Implementation of a Low-cost AIS Transmitter based on SDR

Romina García
Universidad de la República
Montevideo, Uruguay
rominag@fing.edu.uy

Máximo Pirri
Universidad de la República
Montevideo, Uruguay
mpirri@fing.edu.uy

Gonzalo Belcredi
Universidad de la República
Montevideo, Uruguay
gbelcredi@fing.edu.uy

Claudina Rattaro
Universidad de la República
Montevideo, Uruguay
crattaro@fing.edu.uy

ABSTRACT

The Automatic Identification System (AIS) was developed primarily to enhance maritime safety. While large ships are mandated to equip AIS stations, the situation differs for smaller vessels, particularly recreational ones, as AIS technology is not obligatory. Additionally, the cost of AIS stations poses a significant barrier for owners of fishing vessels or boats. This study outlines the creation of a low-cost AIS transceiver station using Software Defined Radio (SDR) technology. The prototype has been meticulously designed, implemented, and evaluated to showcase its viability. As a tangible outcome, an out-of-tree GNU Radio module has been developed, with its code readily accessible online.

CCS CONCEPTS

• **Hardware** → *Analog, mixed-signal and radio frequency test*; • **Applied computing** → **Engineering**; **Telecommunications**.

KEYWORDS

Automatic Identification System, Software Defined Radio, GNU Radio, Transceiver

1 INTRODUCTION

Communication has been a key factor in human life ever since its beginning. Most of the technological devices developed in the last centuries have had the purpose of facilitating connections between people [18]. Inventions such as the telegraph or the radio succeeded in satisfying this need. Maritime communications, aside from allowing for the exchange of information, seek to provide improved safety conditions to individuals at sea.

Radars are one of the first forms of technologies which looks to provide maritime safety [14]. They are designed to send radio frequency (RF) signals in all directions and detect their bounces. This allows for a mapping of the surroundings of the ships to be created. However, radars don't provide information about which objects have been identified. Their functionalities are not trustworthy when there is no line of sight with potential obstacles. There was a significant milestone in maritime safety in the beginning of the twenty-first century, with the introduction of the Automatic Identification System (AIS).

AIS is a technology whose main objective is to improve the safety of vessels at sea. It was defined by the International Telecommunications Union in their ITU-R M.1371-5 recommendation [24]. Periodic messages are broadcasted in the Very High Frequency (VHF) band by the stations. A considerable amount of information is encapsulated into the different existing AIS messages. The most commonly sent information is related to the ship's position, speed and course, but static data (such as identification numbers or ship dimensions) is also shared.

Many advantages of the AIS technology in comparison to radars can be identified. The use of RF channels for the exchange of information allows for better performance. There is no longer a need for line of sight to know about the presence of other vessels. AIS also collects a lot of updated information about maritime traffic [1]. In contrast, radars operate at higher frequency bands like 3, 5, or 9 GHz [23], resulting in diminished performance during adverse weather conditions.

The ITU-R M.1371-5 recommendation defines a wide variety of AIS systems. The most common are the mobile AIS stations, which are mounted on ships and allow for their communication with other stations at sea or at shore. There are two types of mobile stations: class A stations and class B stations. Depending on the size or the purpose of a certain ship, one or the other class should be used. Class A AIS stations are commonly used by large ships. Meanwhile, class B stations are recommended for smaller, usually recreational vessels. These systems are designed to allow for inter-operation. Their differences are mainly based on transmission power requirements or message sending intervals.

This work focuses on class B AIS stations. There are two subtypes to be considered, according to the medium access mechanism used. The two frequency channels must be shared between all nearby stations. A time division based multiple access (TDMA) is

used, considering the Universal Time Coordinated (UTC) reference. Minutes are called *frames* and divided into 2,250 slots of about 27 ms each. Only one station is allowed to send information in a channel during said slot. The two medium access schemes used by AIS are self-organized TDMA (SOTDMA) and carrier-sense TDMA (CSTDMA). The first allows stations to make reservations on certain slots for transmissions in advance. All stations will then have a mapping on who will transmit and when. The second is based on measuring the noise level in the channel in the beginning of a selected slot. If, according to a dynamic threshold, the station determines the channel is not occupied, it will transmit. Otherwise it will wait until the following selected slot.

