TELECOMMUNICATIONS
S9 PROJECT

S9 Project: Final Report

Students:
Elias DELMASURE–DHAENE
Rémi HINSCHBERGER-BOCHER
Alexandre VALADE
Guillaume AYRAULT
Yahya MESMOUDI

Supervisor:
Romain TAJAN

22 January 2023

Table of Contents

List of Figures

1 Introduction

Today more than ever, the security of wireless information transmission is an important issue. Most of the proposed solutions are based on encryption of information, but there is another way to transmit information securely. This solution is based on polar codes. [1] In this project, the goal was to create a demonstrator capable of creating a secure communication between two interlocutors without encryption. [2]
The demonstrator is constructed as follows: a first person (Alice) transmits a message to a second person (Bob) while a spy (Eve) tries to intercept this communication (Figure 1). This demonstrator illustrates that under certain conditions, Bob can perfectly receive the message while Eve only receives noise. This report details the steps taken to make the demonstrator work, starting with a definition of polar codes and the wiretap channel.

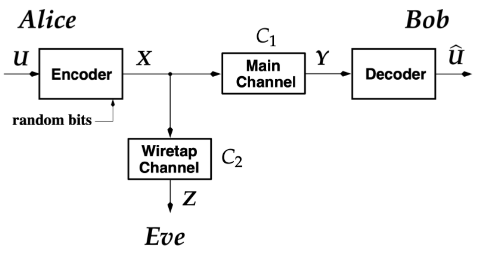


Figure 1: The wiretap channel

In an increasingly connected and virtualised world, where our personal information and interactions also have a digital dimension, and given that wireless connections are easily intercepted, it is crucial to have secure communication systems.

2 Polar Codes and Wiretap Channel

2.1 Polar Codes

Polar codes are a class of error-correcting codes that can be used for reliable transmission of data over noisy channels. They were invented by Arikan in 2008 and have demonstrated their capability for a wide range of channels, including the additive white Gaussian noise channel (AWGN). Polar codes work by exploiting the symmetry of the channels over which they are transmitted. The key idea behind polar codes is that certain channels are "better" than others in the sense that they are more reliable for transmitting information. These effective channels can be identified using a channel polarization process, and the information can be encoded in such a way that it is transmitted over the effective channels while noise-free dummy bits are transmitted over the ineffective channels.

Channel polarisation is achieved through a combination of bits. The schema depicted in Figure 6 is repeated as many times as required, each time creating a degraded channel (the one on the top in the figure), and an improved one. The other channel is improved in terms of reliability of the decoding, due to the concept of *frozen bits*. If a channel is degraded to the point that it cannot be reliably decoded, it will be set to 0 on both the encoder and the decoder sides. Using this information then helps predict the bit on the other channel, since the two input bits u_1, u_2 and the frozen bit sum to 0. This new knowledge improves the probability of correctly decoding the bit on the second channel.

In the following sections, *frozen bits* are referred to as ineffective bits, or ineffective channels.

2.2 Wiretap Channel and Weak Secrecy

A wiretap channel is a communication channel in which a person eavesdropping (the "wiretap") can overhear the transmission. The eavesdropping channel is a variation of the standard communication channel, with the additional constraint that the transmission can be overheard by an adversary. The objective in this scenario is to design communication codes and strategies that maximise the information transmitted to the intended receiver while minimising the information transmitted to the eavesdropper. In this case, the objective is to use polar codes to successfully transmit information to Bob without Eve being able to decode it. To do this, the information must be transmitted on Bob's effective channels which are ineffective for Eve as illustrated in Table 1.

EVE / BOB	Effective channel	Ineffective channel
Effective channel	Noise	Information bits
Ineffective channel	Frozen bits	Frozen bits

Table 1: Information bits and frozen bits distribution table

Thus, by using the definitions of effective and ineffective channels given in Section 2.1 and knowing the estimate signal to noise ratio for both Bob's channel and Eve's, one can deduce which channels will be both effective for Bob and ineffective for Eve.

2.3 Strong Secrecy

Despite being a good starting point, weak secrecy has a major flaw. If Eve decides to still listen on her bad channels, there might be some where she will still be able to read part of the information, depending on the quality of her reception.

