FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Short Wave Transmitter for STRAPLEX

Nuno Moreira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Sérgio Reis Cunha (PhD)

July 25, 2018

Resumo

STRAPLEX é uma plataforma educacional que permite aos estudantes levarem as suas experiências até à estratosfera. Para o efeito, as experiências são colocadas dentro de cápsulas que são elevadas por balões de hélio. Quando atingida uma altitude entre os 35 e os 40km o balão rebenta e é iniciada a descida da cápsula, com o auxílio de um paraquedas. Uma vez que não existe nenhum mecanismo de controlo na fase descendente, capaz de definir o local exato de aterragem, é necessário obter informação periódica sobre a sua localização.

O objetivo desta dissertação é desenvolver um emissor de HF que será colocado a bordo de uma cápsula STRAPLEX. Este emissor irá receber informação sobre a localização da cápsula, através de um recetor GPS, codificar esta informação num protocolo robusto, MFSK16, e transmitir estes dados periodicamente. Para o efeito será utilizado um recetor GPS comercial, um microcontrolador que será responsável pelo controlo de todo o sistema, uma FPGA de forma a garantir rigor temporal do protocolo na codificação dos dados e gerar o sinal de saída resultante da codificação da informação e um amplificador de potência, que será responsável por amplificar este sinal. A FPGA irá gerar uma onda de frequência próxima de 10.07MHz e irá modular a informação usando diferentes frequência espaçadas 15.625Hz entre elas.

Foi obtido um sistema capaz de transmitir informação codificada com baixa potência. Testes mostraram que é necessário um nível de sinal significativamente baixo relativamente ao nível de ruído para total descodificação desta informação. Tendo em conta as propriedades das ondas HF, como resultado é também esperado conseguir-se comunicar a longas distâncias, fora da linha de vista. Considerando o protocolo utilizado é também esperado obter-se um sistema robusto a fading.

Abstract

STRAPLEX is an educational platform that allows students to fly their experiences into the stratosphere. For this purpose, the experiments are placed into capsules which are raised by helium balloons. When it reaches an altitude between 35 and 40km the balloon bursts and the descent phase of the capsule is initiated, through the aid of a parachute. Since there is no control mechanism capable of defining the exact landing location, it is necessary to obtain periodic information about its location.

The objective of this dissertation is to develop an HF transmitter that will be placed on-board of a STRAPLEX capsule. This transmitter will receive information about the location of the capsule through a GPS receiver, encode this information in a robust protocol, MFSK16, and transmit this data periodically. For this purpose, will be used a commercial GPS receiver, a microcontroller, that will be responsible for the control of the whole system, an FPGA that will guarantee the temporal accuracy in the coding of data and generate the output signal that encodes the information to be transmitted and a power amplifier, which will be responsible for amplifying this signal. The FPGA will generate a wave around 10.07MHz and will modulate the information using different frequencies spaced 15.625Hz between them.

A system capable of transmitting coded information using low power was obtained. Tests have shown that for total decoding of this information the required signal level in relation to the noise level is significantly low. Taking the properties of the HF waves into account, it is also expected to be able to communicate over long distances, beyond line of sight. Considering the protocol used it is also expected to obtain a system that is robust to fading.

Agradecimentos

Ao meu orientador, Professor Sérgio Reis Cunha, por toda a dedicação, disponibilidade e pelo saber que me transmitiu ao longo de todo o trabalho realizado.

Aos meus colegas e amigos Américo Duarte, Bruno Correia e David Leite por todo o apoio, momentos e aprendizagens.

À minha família, especialmente aos meus pais, por terem sido o meu suporte e por me terem dado a oportunidade de chegar até aqui.

À minha namorada Catarina, por toda a compreensão, paciência e amor demonstrado.

Nuno Moreira

"A person who never made a mistake never tried anything new."

Albert Einstein

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

APRS	Automatic Packet Reporting System
ARRL	American Radio Relay League
BPSK	Binary Phase Shift Keying
DAC	Digital to Analog Converter
DRC	Design Rule Check
FEC	Forward Error Correction
FF	Flip Flop
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
HF	High Frequency
HW	Hardware
MFSK	Multi Frequency Shift Keying
MSB	Most Significant Bit
PCB	Print Circuit Board
PSK	Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
Rx	Receive
STRAPLEX	STRAtospheric PLatform EXperiment
SNR	Signal to Noise Ration
SW	Software
TCxO	Temperature Compensated Crystal Oscillator
Tx	Transmit
UTC	Universal Time Coordinated
VFO	Variable-Frequency Oscillator
VHF	Very High Frequency
WPM	Word Per Minute

Chapter 1

Introduction

The purpose of this chapter is to give an introduction to the work described in the following sections. It indicates the motivation that gave rise to the work as well as the main objective in the accomplishment of the same. The subchapter 1.3 also contains the structure of the document.

1.1 Motivation

STRAPLEX is an educational platform that allows students to fly their own experiments in a capsule transported by a balloon filled with helium into the stratosphere. Experiments usually contain sensors and collect data that is locally stored. In order to have access to that data, it is necessary to recover the capsule. For this to happen it is necessary to know the landing site. Currently, this location, given by a GPS receiver is coded using Automatic Paclet Reportign System (APRS) protocol and transmitted through a VHF channel. Although this protocol is widely used by radio amateurs, this implementation is only reliable for line of sight transmissions. Using a HF transmitter and a robust communication protocol it is possible to reach longer distances, beyond the line of sight, taking advantage of the ionospheric reflection characteristics of these short waves.

1.2 Objectives

The main objective of this work is to develop a HF transmitter that will be on-board of a STRAPLEX capsule. This transmitter will code location data with a reliable and robust protocol, MFSK 16, and will transmit periodically this information. An unidirectional communication link will be established from capsule to the ground station, allowing to obtain coordinates of the landing site of the balloon. The coded data will contain information about position and will be sent each two minutes, which will be enough to find the balloon location. The protocol that will be used is reliable and widely used by amateur radios, it uses forward error correction, presents a high robustness to non-ideal effects as fading and multipath and performs well at low signal strengths, once the SNR necessary to decode the information is very low.

1.3 Thesis structure

This thesis is organized in six chapters. The first one has an introductory character addressing the objectives and the motivation for the accomplishment of this dissertation. In the second chapter there will be a description of the problem as well as the approach used to solve it. It also includes the state of the art. The third chapter contains a theoretical component about HF communications and the detailed description of the used protocol. The implementation of the system is described in the fourth chapter and the tests and results are presented in the fifth chapter. The sixth chapter contains the conclusion and the future work.

Chapter 2

Overview

This chapter is divided into four subchapters. The first one aims to give an overview about the STRAPLEX project, the second one describes the problem and third one the proposed approach to solve the problem. The last subchapter aims to give an insight into the work done by some authors so far. Although the scientific work is not very extensive in amateur radio, it is possible to find many information and works published by various authors in their personal websites, as well as official websites of various organizations and associations.

2.1 STRAPLEX

STRAPLEX stands for STRAtospheric PLatform EXperiment and is a program of the Faculty of Engineering of the University of Porto, created in 2005. This program offers students the possibility to fly their own experiments into the stratosphere in order to be tested in a very close space environment. The experiments are placed in a styrofoam capsule which is raised through a helium balloon. Depending on the mass of the payload, this platform allows the experiences to reach 30km of altitude. On board the styrofoam capsule there is also a navigation system consisting of a radio and an APRS modulator, as well as the transmission of video, or analog video or more recently digital video. The aim of this system is to locate the capsule in such a way that it is possible to retrieve it as well as collect the obtained data, often stored only locally. A launch of a STRAPLEX balloon can be divided into five distinct phases: the launch, where the latex balloon is filled with helium, the box is turned on and is verified if the coordinates are sent by the navigation system to the ground station; the ascending phase, which is usually the most time consuming and where most of the data is collected; the descending phase, controlled only by a parachute, the landing and consequently the retrievement of the capsule. In the ascending phase the balloon rises up to the stratosphere, as it rises, the pressure decreases, leading to the expansion of the helium and consequently the expansion of the balloon, causing its bursting around the 30km of altitude. It is also possible to send a cut-off signal allowing the capsule and the parachute to separate from the balloon, thereby interrupting the ascending phase and commencing the descending phase. After the balloon burst, due to the non-existence of air resistance, the beginning of the descent

is somewhat uncontrolled, however the lower the altitude, the higher the friction is caused and the parachute begins to slow down the capsule. The landing site is then defined only by wind. During the ascending and descending phase, the balloon is tracked, based on the coordinates received. This is done both in faculty and in a car equipped with an antenna and a radio that follows the capsule to the place of landing. Currently, this platform also supports another project, BEE (Balões Estratosféricos nas Escolas), which means Stratospheric Balloons in Schools, which allows secondary school students to build their own capsules and experiments and thus arouse interest for engineering. The balloons are launched in the schools that manufacture the capsules, which are located mostly in the Porto district.

2.2 The Problem

The STRAPLEX capsules are manufactured in such a way that it is possible to carry an experiment into the stratosphere. This experience varies according to the purpose of the test. They are often constituted by sensors that generate data that is stored locally on an on-board computer or by biology experiments. In any case, it is necessary to recover the capsule in order to collect the data obtained during the flight. For this purpose a VHF channel communication system is used using an APRS protocol that encodes and sends coordinates received by a GPS. Although this system is reliable, it is not very compact and consumes considerable power. Since it is intended to fly to the stratosphere it is necessary to reduce the mass, so the amount of helium required is also smaller. This will make possible to reach higher altitudes. Experience is also supplied by batteries, which have a limited life-time. Reducing the consumed power it is possible to perform larger flights and take more time during the capture of the capsule, since capsule recovery is not always easy. VHF channels only allow communication in the line of sight, so it is not only necessary to follow the capsule during the flight in order to collect coordinates constantly as after landing the capsule it is difficult to obtain data from this type of communication, since often lands in areas with a lot of vegetation, there are too much obstacles that prevent the correct reception of the signals.

2.3 The Proposed Solution

Using a dedicated modulator and a robust protocol on an HF channel it is possible to establish communication over long distances as well as reduce mass and energy consumption. The chose protocol, Multi Frequency Shif Keying 16 (MFSK 16), is robust to external interference, multipath and fading. Tests have shown that it is possible to decode information even when the noise is 21dB higher than the signal level. Since the speed of the capsule is relatively slow, and after landing its position is not changed, it is sufficient to send information every two minutes. Correspondingly, a the low data rate is sufficient to meet the requirements. The block diagram of the transmitter can be seen in the figure 2.1.
All components will have a wake up/sleep system in order to reduce energy consumption. The entire system will be controlled by the microcontroller that will receive data from the GPS receiver

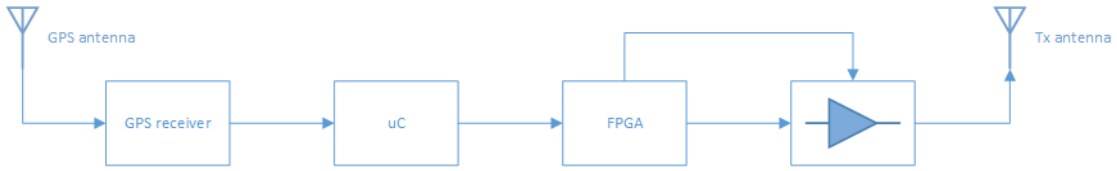

Figure 2.1: Proposed Solution Block Diagram

each two minutes, wake up all system components and perform processing along with the FPGA to modulate the signal. The output FPGA signal will be then amplified by a power amplifier and transmitted.

The main objective of this solution is to take advantage of ionospheric reflection. During the flight the information will be transmitted via direct wave; however after landing it is intended that the emitted wave be reflected in the ionospere and be correctly received in the ground station. Landing sites are often dense in vegetation, making it difficult to receive VHF signals. Using HF communications is intended to communicate over large distances using low power. The two types of possible received waves are shown in the figures 2.2 and 2.3.

2.4 State Of The Art

Radio amateur is a scientific-technological hobby which consists of the use of a radio station to communicate, this activity is regulated by issuing certificates of authorization by competent entities. There are several types of communication and can be used for various purposes. It is possible to communicate within a locality or between different continents without using internet or telephone service. There are different modes, some of them specified in [1]. Among these modes is the Morse code that although it is one of the oldest is also one of the most popular. Data representation can be done in text, image, audio or video, and there are appropriate modes depending on the type of data to be transmitted. Among the various applications of radio amateur one that stands out is the use of amateur signals in situation of natural catastrophe. Steve Richards [2] explores this concept and tests several ways that can be used through experimental procedures. Among these modes are PSK31, MFSK16 and MT63, suitable for sending data in text format over HF channels.