The AIS technology was first introduced in the 1990s. Only class A stations were specified, meant to be used by cargo or passenger ships of considerable size. It wasn't until it became mandatory worldwide to have an AIS station in 2008 that its use became widespread among large ships [14]. In 2006, class B AIS stations were defined [19]. However, their use is generally not mandatory.

The situation in Uruguay is similar to that of most countries. Class A AIS stations must be mounted on large ships, but it is optional for smaller vessels to have a class B AIS station [9, 25]. These regulations, together with the considerably high prices of stations, cause most people to opt out from using AIS. This incredibly compromises the safety of people at sea, taking into consideration the technologies available nowadays.

This article details the development of a cost-effective Class B AIS station, delineating the design, implementation, and validation phases. Leveraging Software Defined Radio (SDR) technology for signal processing and transmission, the station employs the CSTDMA medium access mechanism. A dedicated GNU Radio module, named *gr-itais*, was created for this purpose, with its source code accessible on our repository [12].

The remaining sections are structured as follows. Section 2 presents previous work on the problem as well as open source tools used in this project. Section 3 describes the software development done. Specifically, the decision and implementation process of the medium access scheme and the generation of messages are detailed. The hardware prototype used for testing is shown in Section 4, and tests and results are presented in Section 5. Finally, Section 6 includes a discussion of the work done and presents lines of future work.

2 PREVIOUS WORK

Significant research and development efforts have been dedicated to maritime communications, with particular emphasis on the Automatic Identification System (AIS). Since its inception, the AIS system has been the subject of extensive study and analysis within the maritime community.

The performance of the medium access mechanisms used by the technology has been evaluated in various articles. Specifically, the comparison between class A stations (using SOTDMA techniques) and class B stations (mainly considering the ones that are CSTDMA based) is studied. In [14], the authors test whether the VHF channel presents signs of congestion when AIS stations of different classes join the shared medium. A simulator is developed to track the movement of ships and usage of the channel. The results show that, although class B stations using a carrier sense medium access strategy are the most affected by congestion, the effects are not significant. However, it is interesting to consider how class A stations modify channel dynamics. In [19], the effects of the addition of class B stations on the previously operating class A stations are studied. Although the authors conclude that the impact is noticeable, it is mentioned that adding a class A station has a much bigger effect on the channel in comparison to the addition of a class B station. Works like [15] and [7] evaluate changes in medium access mechanisms, such as the use of satellites to improve surveillance capacities.

The belief that AIS stations provide for better safety conditions at sea is universal. Multiple works describe the development of simplified or low cost stations [8, 11, 13, 17]. Most of these precedents are based on dedicated hardware (not general purpose hardware like SDR technology), lack available code, or have only a partial implementation of the AIS transmitter function.

On the flip side, a significant volume of open source research has been conducted on the implementation of AIS stations. Various projects focus on either the transmission or reception aspects of AIS. Common examples include receivers like [2, 4, 16], typically developed in C or C++. Additionally, some endeavors present fully functional devices [3]. Among the most valuable resources that served as the basis for our prototype are two GNU Radio modules[1] outside the tree: *gr-aistx* [22] and *gr-ais* [5]. Building upon GNU Radio modules is straightforward due to their block-based development approach. The software seamlessly integrates with SDR devices.

2.1 The *gr-aistx* Module

This is an open source implementation of an on-demand transmitter. Considering a string of bits that represents the payload of a message, a block is created to build the AIS frame and encode it using non-return to zero inverted (NRZI). The module includes an example GNU Radio flowgraph that also contains blocks for Gaussian Minimum Shift-Keying (GMSK) modulation, as well as channel selection and transmission.