Strong secrecy forbids this behaviour by doing another selection on information channels. By using a quantity called Bhattacharyya's distance, it is possible to select only channels where the probability for Eve to be able to read any data is near 0, thus making it almost impossible for data to leak.

This method's main downside is that one is not guaranteed to find a channel on which they can communicate while the eavesdropper can never decode anything. This criterion is very strict and filters out most of the potential channels, which also reduces the communication rate.

3 Contributions to the Subject

In order to create a demonstrator for the wiretap channel and weak secrecy, a few tools are used.

Material-wise, two USRP-B100 radios are used. One serves as the transmitter, and the other as the receiver. In order to switch between Bob and Eve for the receiver, a loss in received power is virtually added.

Software-wise, the communication chain is implemented using the AFF3CT library [3]. It is an open-source C++ library focused around error correction, with a compatibility-mode for PYTHON called PY-AFF3CT. Both languages are used in this project, the former being used sparsely for performance-dependent tasks, while the latter is used to wrap the communication chain.

The developed demonstrator is shown in Figure 7. The following section introduces the custom modules programmed for this project.

3.1 SPLIT Module

The Split module (Figure 8) serves as a source for the sequence. It takes a binary sequence and splits it into frames of a given length. The decoded frames are also stored by the Split module at the end of the sequence. If the whole sequence has been sent, the Splitter loops back to the beginning and continues sending.

3.2 MUX Module

The MUX module is designed to address the unique characteristics of polar codes when used in a wiretap channel. In this scenario, the transmitted information should be easily accessible to Bob, the intended recipient, while being difficult for Eve, the eavesdropper, to decode. There are additional specificities whether it is weak or strong secrecy.

3.2.1 Multiplexer

The *multiplexer* block (Figure 9) is located between the source and the encoder. The inputs of this block are the data to transmit. Those data are split in smaller portions of size **K**. For each portion, the bits are intertwined to a specific position in a random signal of size **N** (**N**>**K**). The intertwined signal is the output of the *multiplexer* and is sent to the encoder.

3.2.2 Demultiplexer

The *demultiplexer* block is located between the decoder and the monitor. This block's inputs are the decoded, intertwined signals. For each of those signals, the *demultiplexer* retrieves the estimated data at the exact same positions. Those estimated bits will then be compared in the monitor to compute metrics such as the bit error rate (BER) or the information leakage.

3.3 PAD Module

The PAD module is used in the communication chain to add bits to the frame and meet the DVBS-2 standard requirements of 16200 bits per frame. It is initialized with the amount of meaningful bits and the size of the padded frame. It packs one function to pad, and another to unpad.

3.3.1 Pad

The *pad* block (Figure 10) appends random bits to the encoded message in order to meet a goal size for the communication standard. In the current case, this size is 16200 bits. These padding bits are never read by the receiver so their value is not important.

3.3.2 Unpad

The *unpad* block does the opposite of the *pad*. It uses the initial size given to it to extract the sequence of meaningful bits from the random bits padding.

3.4 Mutual Information

One of the metrics used in this project to illustrate the ability of a receiver to decode the right data is the mutual information between the transmitter and the receiver. This computation is done by comparing each decoded bit on the receiver's end to its counterpart in the original transmitted sequence. The mutual information is then deduced from the estimated probability distributions using equation 1, in Appendix B. The closer both distributions are, the more the mutual information approaches 1, whereas it is close to 0 if the distributions are uncorrelated.

3.5 Transmitter

The transmitter comprises of various components (Figure 2) that enable the data to be read and processed efficiently to ensure secure transmission and optimal synchronization with the receiving radio. (Figure 11 is more detailed)

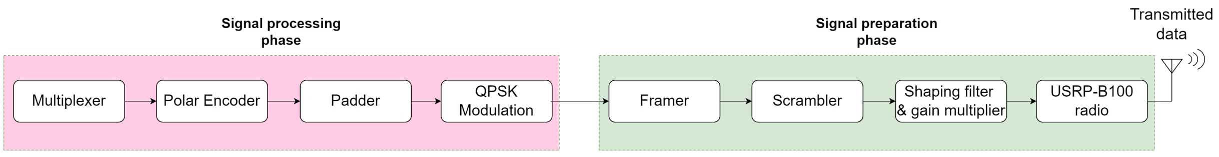


Figure 2: Block representation of the transmitter

3.5.1 Sources

The *Source_user_binary* block reads the data from the input file and returns a sequence of bits. The first *Source_random_fast* is utilized to generate the ineffective bits that will be sent to the multiplexer. Meanwhile, the second block is used to balance the constellation of symbols, as padding adds the same pattern of bits repeatedly (e.g. `010101`...), resulting in a predominance of the symbol associated with `01`.