PSK 31 is a phase-shift keying mode, developed by Pawel Jalocha (SP9VRC) and Peter Martínez (G3PLX). This mode is explored in articles [3], [4] and [5] and defined in the ARRL website [6]. It is subdivided into two modes BPSK 31 (binary phase-shift keying) characterized by the use of two phases and QPSK 31 (quadrature phase-shift keying) in which four phases are used. Both have a baudrate of 31.25baud, each symbol is composed by a bit, which alternates the phase, in BPSK 31, or by a dibit, which defines the phase of 0^o, 90^o, 180^o or 270^o, in QPSK 31. Each character is encoded in a binary pattern of variable size, varicode, with the most frequent symbols being smaller in size. The varicode table corresponding to PSK31 mode can be seen in [4]. Due to this feature it is possible to achieve an average typing speed of 50WPM. They have a

narrow band, 31Hz, and it is necessary stable VFOs to decode their contents. The binary version has no error correction code; however the QPSK 31 version corrects errors through a Viterbi decoder that based on the previous 20 symbols produces estimations of the received symbol phase. Despite correcting errors, this decoder introduces a delay of 640ms.

MFSK 16 is a frequency-shift keying mode, characterized by its good behavior when subjected to external interference and noise. The characteristics of the MFSK 16 mode are presented in [1], [7], [8] and [9]. This mode has a baudrate of 15.625baud and 16 different tones. One tone is sent at a time, in the temporal domain, and are spaced 15.625Hz among each other in the frequency domain. Each tone encodes four bits. The total bandwidth of the signal is 316Hz. This mode uses binary varicode [10], however the table used is slightly different from that used in PSK 31, allowing a data rate of approximately 30WPM. In order to correct errors, convolutional coding algorithms are used and the output is mapped via gray coder. Since this mode is very sensitive to frequency variations, at the beginning of the transmission there is a preamble in order to synchronize the transmitter and the receiver clocks; the transmission starts with a small burst at the lowest frequency, which works as a tuner. Since the errors in fading HF channels usually occurs in bursts, a temporal interleaver is used in order to exchange the errors temporally.

MT 63, developed by Pawel Jalocha (SP9VRC), is a OFDM mode, has three different versions: MT63-500, 500Hz bandwidth, MT63-1k, 1kHz bandwidth, and MT63-2k, 2kHz bandwidth. It is characterized by 64 2-PSK modulated tones that are transmitted simultaneously in time. The symbol rates are 5baud, 10baud and 20baud, respectively, corresponding to typing rates of 50WPM, 100WPM and 200WPM. The modes have high level of error correction: use Walsh functions that spread 7-bit ASCII characters in 64 tones; the interleaver in the temporal and the frequency domain results in a rejection of the impulsive noise. Given the high error rate, the tunning is not critical: the MT63-1k receiver can be tuned up to 100Hz on the side and MT63-2k tuned up to 250Hz. This mode despite having high error correction is quite sensitive to steady-state noise. MT 63 is the basis of the work presented in [11], [12] and [13] and also presented in [1].

The objective of the experiments carried out by Steve Richards [2] was to characterize these three modes, as well as their viability for communicate during emergency situations. To that end, it was defined the test criteria: accuracy (error correction), data rate, weak signal performance, tolerance to interference, efficiency, ease of operation, compatibility and adaptability, reliability and availability. The experience consisted of using the sound cards from two different computers in order to transmit information between them, the CoolEdit program was used to change parameters such as noise and introduce delays. Concerning performance under noise, a steady-state noise test was preformed, which consisted of the transmission of a text excerpt, modulated in each of the modes with white Gaussian noise added. It was found that the signal was fully decoded when the noise signal ratios were higher than those shown in the table 2.1.

It is possible to verify that under these circumstances MFSK16 performed better than the other modes. The behavior of these modes was also tested when transmitting the original signal along with a delayed copy in order to emulate multipath interference. The mode with the best behavior was found to be MT63-1k, where no interference was noted, and the worst performing mode was

Table 2.1: Performance Under Steady-State Noise

	MT63-2k	**MT63-1k**	**BPSK31**	**QPSK31**	**MFSK16**
Audio S/N ratio	-8dB	-10dB	-11dB	-14dB	-18dB

BPSK 31. Regarding ease of operation, it was concluded that the easiest way to decode would be PSK 31, the most difficult being MFSK 16 After performing several tests the author summarizes the results and concludes which modes should be used and under what conditions. Steve Richards says that MT63 should be used when conditions are good and bandwidth is not a problem, MT63-2k being the preferred one since it has a higher data rate and is more robust to errors. When the signals are weak or unstable, MFSK 16 mode is preferable as it performs well on channels with a lot of noise. However, if bandwidth is a problem, we want to communicate on a congested channel, then PSK31 should be used, and QPSK 31 mode is appropriate if we want an error correction.

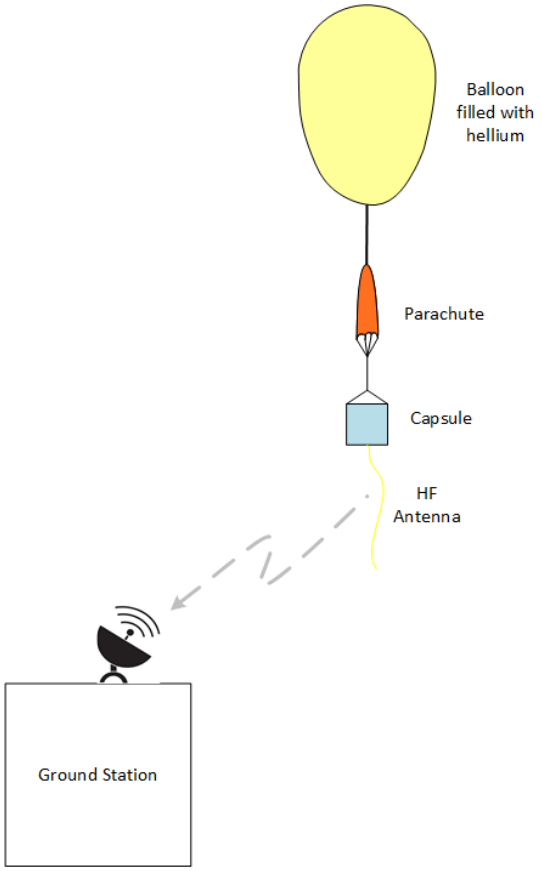

Figure 2.2: Direct Wave

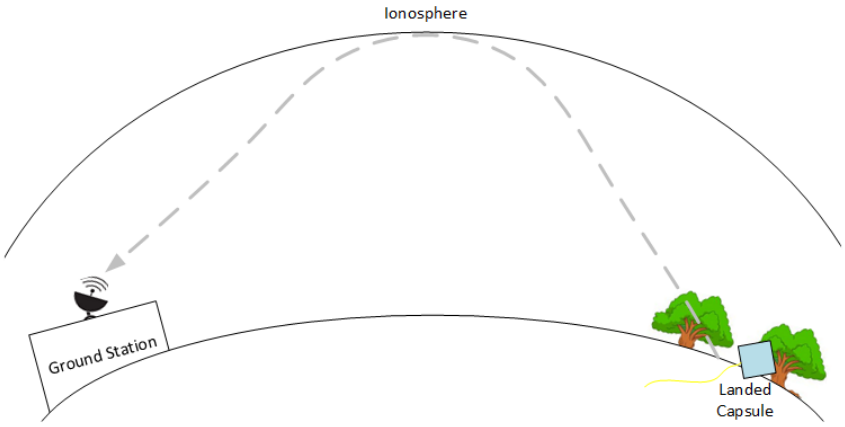

Figure 2.3: Reflected Wave

Chapter 3

HF Communications

This chapter describes the protocol used to implement the proposed solution. It is intended to give a deeper insight into how messages are encoded. It also addresses the purpose of HF communications as well as their properties. This chapter is an introductory chapter to the description of the implementation in Chapter 4.

3.1 Short Wave Characteristics

This subject is described in the article [14]. However, given their relevance to the operation of the system, some features of HF communications as well as important features of the Ionosphere are discussed in this subchapter. The HF band comprises electromagnetic waves with frequencies above 3 MHz and below 30 MHz. These bands are of particular interest because of a phenomenon that occurs in the ionosphere known as ionospheric reflection. This phenomenon allows the propagation of the waves to greater distances, which is not the case with VHF waves or upper bands. This phenomenon is characterized by the reflection of radio waves at frequencies below 40 MHz.

The ionosphere is a layer of the atmosphere that is located between 60 km and 1000 km altitude. In the ionosphere the pressure is low enough for the ions to travel freely. Ionization is essentially caused by ultraviolet rays. Due to this fact, the characteristics of this layer of the atmosphere vary with the rotation of the earth.

A wave passing through the ionosphere is successively refracted and, under certain conditions, can be returned to Earth similar to reflection. This event depends on the density of ionization of the ionosphere and the frequency of the wave. The higher the ionization density, the greater the bending angle, however, the bending angle decreases with frequency. The ionosphere is composed of three layers, layer D (50-90 km), layer E (100-115km), layer F (160-420 km). The different layers are illustrated in the figure 3.1.

The D layer only exists during daylight, and usually disappears 30 minutes after sunset. It is particularly dense. Given its density there is a large amount of particle collisions leading to the recombination of the ions in neutral atoms. This layer disappears at night. This layer instead of refracting and propagating signals, absorbs large amounts of energy. The energy of the wave

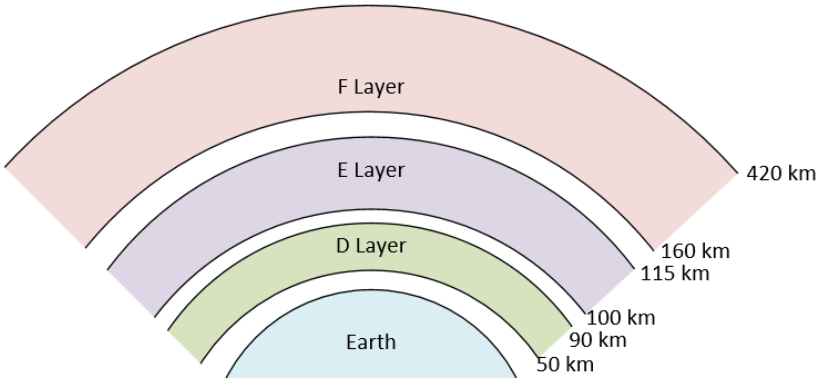

Figure 3.1: Ionospheric Layers

passing through this layer is transformed into heat or motion energy due to collisions between the wave and the particles. The lower the frequency, the greater energy is absorbed. This makes it impossible to perform long distance communications in the 160, 80 and 40 meter bands during daylight hours. The bands 20 meters and above are not significantly affected by this absorption effect.

Layer E supports propagation of radio waves and have characteristics similar to layer D. The maximum ionization E occurs in the noon and decreases rapidly after sunset. The E layer absorbs energy in amateur bands of lower frequency, but not as much as in layer D.

The F layer is the layer that allows long distance HF communications. As the F layer is at a higher altitude, the rarefied is also larger, causing fewer collisions of particles. This causes the F layers to remain ionized at night. Maximum ionization occurs at noon and the minimum occurs one hour before sunrise.

Another important aspect has to do with the transmitted frequency and the critical angle. The critical angle is the minimum angle of the transmitted wave that allows its reflection in the ionosphere. Associated with this concept is the skip distance, which is an area where it is not possible to propagate the reflected wave by being very close to the emitter. The frequency also varies depending on the distance that it is desirable to reach. The critical angle and the skip distance concepts are illustrated in the figure 3.2

3.2 Protocol Description

The communication protocol used to establish communication between the capsule and the ground station is the MFSK16. It is an Amateur Digital Mode used for text communications. This mode uses a FEC and an interleaver code to be robust enough for long-path communications in HF bands. This mode is described in [1], [7], [8] and [9].