The developed transmitter does not implement any intelligence. That is, there are no timers to measure when messages should be sent or blocks to consider medium access. The main objective of the project was to contribute to the community with means to run simple simulations. The authors also provide users with a message generation script. The code was written for GNU Radio 3.7, with blocks in C and C++, but was later adapted to a GNU Radio 3.8 version [6].

2.2 The *gr-ais* Module

This module includes a simple receiver developed for GNU Radio 3.8. Some blocks are defined in order to detect the preamble for AIS messages, perform synchronization and then retrieve the original payload. The received signal is GMSK-demodulated and NRZI-decoded. The data field of the message is then expressed as a NMEA sentence and printed to terminal.

[1] GNU Radio modules are small projects consisting of a few blocks developed by members of the community. These are usually dedicated to a specific technology (such as AIS, WiFi, Bluetooth). The blocks can be written either in C/C++ or Python, and both can coexist within the same project.

The development of the blocks did not initially include a graphic representation. For this reason, the main processing of samples is done using a Python script that encodes a GNU Radio flowgraph. This scheme is then replicated in our *gr-itais* module, whose functionalities are also defined on a script that interfaces Python and GNU Radio.

3 SOFTWARE DESIGN AND DEVELOPMENT

To comply with the ITU-R M.1371-5 recommendation, an algorithm was needed to be able to access the channel in an organized manner. Once the channel can be accessed, a way of creating the messages is needed. Which messages must be sent depend on the type of the selected class B equipment. In this section the decision for which medium access mechanism to implement is presented, as well as its development. In addition, we detail a description for needed messages and the process for their creation. The code within this project was mainly written using the Python programming language, using GNU Radio 3.8.

3.1 Medium Access Mechanism Development

When working with AIS technology, a wireless physical medium for transmissions is used. A channel free of collisions is ideal, in order to allow all of the messages to be decoded correctly. In the ITU-R M.1371-5 recommendation, two different ways of accessing the shared medium are described for class B equipment. Both of them are based on TDMA strategies. This implies that stations take turns to send messages and avoid collisions. In each time slot, only one station is allowed to transmit and transmissions should not exceed slot duration. All stations take UTC as a reference to determine the beginning of frames.

Class B equipment can use either SOTDMA (also called SO) or CSTDMA (also called CS) for medium access. The idea behind SOTDMA is that different AIS stations can make reservations for potential transmission slots. Other nearby SO equipment will receive said reservations and avoid those slots for their own transmissions. On the other hand, CSTDMA is based on choosing candidate slots. The equipment will sense the channel for a short period of time in the beginning of each slot, as it is shown in Figure 1. The sensing starts around 800 μs after the time slot has begun, in order to allow some time for other stations to turn on their RF power. According to the comparison between the channel noise level and a dynamic threshold, the medium can be classified as free or occupied. If the channel is free transmission is carried out. Otherwise, the station retries in the following candidate slot avoiding collisions.

Based on the previous description, it is clear that CS is a simpler mechanism. Class B CS stations are required to send fewer messages and meet looser specifications. This, together with the fact that less transmission power is necessary (which translates to reduced costs), caused an inclination towards the implementation of the CS mechanism. However, SOTDMA has a certain priority over CSTDMA since it can book slots. This could potentially lead to constant failed transmissions for CS stations. In order to determine whether this is a disadvantage or not, simulations were carried out. These preliminary simulation results were presented in detail at the XXX Conference of Young Researchers AUGM-UNA [20]. A

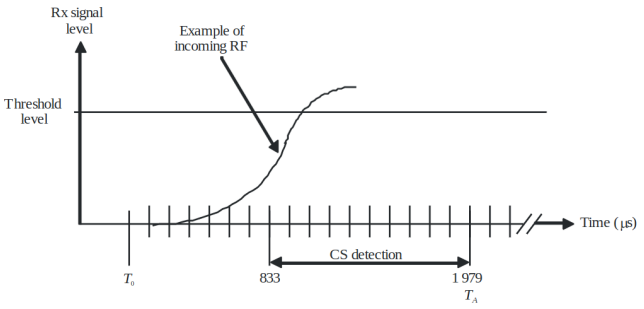

Figure 1: Sensing of Channel Occupation during the Beginning of a Time Slot. This Image is Taken from the ITU-R M.1371-5 Recommendation, where it is Referred to as Figure 35 [24]

simple tool for simulation was developed in Python, included in the repository of the project [12].