3.5.2 Framer

The framer structures the data frames by appending headers and trailers at the beginning and end of each frame, facilitating proper processing of the received data by the receiver.

3.5.3 Scrambler

A scrambler is employed to scramble the transmitted data. It alters the data in a random manner prior to transmission, thereby thwarting interference, minimizing transmission errors and fortifying the data against surveillance or unauthorized interception.

3.5.4 Shaping Filter

The shaping filter is designed to match the data signal to the available bandwidth. It restricts the range of frequencies employed for data transmission to mitigate interference with other transmissions. It also optimizes the transmission power for a given bandwidth, making it a vital component in ensuring optimal transmission quality.

3.5.5 Gain Multiplier

The gain multiplier enables the amplification of the signal to reach the necessary transmission power to cover the required distance. It is used in conjunction with the shaping filter and the modulator to tailor the data signal to the available bandwidth and the shape of the carrier wave utilized to transmit data.

3.6 Receiver

The receiver first synchronizes in both time and frequency to ensure proper processing of the received data by performing the inverse operations of those carried out during transmission.

Figure 3 shows the receiver's communication chain. (Figure 12 is more detailed)
To align with the overall design described in the figure 7, additional processing blocks have been implemented before the demodulation step, since the processed signals come from radio sources. Indeed, during transmission, the transmitted signal is subject to various disturbances such as interference from other sources, noise, or degradation caused by the transmission medium. These perturbations can affect the quality and accuracy of the received signal.

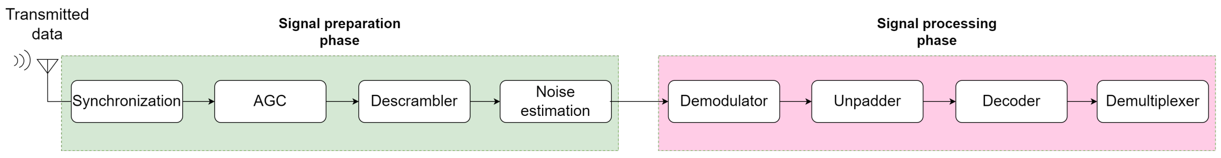


Figure 3: Block representation of the receiver

3.6.1 Synchronization

Synchronization refers to aligning the timing and frequency of the transmitter and receiver. Timing synchronization ensures that the receiver knows when a bit starts and ends, so that it can correctly interpret the data being transmitted. Frequency synchronization ensures that the receiver knows at which frequency the transmitter is operating, so that it can correctly interpret the signal being transmitted.

3.6.2 AGC

The purpose of the automatic gain control (AGC) is to keep the output signal level constant despite variation in the input signal level. AGC is often used to ensure that the output signal remains at a constant level, despite variations in the input signal level.

3.6.3 Descrambler

A descrambler is used to reverse the scrambling process and recover the original signal.

3.6.4 Noise Estimation

Noise estimation is the process of determining the level of noise present in a signal. The goal of noise estimation is to determine the amount of noise present in a signal so that it can be removed or reduced.