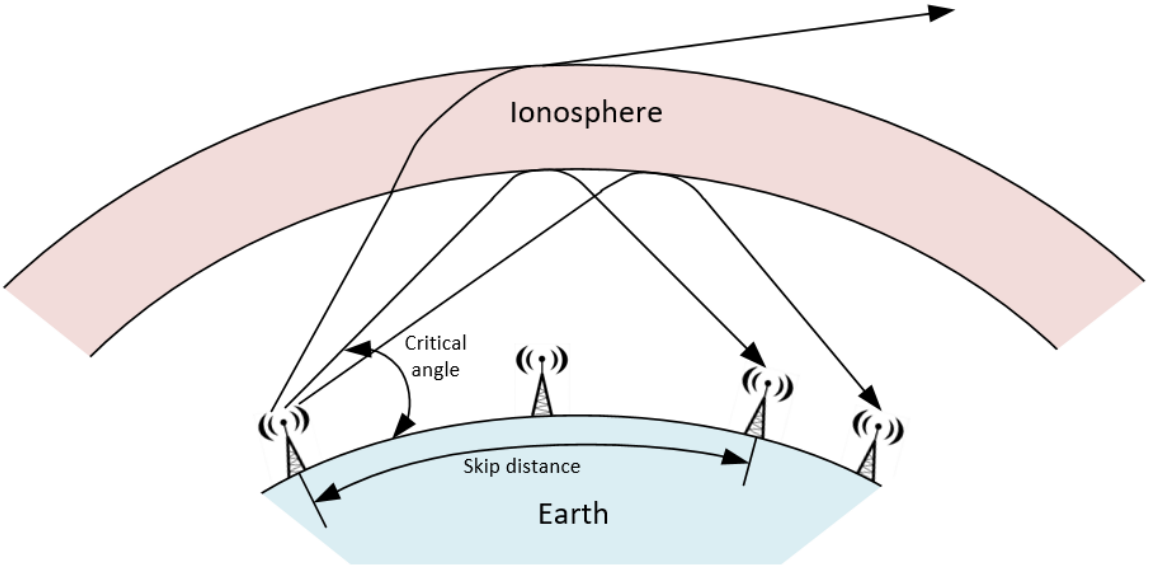

Figure 3.2: Ionospheric Reflection

3.2.1 MFSK Modulation

The MFSK modulation characterizes by the codification of M bits into $\log_2(M)$ different frequencies, or tones. Each symbol has the same duration, T and the same amount of energy, E. The space between two frequencies is $\frac{1}{T}$. f_0 is the lowest tone. The transmitted signal is defined by the following equation:

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos\left[2\pi t(f_0 + \frac{n}{T})\right], \quad 0 \leq t \leq T, \quad 0 \leq n \leq (M-1) \quad (1)$$

In this particular mode, MFSK16, each symbol encodes four bits and is transmitted for 64 ms. The symbols are transmitted continuously, with no interruption in transmission between different symbols. The baudrate of this mode is 15.625 baud and the spacing between frequencies is 15.625 Hz. This results in a theoretical bandwidth less than 500Hz. However, using real signals it is found that most of the transmitted power is in a bandwidth of approximately 300Hz. The figure 3.3 illustrates the power spectral density of a signal encoded in this mode. The signal was recorded in audio, the sampling frequency is 48 kHz. We can verify that much of the transmitted power is at approximately 200 Hz. In the implementation described in this document, f_0 will be 10.070 MHz, with the upper tones being in higher frequencies, multiple of the frequency range, the highest frequency being 10.070234375 MHz. In order to better understand the evolution of signal in time and frequency it is possible to see in the figure 3.4 a spectrogram of an example signal. The frequencies of the zero and fifteen tones are also indicated, in order to have a perception of the useful bandwidth used.

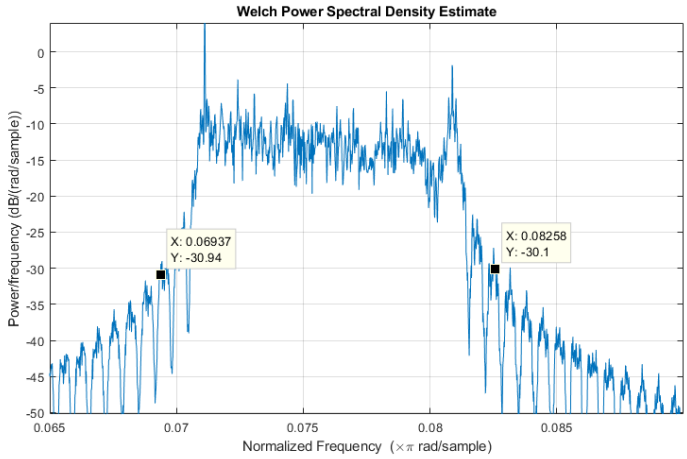

Figure 3.3: MFSK16 Signal Spectrum

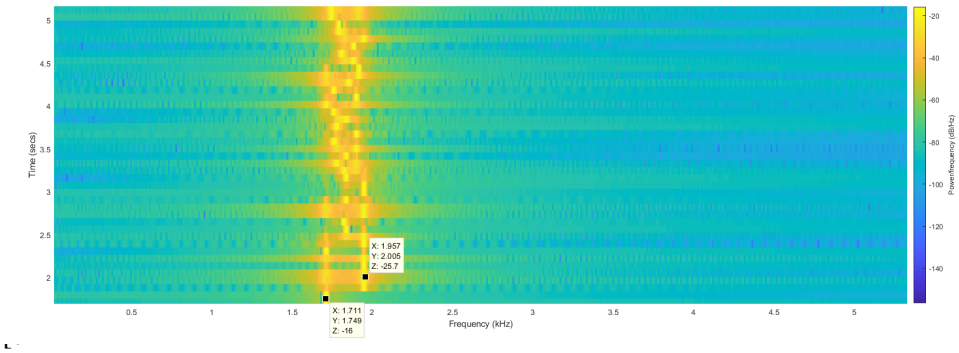

Figure 3.4: MFSK16 Spectrogram

3.2.2 Varicode

Another feature of this mode is the use of a varicode alphabet, developed by Nino Porcino and Murray Greenman, and described in [10]. This alphabet is characterized by the encoding of ASCII characters in a variable number of bits, with the most frequent letters having fewer bits than the less frequent ones. For example, the character "e" is encoded in 4 bits (1000) and the special character "#" is encoded in 10 bits (1011011000). This alphabet is case-sensitive. Since lowercase characters are more frequent than uppercase or numbers are also encoded in fewer bits. This alphabet is optimized for English language, and it is possible to transmit 43WPM using this mode with this alphabet. Another feature of this alphabet is the non-existence of the sequence 001. Whenever this sequence is verified it indicates the change of character.

3.2.3 FEC Coder

This mode also uses a standard convolutional code of rate $r = \frac{1}{2}$ and maximum size $k = 7$, for correction of random errors. This means that for each bit in the input, a dibit will be produced at the output, taking into account the current bit and the six previous bits. Shift-registers are usually used in order to implement the these convolutional codes. The convolutional codes are

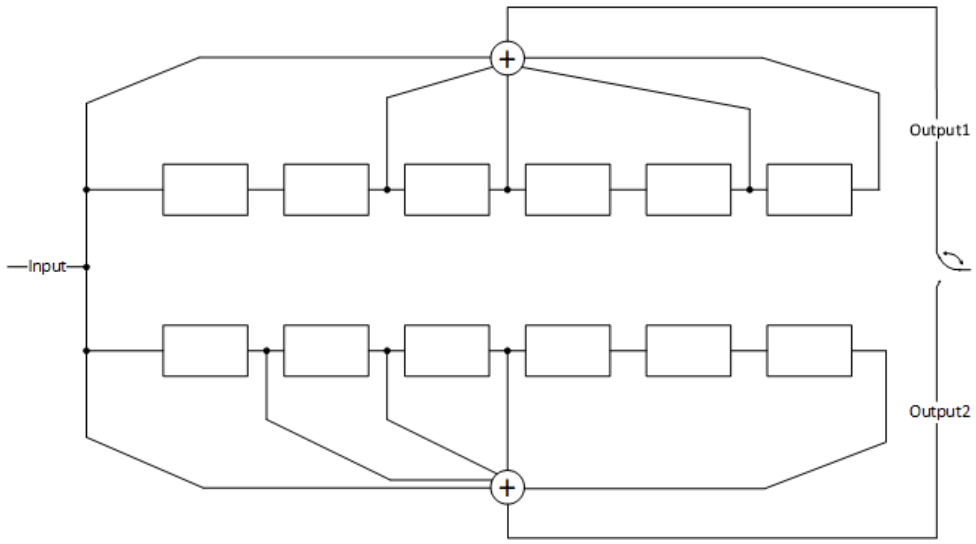

Figure 3.5: FEC Coder R=1/2, K=7

also characterized by their generator polynomials. The polynomials that define this code are given by:

$$O_1 = I_0 + I_2 + I_3 + I_5 + I_6$$

$$O_2 = I_0 + I_1 + I_2 + I_3 + I_6$$

I_0 is the input at the current instant and I_6 the oldest input and O_1 and O_2 are the outputs. At the output of the FEC coder, the bits are demultiplexed into a single data stream and this is organized with $O_1 O_2$, MSB first. The sums used are of modulo 2. A schematic of this convolutional code can be visualized in the figure 3.5. This error correction code is described in [15].

3.2.4 Interleaver

Since noise in HF communications generally occurs in bursts so that no part of the sent message is lost an interleaver is used. The effect of an interleaver is to spread the bits of a message in time before transmitting them, shuffling them so that when receiving the message and reorganizing the bits the errors are treated as random, avoiding losing information of the transmitted message. When being treated as random errors it is possible to use conventional error correction algorithms. The interleaver in this way consists in choosing each four-bit bit until a ten-bit message is obtained:

Original data stream: $B_0\, B_1\, B_2\, B_3\, B_4\, B_5\, B_6\, B_7\, B_8 \ldots B_n$

Interleaved data stream (1): $B_0\, B_4\, B_8 \ldots$

Whenever a ten bit message is obtained, it starts grouping each two bits and place them in the data stream by the following order:

Table 3.1: Tone Encoding

Tone	Bits to encode	Tone	Bits to encode	Tone	Bits to encode	Tone	Bits to encode
0	0000	4	0110	8	1100	12	1010
1	0001	5	0111	9	1101	13	1011
2	0011	6	0101	10	1111	14	1001
3	0010	7	0100	11	1110	15	1000

Interleaved data stream (2): $B_0\,B_4\,B_8\,\ldots B_{36}\,B_{40}\,B_1\,B_{44}\,B_5\,\ldots$

When message get twenty bits, it starts grouping three to three and when it has thirty bits in the message it starts grouping four to four. Until the end of the message. If there are not enough bits to group the whole message, null bits are added in order to spread all the bits of the message in time.

Interleaved data stream (3): $B_0\,B_4\,B_8\,\ldots B_{36}\,B_{40}\,B_1\,B_{44}\,B_5\,\ldots B_{120}\,B_{81}\,B_{42}\,B_3\,\ldots$

The interleaver is described in [16].

3.2.5 Gray Code

Another characteristic of this mode is that each tone codes four bits using gray-code. This code is characterized by the difference of only one bit between two adjacent tones. This is, gray-code ensures that the hamming distance between two adjacent tones is always one. The table 3.1 summarize the bits that codes each tone. This table can also be found in [8].

3.2.6 Other Characteristics

Other features that are important to refer to is the beginning of the transmission, in which it is necessary to send zero tone to eight symbol periods. However, at the end of the transmission, the transmission must end after the last symbol is transmitted, after which the transmission buffer is cleared with zero-value bits. When there is a pause in the transmission for a time greater than 20 ms, the NULL characters must be initially sent. This avoids long periods in which the same tone is transmitted, usually the zero tone.

3.3 Ground Station

The ground station used for this purpose will consist of an HF antenna, a radio and a computer. The radio will receive the signal through the antenna that sends the resulting sound into the line-in input of the computer running a protocol decoding software. The software used is a free license whose name is MixW 3.2. This software is available for download at [17]. The software also offers the possibility to decode and encode different amateur radio modes, such as BPSK31 and MT63. The sent string that will be decoded will contain coordinates that will be placed on a map in order to identify the best course for recovery of the capsule after landing.

Chapter 4

MFSK16 Transmitter

4.1 Implementation

The purpose of this subchapter is to give an overview of the implementation of the HF transmitter. It is divided into three sections. In the first section it is possible to see what information is sent from the STRAPLEX capsule to the ground station and how it is encoded. In the second section it is given a brief description of the main blocks of HW responsible for the codification and transmission of the information. The last section gives a brief explanation of how these blocks communicate with each other, as well as the protocols used.

4.1.1 Communication Protocol

In chapter 3.2 the characteristics of the protocol used are described. In order to encode the data, these characteristics must be taken into account. The aim of this thesis is to obtain the location of the STRAPLEX capsule throughout the flight and after landing. For this purpose, it is necessary not only to use a GPS receiver, in order to obtain coordinates, but also to select a message that contains all the desired information to be transmitted. Since this protocol is text-oriented this message may simply be a string, with different fields that will be populated with information received by the GPS receiver. The relevant information is: latitude and longitude, so as to know the exact location of the capsule, and altitude, to know in which phase of the flight is the capsule (ascent, descent or post-landing). To know that we are in the presence of a valid string, a special character, "#", is placed at the beginning of this string. The sent string is schematized in the figure 4.1. The figure 4.2 gives an overview of the processing performed. The characteristics of each block can be read in 3.2 and its implementation will be described in the following subchapters.
Briefly, a GPS receiver connected to an active antenna receives the GPS coordinates, the relevant information is placed in a string. Each character of the string is encoded using the varicode alphabet. The resulting data stream passes through a FEC coder and an interleaver. Each four-bit set is then encoded in a tone. The resulting signal is amplified and sent through an antenna connected to the output of the transmitter.

