessed 15.12.2015).

[13] Jon Petter Skagmo LA3JPA. GitHub: NGHAM repository. https://github.com/skagmo/ngham. (accessed 15.12.2015).

[14] Wikipedia page on Cyclic redundancy check (CRC). https://en.wikipedia.org/wiki/Cyclic_redundancy_check. (accessed 17.12.2015).

[15] Wikipedia page on Reed Solomon. https://en.wikipedia.org/wiki/Reed-Solomon_error_correction. (accessed 17.12.2015).

[16] Wikipedia page on scrambling. https://en.wikipedia.org/wiki/Scrambler. (accessed 17.12.2015).

[17] Wikipedia page on Minimum Shift Keying. https://en.wikipedia.org/wiki/Minimum-shift_keying#Gaussian_minimum-shift_keying. (accessed 15.12.2015).

[18] Simon Haykin. *Communication Systems*. Wiley, 4 edition, 2000.

[19] Wikipedia page on GNU Radio. https://en.wikipedia.org/wiki/GNU_Radio. (accessed 16.12.2015).

[20] GNU Radio Manual and C++ reference: Handling flow graphs. http://gnuradio.org/doc/doxygen/page_operating_fg.html. (accessed 15.12.2015).

[21] GNU Radio Manual and C++ reference: Polymorphic Types. http://gnuradio.org/doc/doxygen/page_pmt.html#pmt_datatype. (accessed 15.12.2015).

[22] GNU Radio Manual and C++ reference: Python Blocks. http://gnuradio.org/doc/doxygen/page_python_blocks.html. (accessed 15.12.2015).

[23] Karl David Vea. NUTS: Ground station with GNU Radio and USRP. Master's thesis, Norwegian University of Science and Technology, June 2015. (accessed 15.12.2015).

[24] Document from the University of Hull (from Wikipedia page on MSK). http://www.emc.york.ac.uk/reports/linkpcp/appD.pdf. (accessed 15.12.2015).

[25] GNU Radio Manual and C++ Reference: Clock Recovery MM. http://gnuradio.org/doc/doxygen/classgr_1_1digital_1_1clock_recovery__mm__cc.html. (accessed 17.12.2015).

[26] Phil Karn KA9Q. Forward error correction library. http://www.ka9q.net/code/fec/. (accessed 16.12.2015).

[27] GitHub: Vector Optimized Library of Kernels (VOLK). https://github.com/gnuradio/volk. (accessed 20.12.2015).

[28] Baudline project page. http://www.baudline.com/. (accessed 20.12.2015).

[29] GitHub: CubeSat Space Protocol (CSP). https://github.com/GomSpace/libcsp. (accessed 18.12.2015).