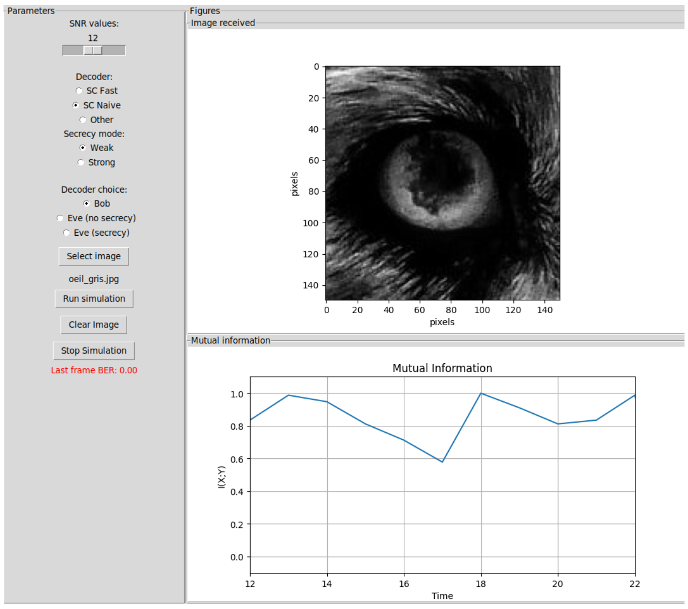
3.6.5 Sink

Once all the signal processing stages are finished, the receiver sinks the outcome of the process to a file on a storage device, preserving it for later access and utilization, instead of losing it after the process ends.

3.7 Interfaces

One graphical user interface (GUI) was created using PYTHON'S TKINTER library to enable users to run simulations and experiments without modifying the code themselves. This interface showed in Figure 4 allow the user to run the entire project by simply clicking buttons.

- The left side of the interface allows users to easily set various parameters, including the encoder and decoder options, the signal-to-noise ratio, and various secrecy options. Additionally, the interface includes buttons for starting, stopping, and controlling the simulation. Users can select the source image to transmit and then run the simulation at their convenience.
- On the right side of the interface, the user can monitor the simulation progress by viewing various metrics such as mutual information. The decoded image is also displayed in real-time. When the receiver is Bob, the original image is reconstructed and can be viewed clearly. However, for Eve, the image will be mostly noise and it will not be intelligible.


Parameters

SNR values: 12

Decoder: SC Fast [✓ SC Naive] Other

Secrecy mode: [✓ Weak] Strong

Decoder choice: [✓ Bob] Eve (no secrecy) Eve (secrecy)

oeil_gris.jpg

Last frame BER: 0.00

Figure 4: Interface for simulations

3.8 Continuous Integration

Continuous integration is necessary for any project with different development teams. It involves practices and tools that facilitate the integration of code changes. Continuous integration allows to detect and resolve conflicts between different version of the project so the development team can work more efficiently.

- Github is the key tool to maintain the stability of the project, regardless of the new features added. Those features are added to different branches within the repository, allowing each team member to work on their part without disrupting the main codebase. Once the bugs are fixed, testing functions are implemented, and results were conclusive, the developed features (eg: the created modules) are then merged into the main branch of the project. Furthermore, Github also provides the ability to revert a corrupt repository to a previous version in case any errors stepped in.
- Testing functions are implemented to guarantee the operability of the different functions implemented in each module. Since our code is in PYTHON, PYTEST is used to check the whole set of test functions within the repository. Once every test is passed, the added features can be pushed to Github.
- Each created module is written with its documentation, giving details about functions composing this module and the inputs and outputs of each function. There is also a sample code with fake data to give an example to an external user on how to implement this module with AFF3CT.

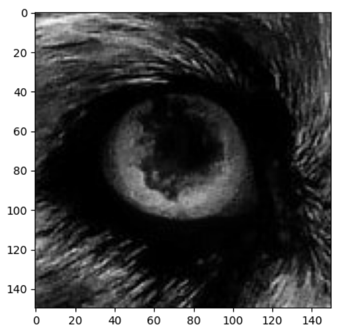
4 Results

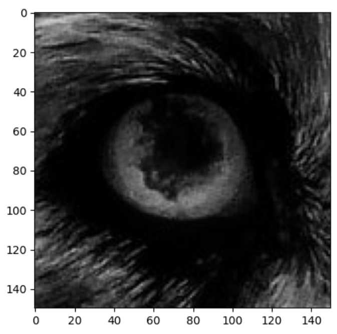
4.1 Image Transmission

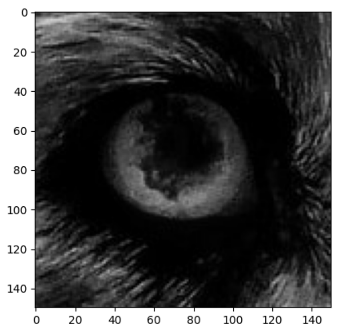
Figure 5 demonstrates the results achieved when an image is transferred through the wiretap channel.