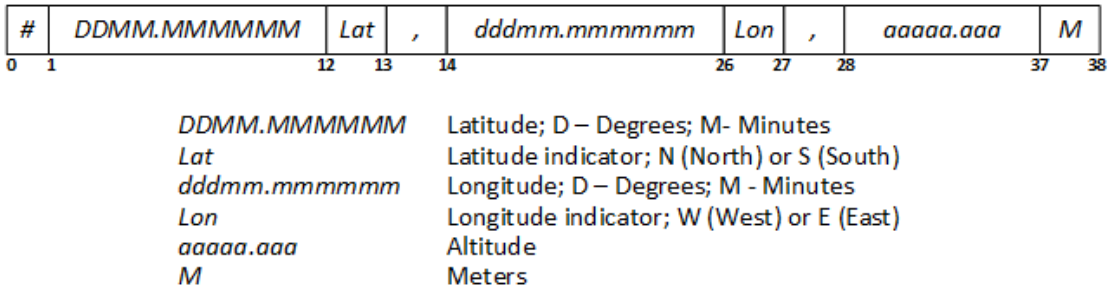

Figure 4.1: String To Transmit

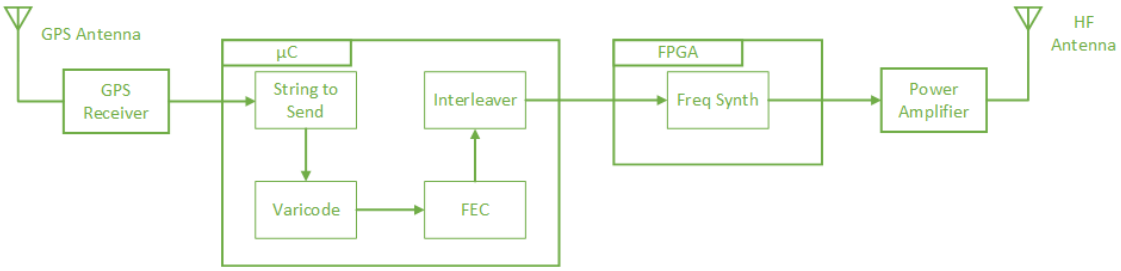

Figure 4.2: Transmitter Block Diagram

4.1.2 System Block Diagram

From a hardware point of view, a uBlox MAX-M8Q GPS receiver was used. This receiver was chosen because of the experience in using this model in other projects, it has already been tested in the stratosphere, and its low consumption; an Atmega328-P microprocessor, embedded in an arduino nano microcontroller, due to the need to reduce space, mass and power consumption; a FPGA Numato Mimas Development Board, which contains a Xilinx Spartan-6 XC6SLX9, this FPGA also features low power consumption and the dimensions are reduced. These two boards are used as daughter boards of a fabricated PCB that has only the necessary connections between them, as well as interfaces with antennas and the power amplifier. It is also used a TCxO that serves as the system reference oscillator. The use of a TCxO is required due to its high oscillation stability. This requirement is to ensure the proper functioning of the system. During the flight there are temperature changes, however it is necessary to ensure that the frequency sent does not change during the transmission. Compensating the temperature factor it is possible to guarantee the high stability of the crystal. The system is also composed by another PCB that contains the power amplifier. In the figure 4.2 it is possible to observe a block diagram of the system, as well as the processing that each subsystem performs. The microcontroller is in charge of processing without time constraints, such as varicode implementation, FEC and interleaver. The FPGA is then in charge of the frequency synthesizer, since the temporal modulation constraints are very strict. In the figure 4.3 it is possible to see the final result of the system. The system is assembled in stack and the connections between microcontroller, FPGA, TCxO and GPS receiver are made on the upper PCB and the lower PCB is the power amplifier.

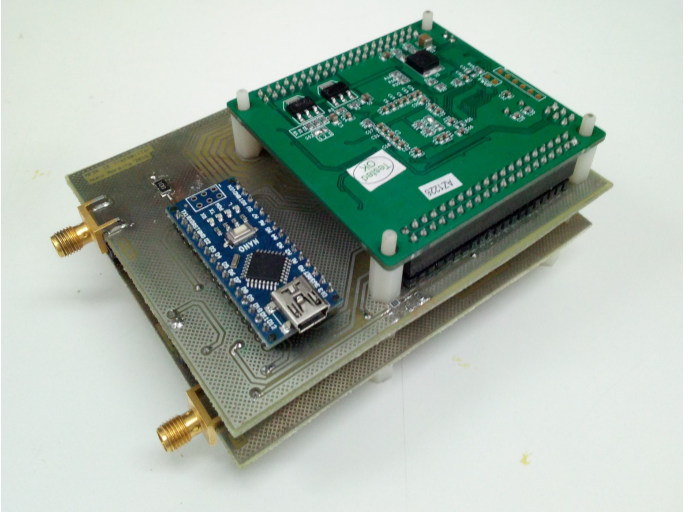

Figure 4.3: MFSK16 Transmitter

4.1.3 Hardware Communication

In order to communicate between the different hardware blocks, since they are slaved by different clocks, it is necessary to establish communication between them using standard communication protocols. The GPS receiver used offers two types of communication, one synchronous, I2C, and one asynchronous, UART. In the case of this work UART was used. Although the I2C protocol presented higher throughputs, 400 kHz, the UART protocol was chosen due to the simplicity of its implementation in the microcontroller and due to the use of the GPS NMEA protocol, which transmits messages in ASCII, and its interpretation is quite straightforward. The baudrate of this communication is 9600 baud and it is possible to be reconfigured. In order to obtain information from the microcontroller a UART interface was implemented with the on-board computer. This interface allows to save locally possible extra information, not transmitted in the protocol. This communication takes place via a USB cable connected between the on-board computer and the microcontroller. This communication is made at 57600 baud. Finally, an SPI interface was implemented between the microcontroller and the FPGA. This interface is used to transmit the tones that will be synthesized by the frequency synthesizer implemented in the FPGA. The microcontroller is the Master and FPGA acts as Slave. The clock of this communication is 4 MHz. This is the main reason for choosing the protocol. The FPGA also has communication with the power amplifier through a digital port. This port turns the power amplifier on and off, thereby lowering the power consumption of the system. A schematic of the described communications can be visualized in the figure 4.4.

4.1.4 Tone Modulator

The tone modulator is described here, although it is implemented in FPGA, due to its importance for the operation of the system. The modulator is implemented as an adder, with a constant defined

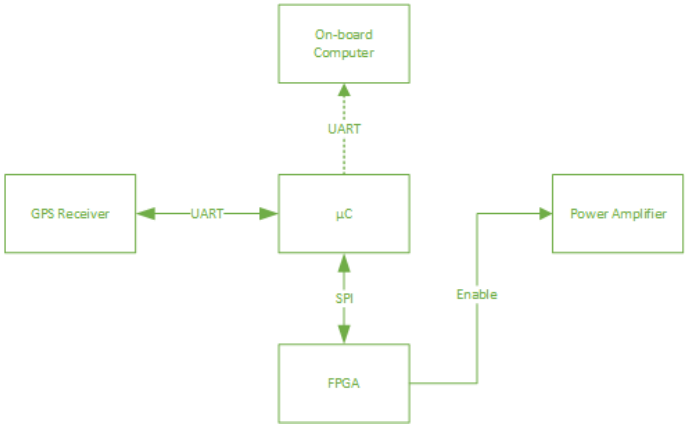

Figure 4.4: Communication Between Different Hardware Parts

taking into account the frequency to be modulated as well as the sampling frequency. The sampling frequency must respect the Nyquist Theorem, which states that the sampling frequency must be more than twice the frequency to be sampled. Taking this into account it is necessary to obtain the constants that must be added. The constant K is obtained by the following equation:

$$K = \lfloor \frac{f}{f_s} \times 2^N \rfloor \tag{2}$$

where f is the frequency to be encoding, fs the sampling frequency and N is the number of bits that are used to encode the constant. N is given by:

$$N = log_2(\frac{f_s}{r}) \tag{3}$$

where r is the minimum resolution. The modulator is an adder, initialized at zero, that when adding the constant exceeds the maximum value of the output of the adder, only the rest of the division of the output with the maximum value of output is considered. That is, if an eight-bit adder is used, and after adding the constant if we obtain 0x153, we consider only 0x53. If this does not occur, the value originated after the sum is considered. The output wave is given by the most significant bit value of the adder.

4.2 Micro-controller

The microcontroller is responsible for controlling the entire system. It is responsible for the initial GPS configuration, coordinate reading and coding of the information, FPGA control and interaction with the on-board computer. In order to perform all these functions, it is necessary to implement several algorithms which will be explained in detail in the following sections. In the figure 4.5 it is possible to observe a flowchart. The purpose of this is to facilitate the explanation of each algorithm, as well as give a temporal perception of the events. In the sections of this subchapter it is also possible to find more detailed flowcharts of the implemented routines. It also explains the approach used in each routine as well as a justification of the use of it. Software development was done in C++ language and using Arduino libraries.

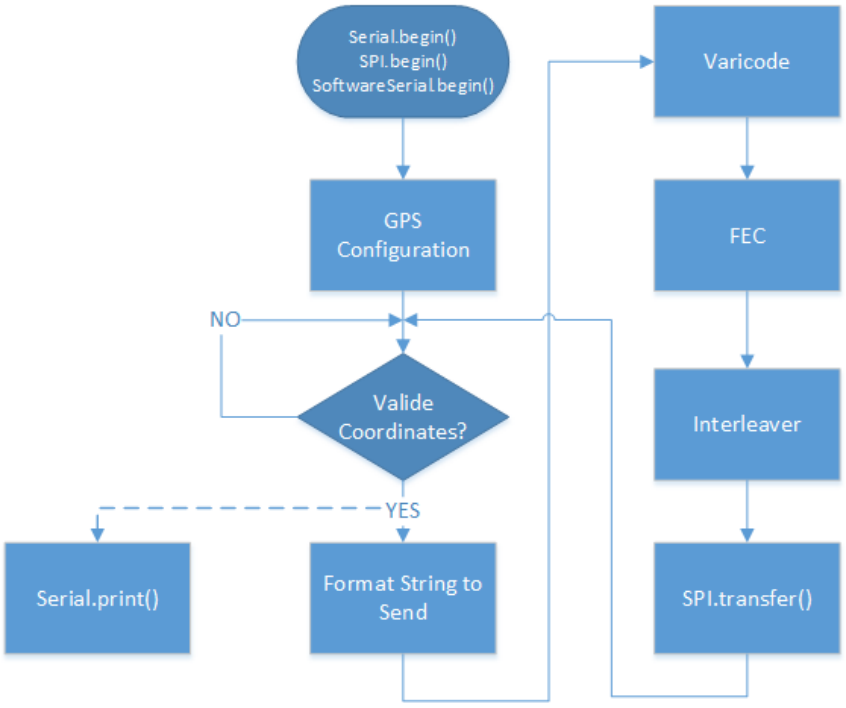

Figure 4.5: Microcontroller Program Fluxogram

4.2.1 UART interface

In order to establish communication between the on-board computer and the microcontroller, an asynchronous serial communication with a baudrate of 57600 is initiated. This communication will allow connecting the whole system to a computer and thus obtaining more information about the system. This communication is not mandatory and is thus dashed in the flowchart of the figure 4.5. The system works the same way even without the existence of this connection. However, it can be used in case of failure of communication with the ground station and consequent loss of some information. Storing this information make it possible to draw a more accurate flight profile.

4.2.2 GPS configuration

The first task of the microcontroller is the reconfiguration of the GPS receiver. This routine was implemented to give reliability to the system and reduce power consumption. This is in case of power failure or in case of a reset of the GPS receiver, it is always possible to maintain communication between the GPS and the microcontroller. The sleep time is also set. Since it is only desirable to send information every two minutes, it is not efficient to have the GPS receiver always on. Thus is established a timer that causes the GPS receiver to wake up every two minutes, get coordinates, send to the microcontroller and sleep again.

For this purpose an Arduino *SoftwareSerial* library is used. This library was used since the version of the microcontroller used, Atmega328-P, only has one serial port. Since the serial port implemented in HW by the microcontroller is responsible for uploading the program to the flash

memory and for sending extra information to the on-board computer then it is necessary to create a second emulated serial port for the GPS receiver. This library is initialized with the choice of microcontroller pins in which to receive and send data, Rx and Tx, as well as the definition of baudrate. The baudrate chosen for this communication was 9600 since it is the default baudrate. So in case of reset it is not necessary to change the GPS baudrate.
To configure the GPS receiver, messages described in the UBX protocol, which can be found in [18], are used. Since the messages are the same on all the initializations they were then saved in flash memory of the microcontroller. Five messages have been defined, which are sent to GPS receiver in the order indicated: the first is *UBX-CFG-CFG*, the purpose of this message is to reset the GPS; the second message is *UBX-CFG-GNSS*, whose purpose is to select the GNSS systems to be used since not all systems are compatible with the power save mode of GPS; the third message is *UBX-CFG-NAV5*, this message configures the GPS to be used above the 18km (60000 feet), since there are restrictions on the use of GPS above this altitude and for speeds above 515 m/s (1000knots); the next message is *UBX-CFG-PM2*, where the maximum sleep time is selected, in which case it is set at two-minute intervals; finally a message is sent, *UBX-CFG-PMS*, that indicates to the GPS that it will be placed in power save mode.
After the GPS configuration, a message is sent via the serial port to the on-board computer with the message "GPS configured" and the microcontroller is held until it receives valid GPS receiver coordinates.