To summarize the simulation results, we have examined various scenarios, utilizing the shortest transmission period for each type of station, thereby creating the most congested channel scenario. The amount of SO stations was increased up to the point were the channel congestion was too high. We found that the upper limit of this kind of stations approaches 160 before the rate of lost transmissions becomes significant (around 30%). In Montevideo, Uruguay (one of the major ports of South America), it is estimated that approximately 100 SO stations would represent the worst-case scenario for the AIS channel.

Complementary, in Figure 2 the amount of successful transmissions is shown as a function of the attempted transmissions. For each curve, the number of total stations increase from 100 to 400, considering the percentage of CS indicated in the labels of the graph. The remaining stations were of class SO. This intended to check whether adding more SO stations in cases of congestion has a negative impact on the channel. Given that SO stations have a smaller reporting interval in comparison to CS stations, Figure 2 could be misleading. This is caused by the fact that the more SO stations, the higher the amount of successful transmissions. However, the graph succeeds in showing that if the proportion of SO stations is higher, the amount of successful transmissions is limited. On the other hand, with the same amount of stations but a greater proportion of CS, the channel availability will allow its use to more stations. The simulations show that CS stations should not suffer from significant losses due to SO stations, if certain amounts are not exceeded. In particular, the situation in Uruguay would allow a successful use of CS stations. These are better at avoiding congestion in a channel and adapting to its conditions. All of these considerations resulted in CSTDMA to be the medium access mechanism chosen for the implementation of the transceiver.

CS stations must be able to listen to the channel to guarantee correct operation. Samples of the channel must be received and filtered, as shown in Figure 3. In this development said samples enter a block called `Potumbral` that was created to compute the

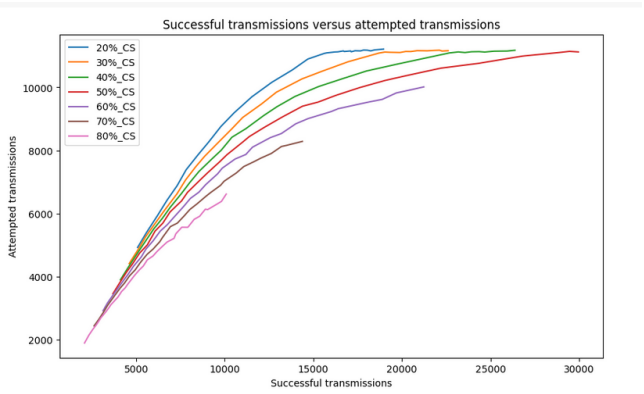

Figure 2: Graph of the Amount of Successful Transmissions as a Function of the Attempted Transmissions for Different Proportions of CS Equipment

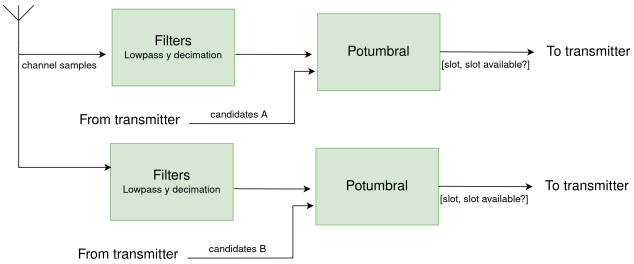

Figure 3: Reception of Samples and Connection with `Potumbral` Block

dynamic threshold[2]. The block also takes into account the noise level of the channel.

In order to compute the threshold dynamically, the samples corresponding to the last minute of the channel are considered. A total of 15 intervals are defined, each representing 4 out of the last 60 seconds of the channel. The value of the intervals is estimated with an average of the noise level in said time period. The smallest value of all 15 will be used as the current threshold of the channel.