The image sent by Alice can be seen in Figure 5a and the image Bob receives is shown in Figure 5b. Since their communication chain is setup to minimize the amount of errors on their channel, it is not a surprise that Bob receives the image correctly.

When Eve uses the frozen bits optimized for her wiretap channel, the image received is the one seen in Figure 5d, whereas if she uses Bob's frozen bits and the degradation of her channel are not too heavy, she receives the image in Figure 5c. This depicts the limits of weak secrecy, since Eve only has to know Bob's decoder's configuration to break the secrecy.


(a) Alice


(b) Bob


(c) Eve with Bob's frozen bits

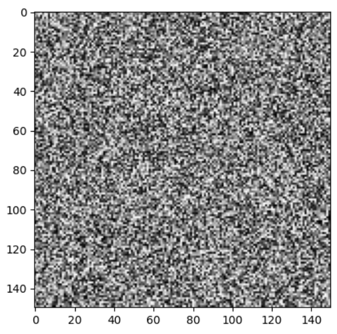

(d) Eve with her own frozen bits

Figure 5: Transmission over weak secrecy

5 Organization and Management

5.1 Expected Objectives

The goal of this project was to complete four objectives. The first objective was to conduct bibliographic research on polar codes and the wiretap channel, using sources such as the thesis [4] and Mr. Tajan's course [5]. A shared document was created to summarize the findings of the research.
The second objective was to simulate the transmission and reception of data from Figure 1 on a single computer to assess the feasibility of the project. An existing code was adapted, and additional modules were created for this purpose.
The third objective was to create a real-time demonstrator using software-defined radios to illustrate the experience. The demonstrator consisted of two USRP-B100 radios transmitting data, with the remainder of the system being software-based.
The fourth and final objective was to conduct the experience using the proposed demonstrator. During the project, additional features such as documentation for the created modules, an interface to aid in understanding the project, and test functions to verify the functionality of specific sections of code were also implemented.

5.2 Organization

At the beginning of the project, each member of the group read a portion of the thesis [4] and summarized it in a shared document to understand their individual tasks without having to read the entire document. The team then split into two groups, with Rémi, Elias and Alexandre working on simulations on their computers and Yahya and Guillaume using radios to perform real communications. Once both groups had completed their respective tasks, the python scripts were merged, and the team returned to working as a single unit. However, each team member continued to work on specific tasks until the end of the project. Figure 13 shows a table detailing the task assignments, with blue representing the sprints while working as a single group and green representing the sprints spent in the two smaller groups.

5.3 Management Methods

As said previously, the group worked in sprints. This way of operating comes from agile management and especially here, the Scrum framework, which includes the roles of product owner and Scrum Master. Elias fulfilled both roles, with the product owner defining and maintaining the product backlog, which is the prioritized list of features that the development team will work on. The product owner also tracks progress using a burn up chart (Figure 14). This chart is a graphical representation of work completed and work remaining in a project.
It is a high-level view of the team's progress, and it is simple to understand how much work is left to be done in the project. The scrum master facilitates the daily stand-up meetings, the sprint planning meetings, the sprint review and retrospective meetings. The Scrum Master also helps the team to understand and follow the principles and practices of Scrum and works to remove any obstacles that may be blocking the team's progress. The rest of the team, referred to as developers, were responsible for completing all necessary work to build valuable increments of the project in each sprint. The team was self-organizing, meaning that they had the autonomy to make decisions and determine the best approaches to their work.
Overall, the results of the project show that the missions were completed on time, thanks in part to an effective assignment and distribution of tasks.

6 Conclusion

To put everything in a nutshell, this project's objective is to use some particular properties of polar encoders to ensure secrecy between the transmitter and the receiver.

The principle of polar codes is to transform one channel, with a certain transition probability into a lot of smaller channels, some which are degraded, and others which are improved. Thanks to this knowledge, two levels of secrecy can be achieved. One the one hand, weak secrecy consists on communicating on channels that are both good for the rightful receiver, and bad for an eavesdropper. It is easy to implement and works well as long as the eavesdropper does not know which channels should be activated among its bad channels, but this sole knowledge breaks the weak secrecy. On the other hand, strong secrecy relies on a heavier filtering of the potential channels, in order to find those on which an eavesdropper can almost never read anything, due to the degradation. It is more complicated to implement and reduces drastically the communication rate, but it is unbreakable by construction.