4.2.3 NMEA protocol

The GPS receiver used, uBlox MAX-M8Q, uses the NMEA 0813 protocol, version 4.0. This protocol defines the electrical interface between the GPS, talker, and the microcontroller, listener. It also defines the format of the messages that are sent from the talker to the listener. The characteristics of this protocol are described in [19]. The protocol defines that communication is made using a serial port, previously defined for GPS receiver configuration, baudrate is 4800, although this receiver uses a default baudrate of 9600, eight bits of data and one stop bit, without parity.
The messages from this protocol are initialized with the special character "$" followed by a talker identifier and a sentence identifier. The data fields are separated by commas and the messages are terminated with checksum, carriage return (<CR>) and line feed (<LF>), which have the hexadecimal values 0x0D and 0x0A respectively. Since there are several messages with different information, the first step is to select the message with the information to be transmitted: latitude, longitude and altitude. The message with sentence identifier GGA was then selected. This message has UTC time information, position (latitude and longitude), number of satellites in view, altitude, among other parameters.
The algorithm implemented to extract the relevant information from this message is: the microcontroller is always listening to the output of the GPS, when verifying the receipt of a message with the GGA identifier, checks the field of the time, if it is invalid, that is, it is not filled , the message is ignored and expects new message with the same identifier; if valid, check the coordinates, if there are no valid coordinates, the previous process is repeated; in the case of valid

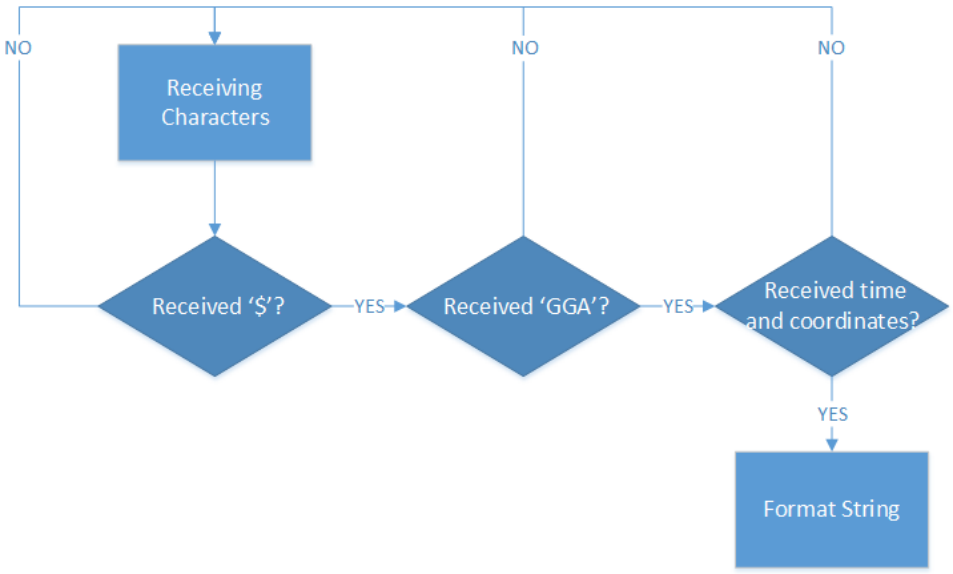

Figure 4.6: Extract Information From NMEA Protocol Fluxogram

coordinates the fields with relevant information are stored in memory and the string described in 4.1.1 is constructed. After defining the string fields, eight NULL characters, with a hexadecimal value of 0x00, are placed at the beginning of the string in order to meet the protocol initialization criteria, referenced in 3.2.6. The figure 4.6 contains a flowchart schematizing the algorithm.

4.2.4 Varicode

The varicode encoding is done by going through the string containing the coordinate and altitude information and mapping each ASCII character to a varicode character. The characters "N", "S", "W", "E" and "M", the algarisms among 0 and 9 and special characters "#", "," and ".", mapping is easily done using a switch-case statement. The values returned by this mapping can contain more than eight bits, so it is necessary to store them in words. Given this and since a word in the microcontroller used contains 16 bits, the routine also returns the bit size used in each word that it is stored in an integer array. The result of the varicode encoding is an array of words and an array of integers, both of 45 elements.

4.2.5 FEC

For the implementation of the FEC coder, the scheme shown in the figure 3.5 was used. The FF chains are initialized with zeros. The words of the array are read and the bits are placed at the input of the shift-register in order from most significant to least significant bit. The array of integers is used to know which bit to read. On the output the bits are placed in an array of bytes, from least significant to most significant. That is, the first output of the shift-register that gives origin to Output1 is placed in bit 0 of element 0 of the array. Output2 is set to bit 1 of element

0, and so on. When element 0 is filled, element 1 is filled in. The total size of this vector is 128 bytes, with only 113 bytes being filled in the worst case.

4.2.6 Interleaver

The algorithm for the interleaver implementation is based on the description made in 3.2.4. The data are all arranged in a array of bytes in which the first bit is in the zero element, in the least bit. In the worst case, the output of the FEC generates 900 bits. The interleaver will never generate more than 1000 bits. This addition of bits is due to the bits with the null value that are added. We start to read the vector from the least bit, every four bits, and start a counter. At the end of ten readings is continued sequentially reading the vector of four bits and merges with the bits read from the first position, also with a 4-bit jump. At the end of twenty readings we begin to read the vector from the second position every four bits and to interleave with the previous ones that continue to be read sequentially. Finally, after thirty more readings, we begin to read the vector from the third element and interleave it with the bits that continue to be read sequentially. This algorithm is schematized in 4.7. The numbers represented are the bit index. Having regard to the scheme, the output data stream from the interleaver is organized from top to bottom and left to right, visible in 4.8.

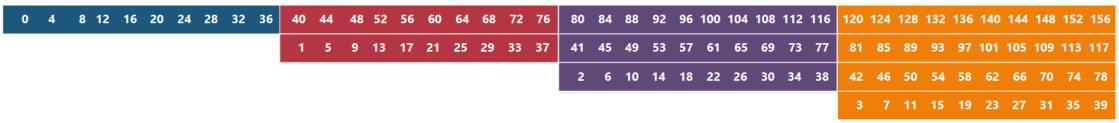

Figure 4.7: Interleaver Schematic

Figure 4.8: Interleaver Output Schematic

4.2.7 Tone Coding

The tone frequencies are encoded in the FPGA in the frequency synthesizer. For this, a 32-bit constant is passed to the FPGA via SPI that will be used in an adder. This constant is obtained through the equations present in the chapter 4.1.4. The number of bits of the constant is 32 so as to obtain a resolution of 0.01 Hz. The sampling frequency of the system is 30 MHz, the tones start at 10.070 MHz, are spaced 15.625Hz among each other up to 10.070234375 MHz. The constants are previously calculated and stored in the flash memory.

To encode the tones to be synthesized, divide the interleaver output into four-bit blocks. After that the bit group is mapped in a tone. The tone number to be sent is given based on the table 3.1. After knowing the tone number the corresponding constant is read, stored in the flash memory, and sent

Table 4.1: Constants To Encode Tones

32-bit Constant	Tone	Bits to encode
0x55EE402B	0	0000
0x55EE48E8	1	0001
0x55EE51A5	2	0011
0x55EE5A62	3	0010
0x55EE631F	4	0110
0x55EE6BDC	5	0111
0x55EE7499	6	0101
0x55EE7D56	7	0100

32-bit Constant	Constant	Bits to encode
0x55EE8631	8	1100
0x55EE8ED0	9	1101
0x55EE978D	10	1111
0x55EEA04A	11	1110
0x55EEA907	12	1010
0x55EEB1C4	13	1011
0x55EEBA81	14	1001
0x55EEC33E	15	1000

via SPI to the FPGA. Initially, the constant corresponding to code 0000 is sent twenty times in order to garantuee the protocol initialization criteria. The output array of the interleaver is then traversed from element zero to element 260 of the least significant bit to the most significant bit. However, the least significant bit corresponds to the most significant bit to code the tone. This is the tones are encoded by the bits of the interleaver output array:

$$b_0b_1b_2b_3;\ b_4b_5b_6b_7;\ b_8b_9b_{10}b_{11};\ ...$$

The 32-bit constant is sent by SPI, divided into four bytes, the most significant byte of that constant being sent first. The constants are present in the table 4.1 with their respective encoding tone and bit code. The transmission is terminated by sending 8 bits with a zero value in order to clear the SPI receive buffer and ensure that all data is correctly received in the FPGA.

4.3 FPGA

In the FPGA, a circuit with well defined temporal constraints is implemented in order to transmit the information defined in the microcontroller through the tone coding. Since the clocks of the two devices are different, a RAM with a well-defined write and read cycle is used. The FPGA receives the constants that will be used in the frequency synthesizer and are stored sequentially in RAM. The information is sent via SPI. During the write cycle in RAM the output of the FPGA is set to low, whereas in the RAM read cycle the output is set to high or low based on the most significant bit of the output of the frequency synthesizer. These cycles are controlled by the microcontroller and the Control Unit via the SS line of the SPI protocol. If the line is low, the FPGA is receiving data from the microcontroller, if it is high, the FPGA is processing the output signal. The circuit implemented in FPGA can be seen in the figure 4.9. It consists essentially of five distinct blocks implemented in Verilog. Two of them, Clock Wizard and RAM, are implemented using Xilinx *ipcores*. The SPI block allows the input of information in the FPGA, the frequency synthesizer allows the output of FPGA information, encoded in the protocol MFSK16. The RAM establishes the bridge between the input and output of the FPGA since the information is transmitted and processed at different frequencies. The bitstream was generated using Xilinx ISE Suite Design 14.7 software.

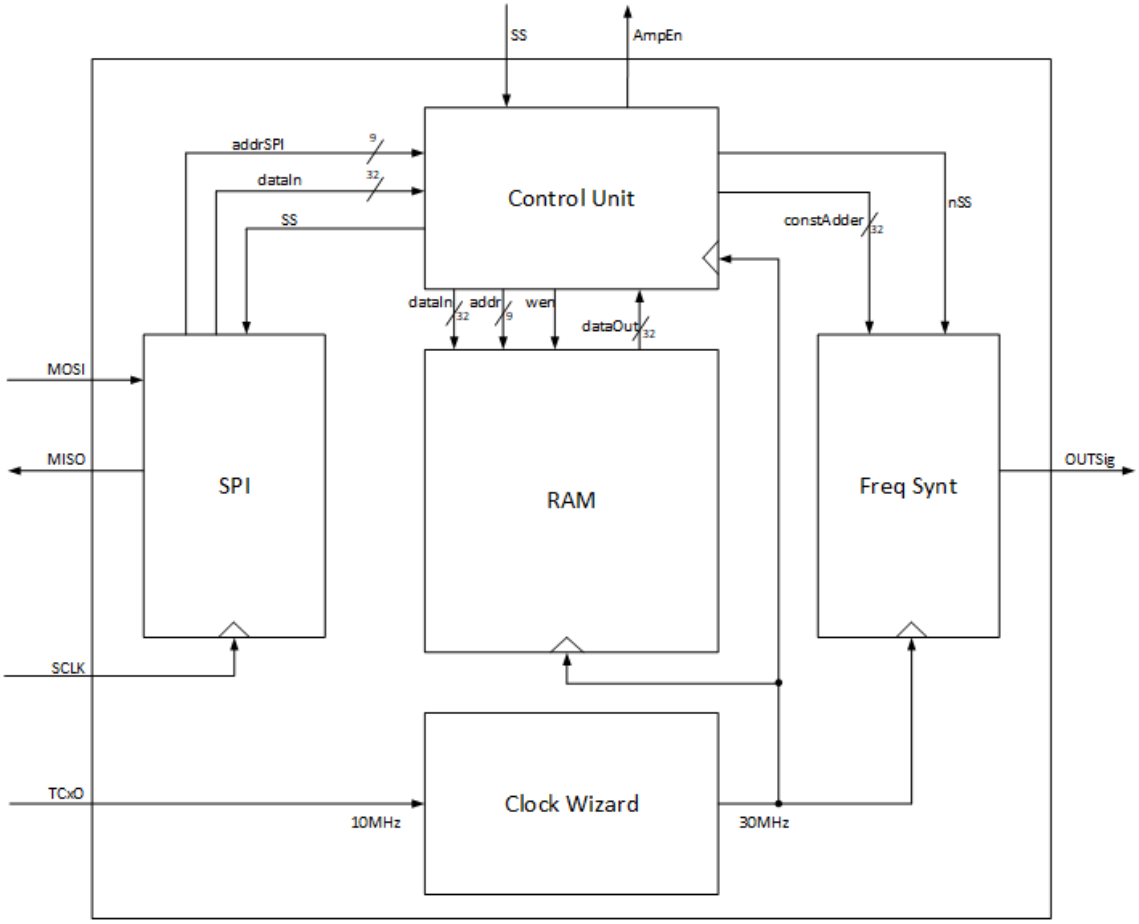

Figure 4.9: FPGA Circuit

The purpose of RAM is to store all constants sent by the microcontroller and used by the frequency synthesizer. In the worst case, 490 constants of 32-bit length are sent. To ensure that all constants are written, a RAM block with 512 32-bit memory locations has been created. To write in this memory the input write enable, *wen*, shall be placed in high voltage, is placed a 9-bit address in the input *addr* and the value to be saved is placed in the input *dataIn*. The data is written in memory when the clock signal goes high. The value is only available for reading on the next clock cycle. To read from memory, without changing the value of a given position, the input *wen* should be low, an address in *addr* is set, and a clock cycle after the output *dataOut* is available. If the input *wen* is high the value present at the given address position is read correctly, however it is modified with the value present in *dataIn*. It is a *NO-CHANGE* memory. That is, when a write occurs the output value is not modified until the next clock cycle. Since the FPGA acts as slave in SPI communication, during the writing cycle in RAM the clock used is given by the SCLK protocol line SPI. This clock is given by the microcontroller, since it is the master. Although the data transfer between microcontroller and FPGA is done at 4Mbit/s, the clock of 30 MHz in the RAM is used. The clock is faster than the reception of the constants causing the writing of the same constant in the same memory location. This does not prejudice the normal operation of the

system. The purpose of the clock wizard block is to multiply the TCxO output frequency that enslaves the FPGA and whose oscillation frequency is 10 MHz. The sampling frequency of the HF transmitter is 30 MHz. This frequency is obtained at the output of the clock wizard block. The block input is *CLK_10MHz* and the output is *CLK*.