The `Potumbral` block receives candidate slots from another block called `Transmitter`. The `Transmitter` block is described in detail in Section 3.3. If the current samples in `Potumbral` lay within the beginning of a candidate slot, channel power is compared to the threshold. If the power is smaller than the current threshold the station should send the message. Otherwise, it will try again in the following candidate slot.

The output of the `Potumbral` block consists of a pair of numbers (n, m). Here, n is the number of the slot that was sensed and m is either 0 or 1. If m is 0, it means that `Potumbral` detected that the power of the channel was greater than the threshold for the slot n and therefore it is occupied. The slot can not be used for transmission. On the other hand, if m is 1 it means that the channel is free.

[2]An independent threshold is considered for each frequency channel. To do so, two instances of filters and of `Potumbral` are used.

Figure 4: Frame of an AIS Message, Composed of Six Main Fields that Add up to 256 Bits. This Figure was Adapted from Figure 6 of the ITU-R M.1371-5 Recommendation [24]

This array is sent to `Transmitter` who will send the instruction to transmit the message (if the slot is available).

3.2 Generation of AIS Messages

Each type of station has a different set of messages it should transmit. Some of them are optional and some of them are mandatory. In this first prototype optional messages will not be considered.

As argued in Section 3.1, a class B CS transceiver will be developed. For this type of stations, only messages 18 and 24 are required. Message 18 consists of dynamic information like latitude, longitude, speed over ground and course. Its reporting interval varies depending on the speed of the ship. Message 24 corresponds to the report of static information. It is divided in two parts, message 24 A and message 24 B. The part A of the message is transmitted every 6 minutes and it is used to associate the identifier of the ship (called MMSI) with its name. Part B of the message is transmitted 1 minute after the transmission of part A and it is used to send the dimensions of the ship and other identifiers.

For the current prototype, a block called `Messages` was created. This block has two inputs. On the first input, information from a GPS module is received[3], which is then used to generate message 18. The second input is connected to the `Transmitter` block, receiving information about which message should be generated.

Some blocks from the *gr-aistx* project [22] are used for the generation of messages. There is also a Python function available in said module that builds strings of bits that represent messages. According to the information received from `Transmitter`, the corresponding function is invoked. The resulting string of bits is the output of the `Messages` block, which is then sent to the `AIS Frame Builder` block. This block was obtained from the *gr-aistx* project and conforms the frame according to the ITU-R M.1371-5 recommendation. Once the frame has been built as shown in Figure 4, the bits are encoded using NRZI. The resulting stream is then fed to a GNU Radio block that uses GMSK modulation to generate a ready-to-send signal.

3.3 Transceiver System

In order to transmit messages complying with the reporting intervals of class B CS equipment, the blocks involved in the project need to exchange information. In Section 3.1 some aspects of the interconnection were mentioned. In particular, it was stated that the receiving part of the prototype (which consists mainly of the `Potumbral` block) needs to send and receive information from the `Transmitter` block.

The `Transmitter` block generates the candidate slots in which the equipment will try to transmit. Based on the reporting interval, a central slot for the next transmission is selected. Then, an interval

[3]The data received from the GPS module, aside from being used to properly locate the ship, is used to receive the UTC reference. This is then injected in the system's clock.