Two demonstrators were built to show this idea. One uses two radios to simulate the exchange between Alice, Bob and Eve. The other emulates the communication. The objective of these demonstrators is to illustrate the fact that with secrecy Eve can no longer decode the image, as well as to demonstrate the limits of the secrecy. The mutual information between the transmitter and the receiver is used as a metric to illustrate these ideas. The demonstrators used in this project could have been improved with more time. For instance, with more options available for the parameters, or by combining different metrics instead of relying only on the mutual information.

References

[1] Hessam Mahdavifar and Alexander Vardy. “Achieving the Secrecy Capacity of Wiretap Channels Using Polar Codes”. In: *IEEE Transactions on Information Theory* 57.10 (2011), pp. 6428–6443. DOI: 10.1109/TIT.2011.2162275.

[2] A. D. Wyner. “The wire-tap channel”. In: *The Bell System Technical Journal* 54.8 (1975), pp. 1355–1387. DOI: 10.1002/j.1538-7305.1975.tb02040.x.

[3] A. Cassagne et al. “AFF3CT: A Fast Forward Error Correction Toolbox!” In: *Elsevier SoftwareX* 10 (Oct. 2019), p. 100345. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2019.100345. URL: http://www.sciencedirect.com/science/article/pii/S2352711019300457.

[4] Khaled Taleb. “Physical layer security : Wiretap polar codes for secure communications”. PhD thesis. Université de Toulouse, 2022. URL: https://depozit.isae.fr/theses/2022/2022_Taleb_Khaled.pdf.

[5] Romain Tajan. *Codage pour la 5G*. Last accessed 10 January 2023. 2022. URL: https://rtajan.github.io/assets/cours/TS345/slides/TS345_POLAIRES.pdf.