4.3.1 SPI

In order to receive data via SPI a shift-register was implemented, using MOSI as input signal, MISO as output signal, SS as high reset and SCLK as clock. Four counters were also implemented that control the data that will be written into RAM. One of the counters counts incoming samples. This counter is used for each eight input bits to be sampled on one of the eight-bit registers. Another of the counters is incremented by eight samples in order to select in which register these samples will be stored. Finally a counter is used that every 32 samples increments another counter that is responsible for giving the RAM address where the value will be stored. The implemented circuit can be seen in the figure 4.10.

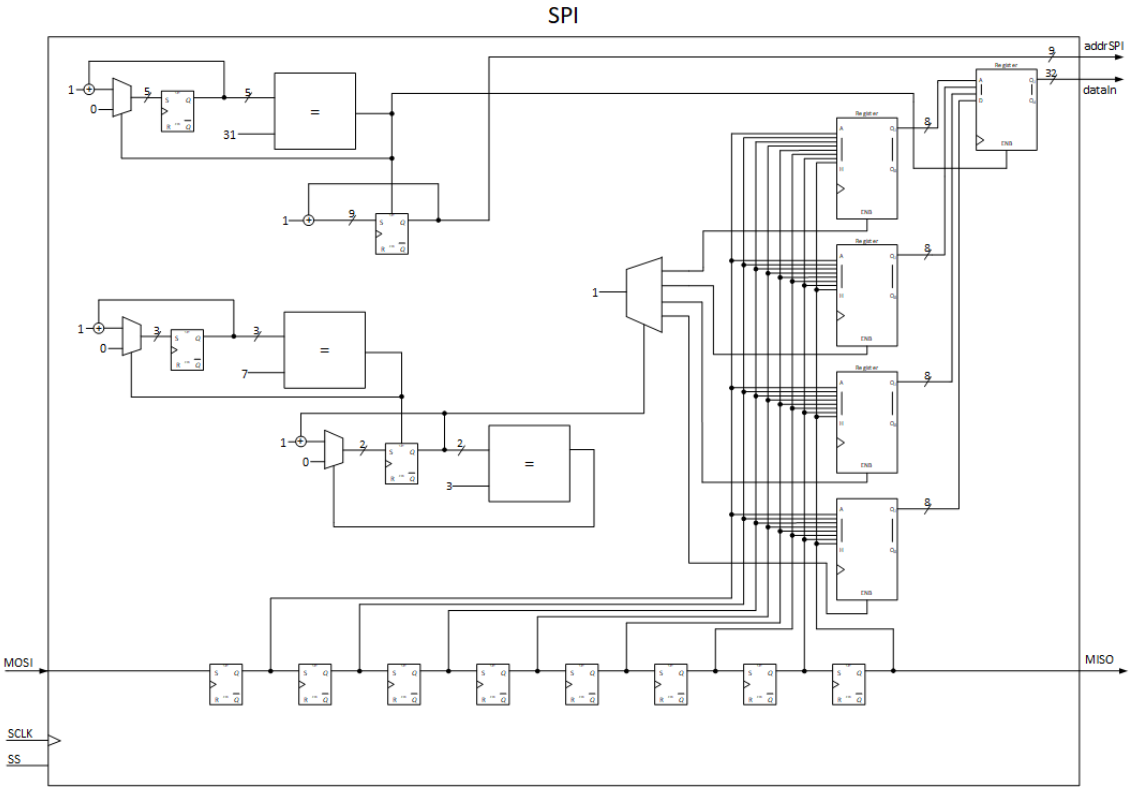

Figure 4.10: SPI Block

4.3.2 Frequency Synthesizer

The frequency synthesizer, represented in the figure 4.11, is composed of an adder that at each cycle increments the constant received from the RAM and a comparator that verifies if the result of the sum does not exceed 0xFFFF, which is the maximum value obtained in 32 bits. In case of exceeding 32 bits the constant 0xFFFF is subtracted from the result of the sum. The output of this block is the most significant bit at the output of the multiplexer. This bit indicates whether we are at the beginning of the period or at the end. The purpose of this block is to vary the input constant so that the output wave varies in frequencies. If the constant is smaller, the period of the output wave will be larger, otherwise the frequency will increase. The constants used are given by the RAM block, and the frequency generated is transparent to the frequency synthesizer. That is, it is possible to generate any frequency with this circuit, being necessary only to define the sampling frequency, CLK, and the input constant, which will give the frequency of the output signal. Taking the constants obtained in 4.2.7 all the frequencies necessary for the coding of the MFSK16 protocol are obtained.

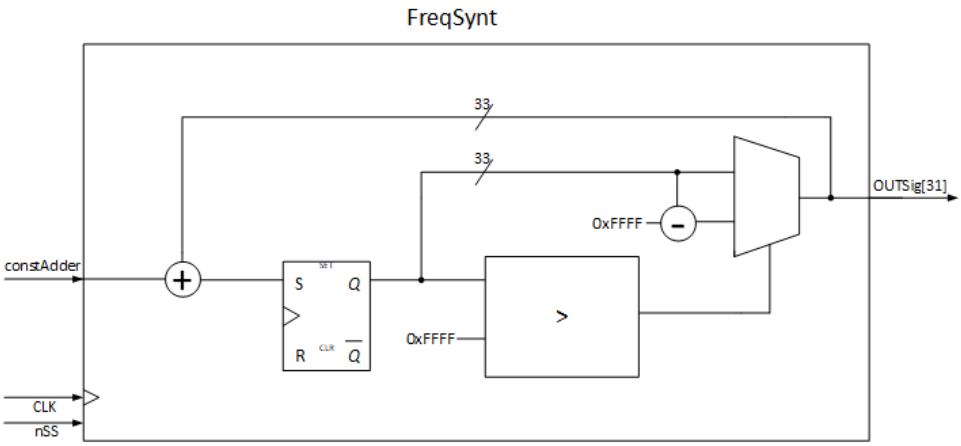

Figure 4.11: Frequency Synthesizer Block

4.3.3 Control Unit

The control unit shown in figure 4.12 is responsible for defining the RAM inputs as well as the inputs of the frequency synthesizer block. This control is mainly given on the basis of the SS value. When SS is high, *addr* is given by the internal counter. When SS is low the value *addr* and *dataIn* is controlled by the values given by the SPI block. The maximum address of the RAM is 511, so the maximum value given by the adder is 511. This value is incremented every 192000 clock periods, at 30 MHz, corresponding to 64 ms. The SS value is also inverted to be used as reset of the Frequency Synthesizer block.

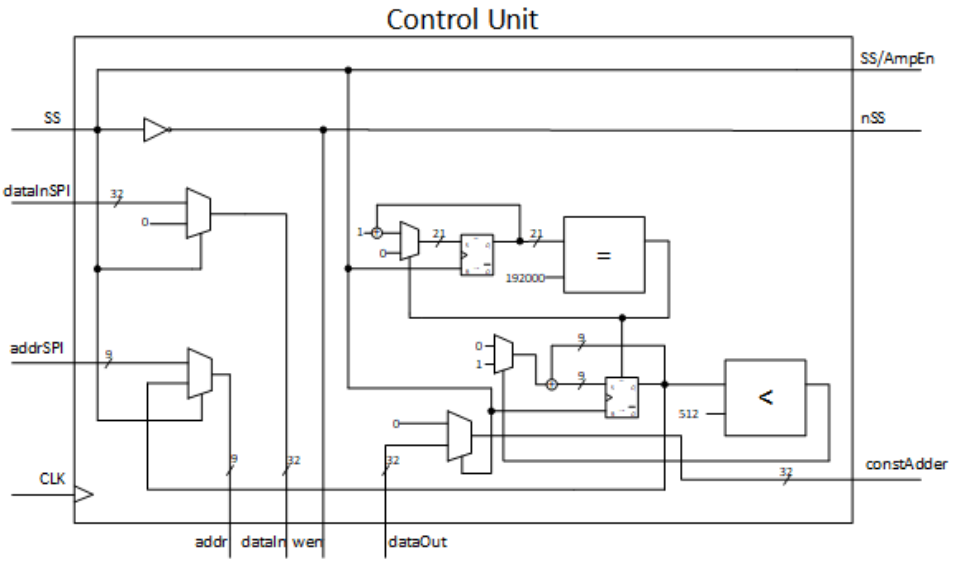

Figure 4.12: Control Unit Block

4.4 Power Amplifier

The amplifier is built on two levels: a preamplifier in order to increase the excursion of the output signal of the FPGA and by a power amplifier whose objective is to obtain current gain. The FPGA output signal is a square wave with frequencies very close to 10 MHz with 3.3 Vpp. In order to increase the signal excursion a common source MOSFET configuration is used, gaining approximately 2.7 times, trying to obtain an output signal very close to 8 Vpp. To increase the current a configuration with two pairs of BJT in common collector is used. Each pair of BJT consists of a NPN BJT and its PNP complement. The amplifier was scaled using NI Multisim software where it was possible to simulate and optimize its behavior. The scheme of the circuit can be observed in the figure 4.13.

Simulation blocks were used: oscilloscope and multimeter to observe the current and voltage output waves of the amplifier and the power consumed by the amplifier. The waveforms can be seen in 4.14. The blue line corresponds to the voltage checked at the output and the green line corresponds to the AC current at the output. It is possible to verify that the maximum voltage value is 6.48 Vpp and the current is 129 mA. The power consumed by the circuit is approximately 0.95 W.

4.5 PCB Design

After choosing the electronic components, PCBs were designed using NI Multisim / Ultiboard software. This software was chosen due to the existence of licenses in the faculty for the use of it. Multisim is used for a schematic representation of the circuit while Ultiboard represents the circuit physically.

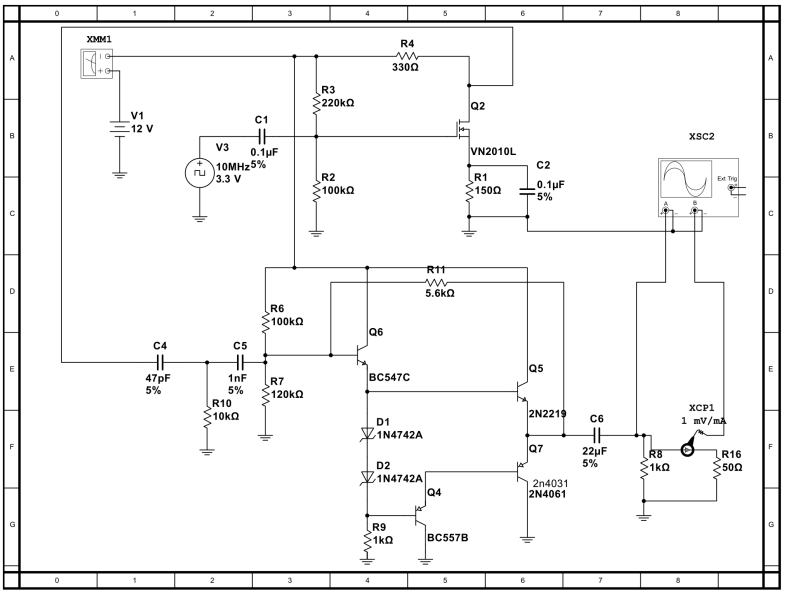

Figure 4.13: Amplifier Schematic

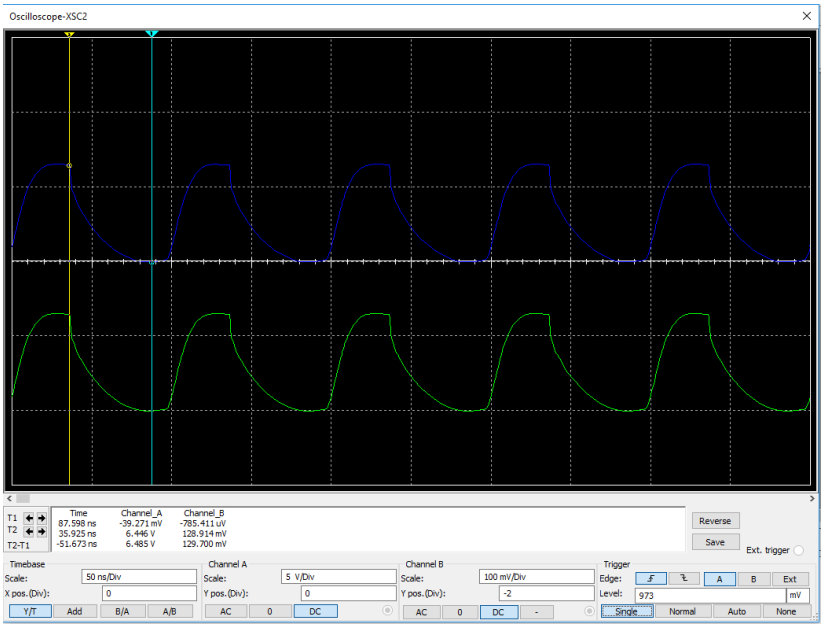

Figure 4.14: Amplifier Waveform (blue line: voltage; green line: current)

The first step in PCB design was the design of the component footprints and the definition of the schematic block for circuit design. The circuit was then schematized generating a netlist that was exported from Multisim to Ultiboard. This netlist serves to establish connections between components as well as to establish Design Rule Check (DRC) constraints. The components were placed and a routing between them was done. The line widths were taken into account, using thicker

tracks whenever it was possible, while never attempting to increase the total area of PCBs. The minimum width used on the lines was 0.30 mm. Ground planes were designed in the free area of the PCB in order to decrease the RF disturbances.