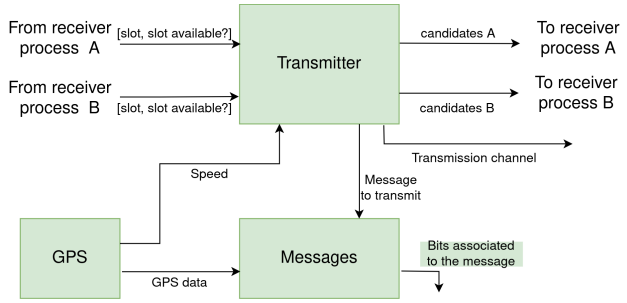

Figure 5: `Transmitter` Block Connection to `Messages` and Output of Bits

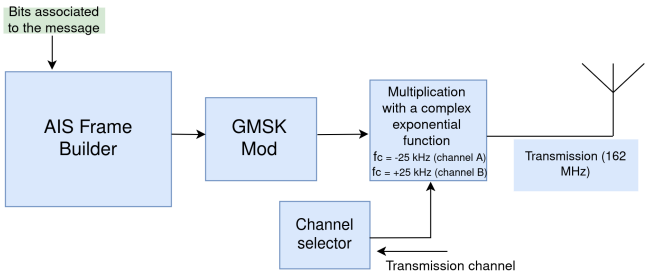

Figure 6: Final Part of the Transmission. The String of Bits of the Message are Built into a Frame (and Coded with NRZI), Modulated in GMSK and Sent through the Channel Indicated by the Selector

is defined around this particular slot, and ten candidate slots are randomly selected. Once the candidates are selected (a few seconds before the scheduled transmission) they are sent to `Potumbral`. This block will then determine whether a transmission can be done or not in each slot. If one of the candidates can be used for a transmission, the `Transmitter` block will inform `Messages` which message to generate. After transmission, a new center slot will be fixed according to the reporting interval. These connections are shown in Figure 5.

Once the system knows when to send a message, and what message to send, transmission can be done effectively. The information follows the path described in Section 3.2 (and shown in Figure 6). The two frequency channels used by AIS are 161.975 MHz and 162.025 MHz. According to the corresponding channel for transmission[4], the signal (originally centered in 162 MHz) is shifted ±25 kHz.

In order to receive messages from other stations, the *gr-ais* module can be incorporated into the transmitter unit. Messages from other stations will be printed to terminal, as well as the station's own messages. This is because in this first prototype message interpretation was not implemented.

[4]Transmission should alternate between both channels. The `Transmitter` block determines the correct channel and configures a channel selecting block.

4 HARDWARE PROTOTYPE

The development of this project focused mainly on software. However, a simple hardware prototype was designed to work with and run tests. The parts were chosen to try and minimize the costs involved, all the while allowing for proper operation.

The code was run in two different computers, one considerably more expensive than the other. The most expensive was an Intel NUC with an i3 processor, including a 16 GB RAM memory and a 200 GB SSD disk. The operating system used was Ubuntu 20.04. A Raspberry Pi 4 Model B was also used, with a 32 GB SD card and 8 GB of RAM. It was running a Raspberry OS version from September 2022. Both of them allowed for good test results, but the Intel NUC was slightly better, as it is shown in Section 5.

SDR technology was used for the project, although the costs of these devices are usually significant. There are several advantages to back up this decision. First, it is not necessary to design specific hardware, which is a long and hard process. Using SDR devices it is possible to create prototypes fast and take into consideration changes in standards or design goals. They allow for processing every sample taken from the RF channel, which was very important for the implementation of the carrier sense mechanism. It is also worth noting that there is a big community online of professionals and amateurs contributing to SDR based open source projects. This helped the development of our project.

An ADALM-PLUTO SDR device was used for sampling the RF channels for noise level measurement and transmission. The carrier sense strategy required the SDR to be calibrated, given that minimum or maximum power levels had to be configured in relation to what is stated in the ITU-R M.1371-5 recommendation. This process involved a spectrum analyzer to compare its received power with the ADALM-PLUTO device.

An extra SDR device was used for receiving AIS messages. Specifically, an RTL-SDR dongle that is only suitable for signal reception.

GPS information was needed to generate messages with accurate data. A NEO-7M GPS module was purchased, along with an active GPS antenna (for improved GPS signal reception). The module was connected to the Intel NUC using USB. The inter-connection to the Raspberry Pi was done using wires, using the General Purpose Input/Output (GPIO) pins included in the computer. Aside from access to GPS data, the module allowed for synchronization with the UTC reference. Its pulse per second (PPS) output was used to configure a Network Time Protocol (NTP) server [10].