A Extra figures

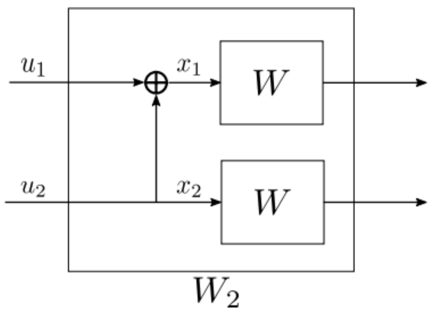


Figure 6: Combination of channel

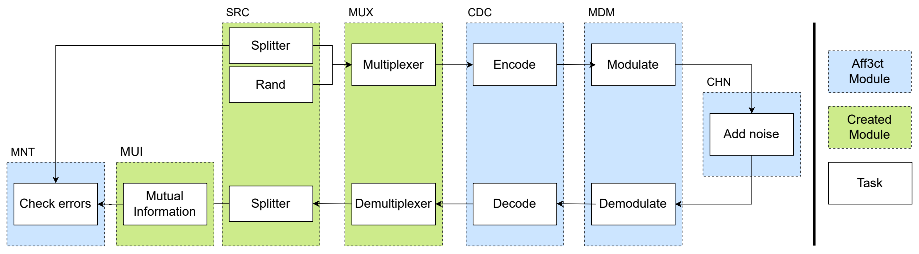


Figure 7: Block representation of the situation

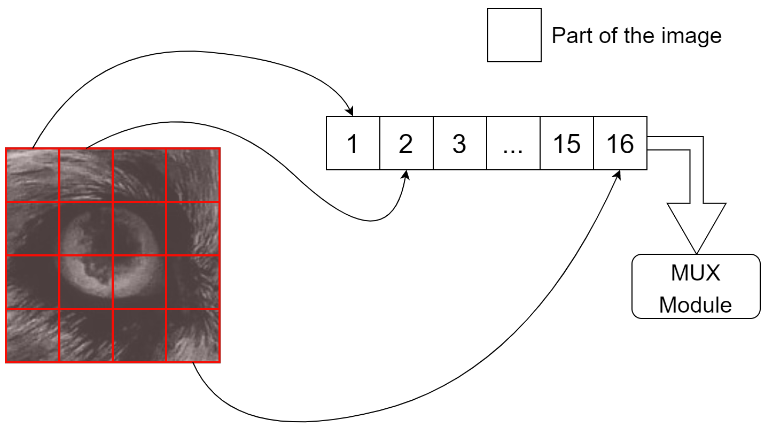


Figure 8: SOURCE module

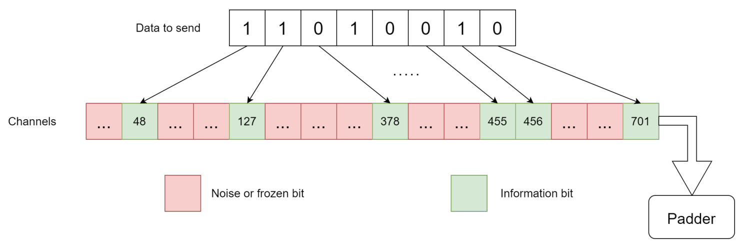


Figure 9: MUX module

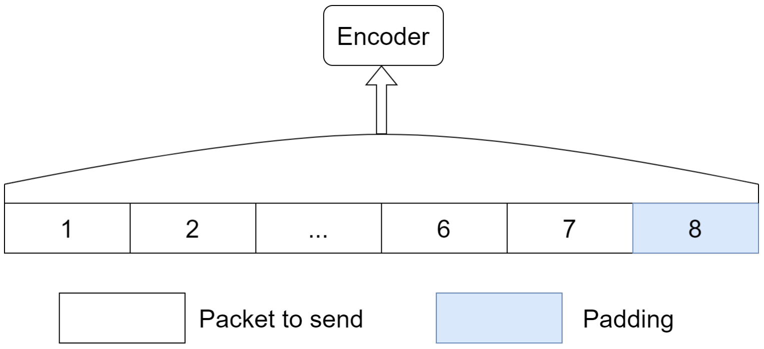


Figure 10: PADDING module

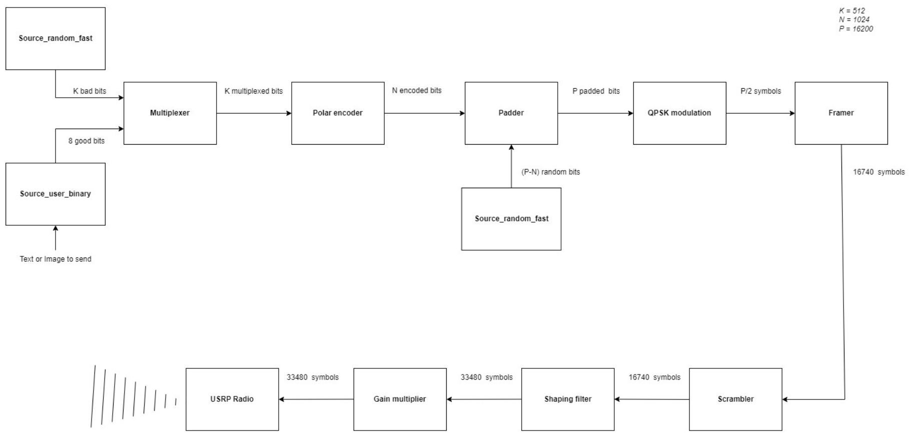


Figure 11: Block representation of the transmitter

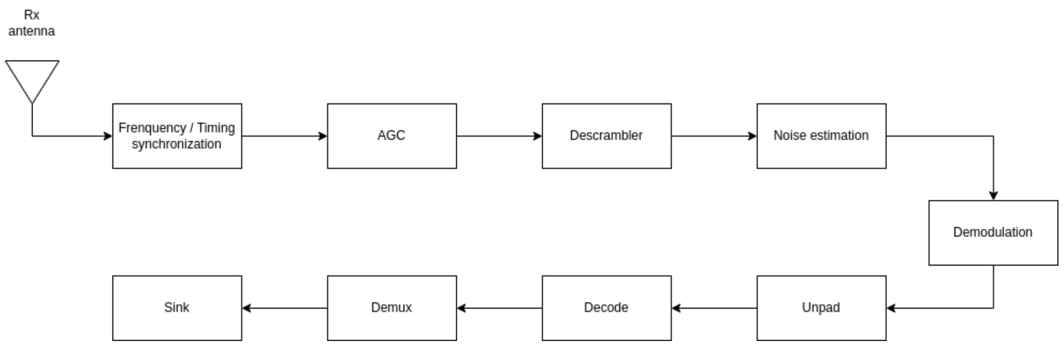
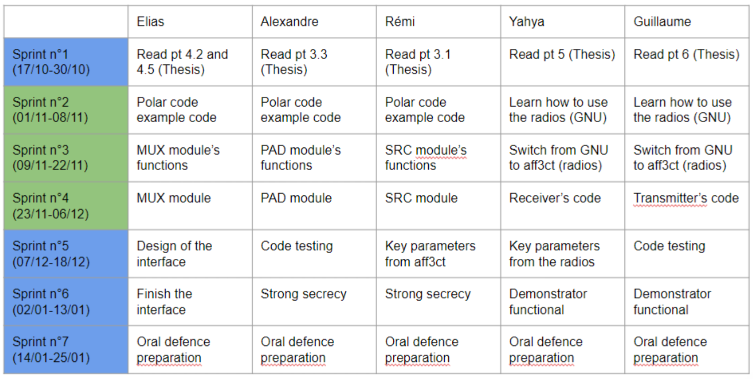


Figure 12: Block representation of the receiver



	Elias	Alexandre	Rémi	Yahya	Guillaume
Sprint n°1 (17/10-30/10)	Read pt 4.2 and 4.5 (Thesis)	Read pt 3.3 (Thesis)	Read pt 3.1 (Thesis)	Read pt 5 (Thesis)	Read pt 6 (Thesis)
Sprint n°2 (01/11-08/11)	Polar code example code	Polar code example code	Polar code example code	Learn how to use the radios (GNU)	Learn how to use the radios (GNU)
Sprint n°3 (09/11-22/11)	MUX module's functions	PAD module's functions	SRC module's functions	Switch from GNU to aff3ct (radios)	Switch from GNU to aff3ct (radios)