Two PCBs were designed, one with the power amplifier and another one that is used as motherboard of the microcontroller and FPGA. They are designed to be assembled in stack and there are links between them. The entire system is powered at 12V, this voltage being converted to 5V, which supplies the FPGA and the microcontroller. There are also components that are powered to 3.3V, such as TCxO, GPS receiver and active GPS antenna. This voltage is supplied by the microcontroller, which has an internal voltage regulator capable of supplying sufficient power to supply these components. There are 4 connection pins between the two PCBs, two for power, 5V and GND, and two other for enable of the amplifier controlled by the FPGA. PCBs are secured using nylon spacers. These spacers give mechanical hardness to the stack. The nylon was chosen because of its non-conductive properties and the ability to absorb impact during landing. Analog signal connections between PCBs are made using SMA cables, as well as antenna connections. This is due to the noise immunity of the SMA cables. The designed layout are displayed in figures 4.15 and 4.16 as well as the manufactured PCBs. The ground planes were omitted in order to better understand the designed layout.

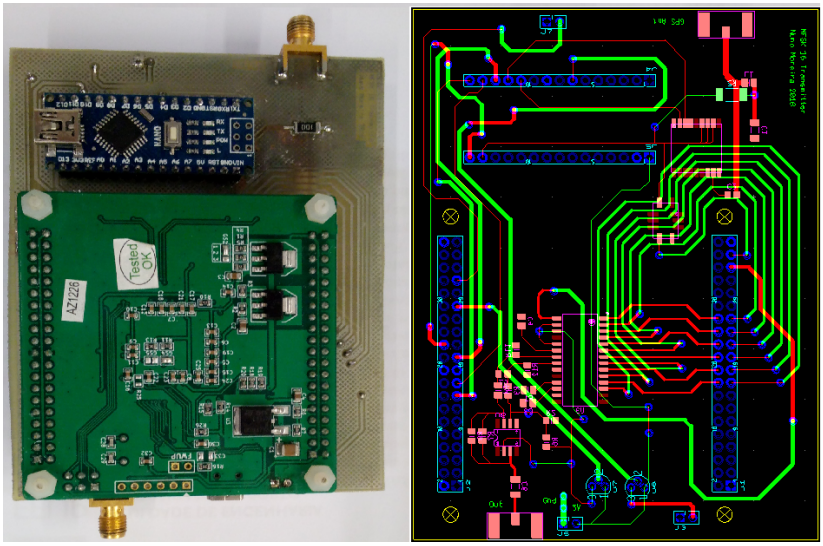

Figure 4.15: Motherboard Layout

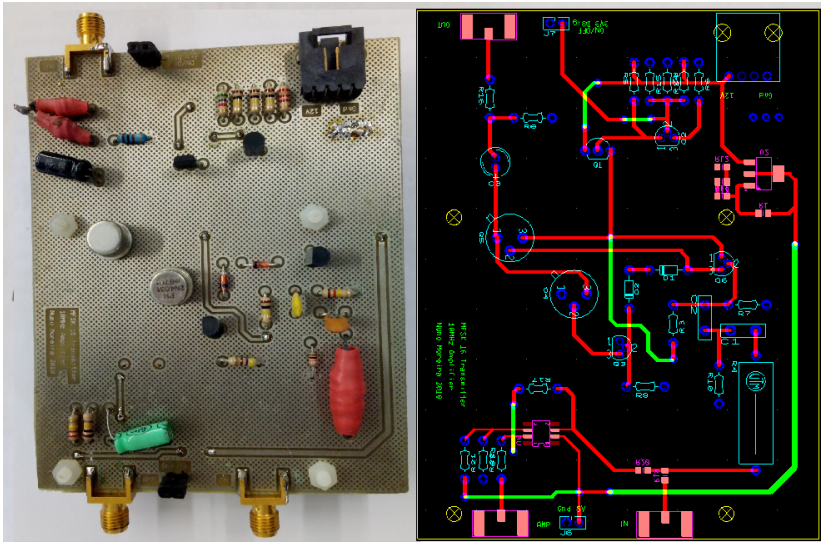

Figure 4.16: Power Amplifier Layout

Chapter 5

Tests and Results

This chapter refers to all the tests that have been performed in order to verify and characterize the operation of the system. It consists of three subchapters. Each subchapter concerns tests carried out at a particular stage of system implementation, with the first subchapter dealing with an earlier implementation phase and the third with the total functioning of the system.

5.1 Protocol Debugging

During the implementation of the protocol it was necessary to verify each coding block and its correct functioning. Initially, the communication between the GPS receiver and the microcontroller was tested. Several configurations were tested until the final configuration was achieved. In order to test the power save mode of the GPS, a multimeter measuring the DC current was used. It was verified that in sleep the GPS consumed less power, as expected. Some coordinates were also received and the correct functioning of the GPS was verified on a map.
After this verification and definition of the string to be sent, the protocol began to be implemented. The routine that implements the varicode encoding has been tested by exporting the characters of several binary messages via serial port and stored in a text file. The characters were compared visually with the dictionary present in [10]. Then the FEC coder was implemented. In order to verify its correct functioning it was implemented a Verilog block implementing a similar shift-register. It was made a behavioral simulation and compared the outputs. Since the interleaver implementation is not trivial, the routine was implemented in the microcontroller and the output was placed in a text file. The verification was done using the first 50 values and comparing them with the values obtained manually. The fact that it was manually tested avoided programming errors. The SPI communication was tested using an oscilloscope and checking the waveforms at the output of the microcontroller.
From the FPGA point of view, several test benches were created to simulate the behavior of each block. Random stimuli were placed at the inputs of each block and the outputs were checked. The waveforms were observed in order to ensure the proper functioning of the system. To simulate the SPI block, binary values were placed on the input and the 32-bit constants that have output to

the RAM were checked. The frequency synthesizer was tested. The output wave was exported and the signal frequency was checked. Given the high number of data, only 500 samples were considered. The clock-wizard and RAM blocks were not simulated since they are implemented in Xilinx *ipcore*. The control unit was also simulated.

5.2 System Simulation

After the implementation of all microcontroller routines and FPGA circuits, a complete simulation of the system behavior was performed. The microcontroller received information via GPS, formatted the string and was exported via serial port. The data were coded and instead of being sent via SPI they were also exported via UART. The exported string and bits have been saved in text files. A test bench was created that received the bits exported from the microcontroller. In this test bench was implemented an SPI protocol, whose clock operated at 4 MHz and the MOSI line had as input data the generated data stream given by the microcontroller. The output waveform of the test bench was exported and saved in text file. The system was simulated in order to obtain the complete waveform that encoded the string. It was estimated that the simulation time required for the effect would be about 30 seconds, corresponding to a real time of approximately 10 hours. The amount of data generated was quite high, approximately 3GB, and difficult to analyze. To verify the correct functioning of the system, the data were imported into MATLAB in small blocks, and were placed in baseband, shifted a 1kHz. A low pass filter was applied followed by an decimation, $M = 100$. At the end, the signal was reconstructed in the audible frequency band and saved in a wav file. The resulting signal was played back on the computer and decoded through MixW 3.2. The obtained signal was decoded without any problem, being able to reconstruct the encoded string, validating the correct operation of the system. It was also generated an audio file by the MixW 3.2. The exported string was encoded using the software. The spectrograms and sounds were compared and it was verified that the resulting sounds were very similar.

5.3 Field Tests

In order to perform field tests, the power amplifier was initially tested. Square and sinusoidal waves were applied to the input of the amplified and verified the operation of the same. It was possible to note the impact of the amplifier on the tests performed.
The transmitter antenna was also tested in order to ensure its efficiency. It was noticed that the antenna is not optimized. The reflection coefficient was measured using a vector network analyzer and it was verified that the S11 parameter is above -10dB. The reflection coefficient is presented in figure 5.1.

The tests were performed during the afternoon and from different locations. Initially the system was tested about 5 meters from the receiving antenna. It was found that it was possible to successfully receive and decode the information. The reception frequency was shifted and an attenuation of 20dB was set and the signal was still decoded. It is possible to see in figures 5.2 and

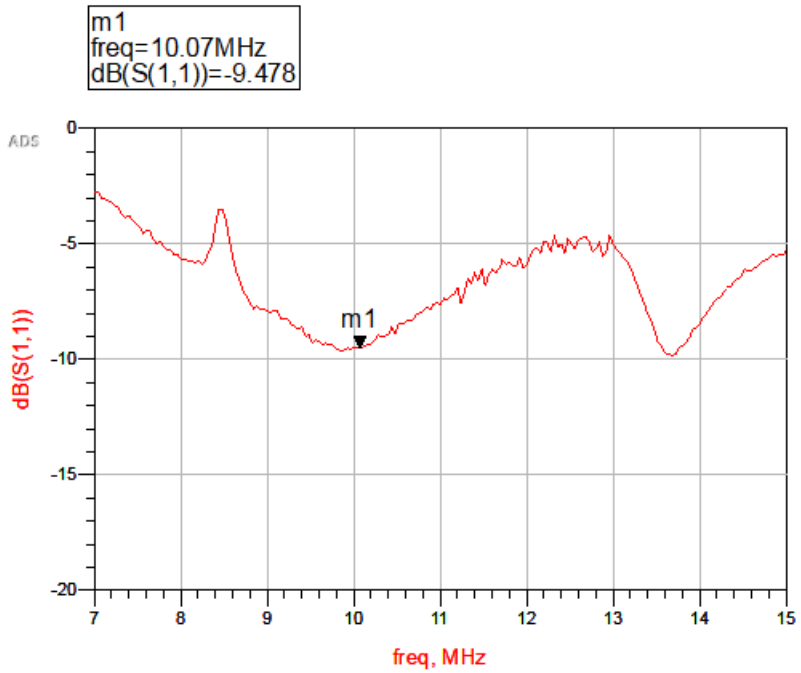

Figure 5.1: S11 Parameter of Tx Antenna

5.3 the received signals with respective attenuation. It is also possible to read the decoded string in the figures. It was also possible to test the system without the power amplifier. It was possible to verify that even if it received a very low signal level, it was still possible to decode it successfully. The result of this test can be checked in figure 5.4.

Then the system was tested about 300m from the receiver. It was noted that the information was still decoded. However, the test was not successful. The signal power received was too low and did not meet the results of the previous test. In order to understand the problem, the SWR of Rx antenna was measured and it was possible to verify that the antenna was unbalanced. Considering the 10MHz frequency the SWR of the Rx antenna was about 6. The reflection coefficient was also measured. The results are shown in the figure 5.5. It was possible to verify that the antenna is matched to frequencies between 48.34MHz and 59.80MHz.