5 PERFORMANCE EVALUATION

An important part of the design of a prototype is the evaluation of its performance. Aside from verifying that the system meets requirements, we evaluate to what extent the ITU-R M.1371-5 recommendation is complied. Several tests were carried out in both computers in which the prototype was developed. The results are shown in Table 1 and are discussed in detail in the following subsections

5.1 Test 1: Transmission Power Evaluation

A relevant parameter to measure is the transmission power. In order to do so, a spectrum analyzer was connected to the transmitting SDR device. Five messages were transmitted in each channel and

Table 1: Results Obtained from the Tests Carried out

Test	Results (NUC and RasPi)	Observations
1	- 1.5 dBm	An amplifier of 31 dB is needed
2	Threshold Update ✓✓	The Rapberry Pi is not fast enough to meet the required value
	Switching time ✓	
3	97%, 75%	All slots should be seen by the code to ensure the reporting intervals are correct
4	Seen in the spectrum: 90%+ Seen by the receiver: ~ 40%	Potential sources of error are the AIS frame Builder block and GNU Radio buffering
5	Accuracy of 100 %, 80%	
6	Delay of 127ms	

the maximum power was obtained by computing the average of the ten transmissions. The values obtained were close to -1.5 dBm, which does not comply with the ITU-R M.1371-5 recommendation. The minimum transmission power is established to be 1 W in class B CS equipment. This value could be reached without significantly increasing the costs of the prototype by using an amplifier of 31 dB.

5.2 Test 2: Threshold Update Time and Channel Switching Times Evaluation

It was tested if the time needed to update the dynamic threshold is enough for the system to work as established in the ITU-R M.1371-5 recommendation. Because of the way it is computed, the threshold will not have real channel values for the first minute since the system was turned on. During this minute, class B CS equipment should sense the channel and update its threshold without transmitting. Transmission will begin only after this time has passed. With the NUC computer this restriction is correctly met, even if *gr-ais* is running for message reception. With the Raspberry Pi computer, the threshold takes an average 1 minute and 4 seconds to update, but this number goes up to 1 minute and 34 seconds if *gr-ais* is running. This difference is mainly because of the fact that the processor on the Raspberry Pi is slower than the one in the NUC. Another short test carried out was the evaluation of the channel switching time. According to the ITU-R M.1371-5 recommendation, it should take the system 25 ms or less to switch channels. This condition is met by both of the computers used.

5.3 Test 3: AIS Transmitter Message Timing Evaluation

The following test was done to verify that the implemented AIS transmitter sends messages according to their reporting interval R. The main objective was to determine whether or not a new message was sent every R seconds. To evaluate this, the system was started and it sent messages during 15 minutes after its initialization phase. In those 15 minutes, 32 messages should have been sent. In both computers the interval is correct in most of the transmissions. In the NUC computer, only one interval was not the expected. The situation is considerably different to that of the Raspberry Pi computer, in which a total of eight intervals were missed. This happens because the system is not able to process every time slot (as it should). Sometimes the slot in which the transmitter should send a message is not seen by the code. This usually results in the message being sent a minute later (once that slot is reached again in the next frame and the code is able to process it). A way of fixing this problem in slower computers like Raspberry Pi could be to analyze less samples. For example, the noise level of the channel could be computed using 1 out of 20 samples. This would mean that one sample will be considered representative of 20 samples. This is not always true and is less accurate. However, it would result in better performance.

5.4 Test 4: Message Delivery Evaluation

Aside from evaluating if the reporting intervals are correct, it is important to determine if messages are effectively being sent. This was tested in both of the considered computers. A third computer was used as an AIS receiver and a fourth one with *SDR++* tool that allowed us the visualize the spectrum. The transmitter was started and stopped once 20 messages were sent. In both computers, 90% or more of the messages were seen in the spectrum, meaning that they were effectively sent. However, less than 40% of those messages were interpreted by the receiver. A couple more tests were run to determine the source of this problem.