Sprint n°4 (23/11-06/12)	MUX module	PAD module	SRC module	Receiver's code	Transmitter's code
Sprint n°5 (07/12-18/12)	Design of the interface	Code testing	Key parameters from aff3ct	Key parameters from the radios	Code testing
Sprint n°6 (02/01-13/01)	Finish the interface	Strong secrecy	Strong secrecy	Demonstrator functional	Demonstrator functional
Sprint n°7 (14/01-25/01)	Oral defence preparation	Oral defence preparation	Oral defence preparation	Oral defence preparation	Oral defence preparation

Figure 13: Task repartition table

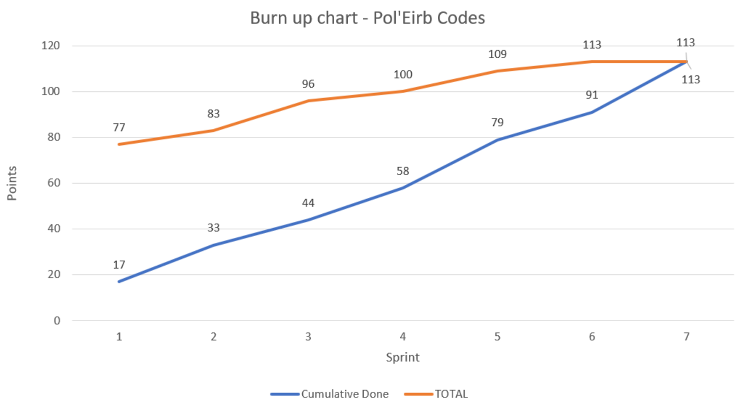


Figure 14: Burn up chart of the project

B Equations

Mutual information

Let X be the bit sent by Alice, and Y its counterpart received by either Bob or Eve. Then, the mutual information between X and Y writes

$$\mathbb{I}(X;Y) = \sum_{x=0}^{1}\sum_{y=0}^{1} \log_2\left(\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}\right) p_{X,Y}(x,y) \tag{1}$$

where X denotes the transmitted bit, Y its received counterpart, and $p_X(x)$ denoted the probability of the event $X = x$.