In order to go further in the tests it is necessary to optimize the Rx antenna. After the optimitization it is intended to test the system in different locations. In the map shown in the figure 5.6, obtained using Google Earth, it is possible to see some different locations identified as: FEUP Parque, Salgueiros, Estádio Dragão, Parque Nascente and Escola Rio Tinto. The straight distances measured through the software used are: 0.31 km, between Parque FEUP and NRA, 0.93 km between Salgueiros and NRA, 1.92 km between Estádio Dragão and NRA, 2.64 km between Parque Nascente and NRA and 3.86 km between Escola Rio Tinto and NRA. Since the distances from the location of the tests to the receiving site are quite small, only the direct wave is tested, not the

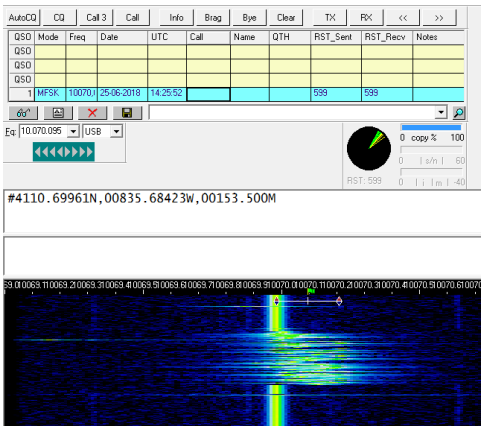

Figure 5.2: Received Signal, No Attenuation

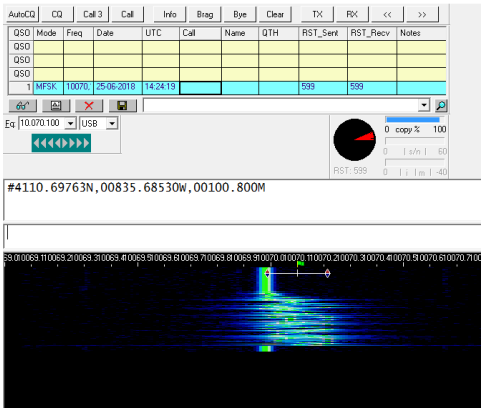

Figure 5.3: Received Signal, 20dB Attenuation

waves reflected in the Ionosphere. Nevertheless, it is possible to characterize the system as well as its potential range. It is also intended to test the system in the real environment. It is intended to preform a STRAPLEX flight in order to characterize the whole system.

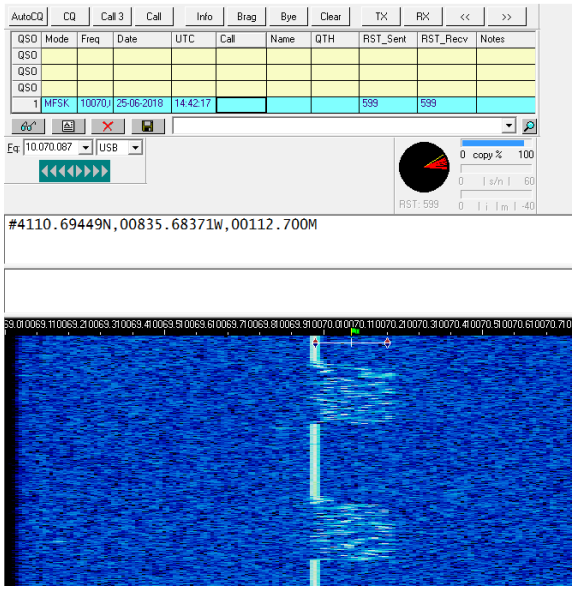

Figure 5.4: Signal Transmitted Without Power Amplifier

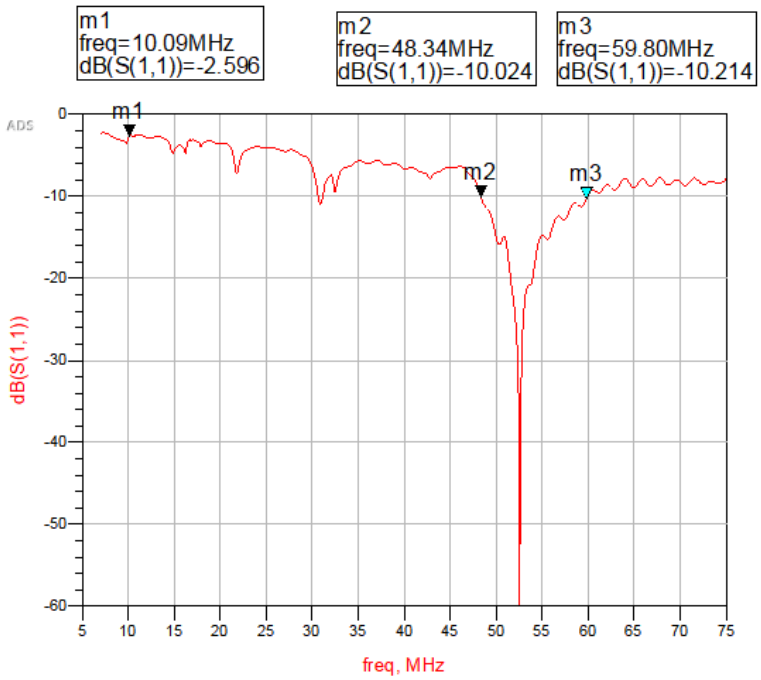

Figure 5.5: Reflection Coefficient of Rx Antenna

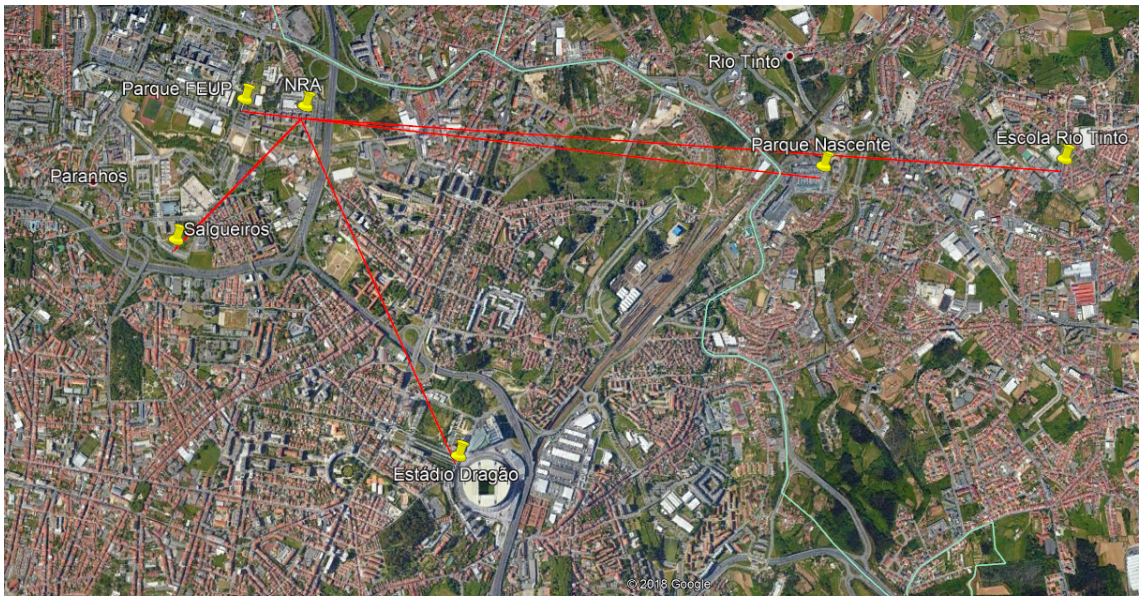

Figure 5.6: Tests Locations

Chapter 6

Final Remarks

This chapter describes the main conclusions drawn from this work. The subchapter 6.1 lists some difficulties presented during the work developed and subchapter 6.3 lists some points to be improved in the future.

6.1 Difficulties Encountered

During the development of the thesis several technical challenges were encountered. I emphasize the difficulty in understanding the protocol and its characteristics. The understanding of the protocol was fundamental to the implementation of the system and the lack of updated information caused a delay in the accomplishment of this task. However, after understanding it, the implementation was straightforward. Another obstacle was the analysis of the data resulting from the simulations. The amount of data generated was very large and difficult to analyze, and algorithms have to be used to split the file into smaller blocks. In addition to these technical challenges, challenges have also been encountered at the bureaucratic level. At the hardware level, the work was substantially delayed, not only due to delays in the delivery of electronic components such as FPGA, which delayed the development process by about a month, as well as a failure in a drill machine responsible for drill the PCBs that delayed in one week the implementation phase.

6.2 Conclusions

HF communications are mainly used for information transmission over long-path. This document describes the implementation of an HF transmitter as well as the characteristics of the protocol used, MFSK16. It is intended to use the HF transmitter on future STRAPLEX flights. It is intended to be included in the hardware of a STRAPLEX capsule, receive GPS coordinates, encode them and transmit to the ground station. These coordinates will be used for the location of a STRAPLEX capsule throughout the flight to be possible to retrieve it.
During the implementation of this transmitter, software routines were implemented, as well as hardware blocks. These blocks were used to comply with the characteristics of the protocol. Tests

were performed in order to validate the protocol. Although the number of tests to validate the protocol coding was high, the number of field tests was insufficient. It was possible to test the encoding of several strings and it was possible to decode all of them. The lack of field tests is due to the fact that the receiving and transmitting antennas are not well calibrated for the transmitted frequencies, and there is a loss of signal energy when the distance between the transmitter and the receiver is increased.
Despite these problems, it is expected that after solving this problem more tests will be conducted and the system will be validated. The transmitted power is high enough for line of sight transmissions at a distance of tens of kilometers, since during a test an attenuation of 20dB was applied and the signal level was still quite high.
From the point of view of the implementation of the protocol the main objectives have been achieved, but more tests are required in order to be able to use only this localization system in future STRAPLEX flights, instead of the currently used system, VHF data transmission using protocol APRS.

6.3 Future Work

This system is intended to reduce the mass and power consumed in a STRAPLEX capsule. The purpose of this transmitter is to give reliability to the telemetry system allowing the focus to be centered during the flight in the data collection. It is also intended to be more comfortable launch, not having to follow the trajectory of the capsule throughout the flight, being possible to get coordinates constantly on the ground station. However there are some points that can be improved:

1. Optimization of the power amplifier - with the tests it was possible to verify that the output power of the amplifier was below expectations, to test another assembly or to optimize the circuit used will increase the transmission distance

2. Optimization of the Tx and Rx antenna - during the tests it was possible to verify that the transmitting and receiving antennas were not optimized for the range of frequencies used; using better antennas it is possible to obtain quite better results

3. GPS Reconfiguration and Tx Frequency - create routines that reconfigure GPS as needed; change the frequency of transmission taking into account the period of the day on which it is being transmitted and the distance from the landing site to the ground station in order to maximize the signal quality received

4. Test in real environment - Run multiple tests with varying distances, including a launch to validate system operation

References

[1] Scott Honnaker. *The ARRL Handbook For Radio Communications*, chapter 16, pages 16.1–16.30. ARRL, Newington, Connecticut, 88 edition, 2011.

[2] Steve Richards. A pratical evaluation and comparison of some modern data modes. pages 4–62, 2003.

[3] Steve Melts. The "New" HF Digital Modes. *QST*, pages 50–51, April 1999.

[4] Peter Martinez. PSK31: A New Radio-Teletype Mode. pages 3–9, July/August 1999.

[5] Steve Ford. PSK31 - Has RTTY's Replacement Arrived? *QST*, pages 41–44, May 1999.

[6] Steven L Karty. PSK31 Spec. http://www.arrl.org/psk31-spec. Acessed in February, 2018.

[7] Murray Greenman. MFSK16. http://www.qsl.net/z/zl1bpu//MFSK/M16.htm. Acessed in February, 2018.

[8] Murray Greenman. MFSK16 Technical Specification. http://www.qsl.net/z/zl1bpu//MFSK/tech.htm. Acessed in February, 2018.

[9] Murray Greenman. MFSK for the New Millennium. *QST*, January 2001.

[10] Nino Porcino and Murray Greenman. The IZ8BLY MFSK Varicode. http://www.qsl.net/z/zl1bpu//MFSK/Varicode.htm. Acessed in February, 2018.

[11] Murray Greenman. MT63. https://web.archive.org/web/20081215124710/http://www.qsl.net/zl1bpu/MT63/MT63.htm. Acessed in February, 2018.

[12] Murray Greenman. MT63 modes. http://www.w1hkj.com/FldigiHelp-3.21/Modes/MT63desc.htm. Acessed in February, 2018.

[13] Murray Greenman. MT63 technical description. https://web.archive.org/web/20081217173125/http://www.qsl.net/zl1bpu/MT63/Technical.htm. Acessed in February, 2018.

[14] Dennis J. Lusis. HF Propagation: The Basics. *QST*, pages 11–15, December 1983.

[15] Nino Porcino and Murray Greenman. The MFSK FEC Coder. http://www.qsl.net/z/zl1bpu//MFSK/FECcoder.htm. Acessed in February, 2018.

[16] Nino Porcino. THE IZ8BLY DIAGONAL INTERLEAVER. http://www.qsl.net/z/zl1bpu//MFSK/Interleaver.htm, January 2000. Acessed in February, 2018.

[17] Nick Fedoseev and Denis Nechitailov. MixW. http://www.mixw.net/index.php. Acessed in January, 2018.

[18] uBlox AG. ublox 6 Receiver Description. https://www.u-blox.com/sites/default/files/products/documents/u-blox6_ReceiverDescrProtSpec_%28GPS.G6-SW-10018%29_Public.pdf. Acessed in February, 2018.

[19] Thomas Anderssén. The NMEA 0183 Protocol. http://www.tronico.fi/OH6NT/docs/NMEA0183.pdf. Acessed in March, 2018.