5.5 Complementary Tests: Identifying Issues in Test 4

In order to verify whether the *gr-ais* receiver was the issue, *AIS catcher* [16] was used. The same results were obtained. Once the receiver was out of the picture, we evaluated if the problem was in one of the aspects developed in this project, such as medium access or message generation. This was done using the *examples* flowgraph from the *gr-aistx* project. The flowgraph sends a fixed string of bits as messages and these are injected into the `AIS Frame Builder` block in Figure 6. Given that the same results (of about 40% of message reception) were seen, it became clear that message generation was not the source of the problem. One last test was carried out, to verify if the issue was related to the channel selection. The code and flowgraph were modified to send all messages in only one channel. The results were the same. Because the GMSK modulation is done with a GNU Radio block (which are usually optimized), it is believed that the problem is in the conformation of the frame in `AIS Frame Builder` or in some buffer of the `Pluto Sink` block.

It is important to clarify that in the previous tests, the messages sent by the implemented transmitter and received by the receiver were decoded using an online NMEA decoder [21]. All of the messages included correct position, speed and direction information, obtained through the GPS module.

5.6 Test 5: Access to the Shared Medium

The next test that was carried out was to determine if the transmitter detected other equipment using the channel. That is, if access to the

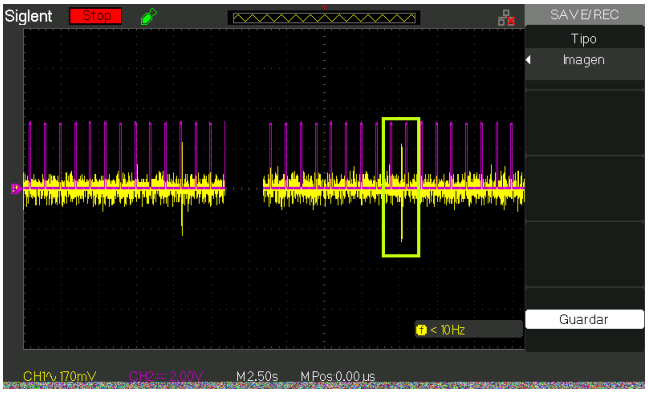

Figure 7: Oscilloscope Screen with a Message Transmission Marked in the Green Rectangle

shared medium was done correctly. An external noise signal could not be transmitted on a specific slot because it would require a lot of precision and because the transmission slots are chosen at random. In consequence, a noise signal was transmitted during 40 seconds so that this time period contained a scheduled transmission. In the NUC computer, in all transmissions the channel was classified as occupied and the message was re-scheduled for the next minute. In the Raspberry Pi this happened 80% of the times, mainly because in some cases the samples took extra time to enter `Potumbral` and the channel noise level was computed wrongly. It is important to mention that in both computers, 100% of the times the channel was free, it was classified as such and transmissions were carried out.

5.7 Test 6: Slot-specific Transmission Evaluation

It was verified that the system was able to enter the channel in an organized manner. However, it was still necessary to check if transmissions were carried out in the corresponding slot. The ITU-R M.1371-5 recommendation establishes that a transmission should be done in the slot classified as free and it should not exceed its duration. To determine whether the transmitter implemented complied with this requirement, an oscilloscope was used. On one channel, the antenna of the transmitter was connected, and on the other the PPS signal of the GPS was used as input. The first aspect to determine was if the transmission was effectively being done in the sensed slot. The PPS signal provided a UTC reference. The oscilloscope allowed us to verify when the message was seen in the channel. This is marked in Figure 7.

When the `Transmitter` block determines that a transmission should be done, the UTC time is printed in terminal. Then, the difference between the printed time and the time in which the signal is observed in the oscilloscope is the transmission delay. The average delay was of 127 ms, meaning that the transmission is done about 5 slots after it should be done. Apart from that the signal was analyzed with more detail to determine its duration. Each transmission is of about 40 ms, slightly exceeding the 27 ms established in the ITU-R M.1371-5 recommendation. Even though these numbers are not ideal, the results are considered good for a first prototype. The main cause of the delays are the processes of frame building and the encoding of bits.

6 CONCLUSIONS

In this work, the development of an AIS transceiver station was described. A prototype for a class B station was obtained, specifically considering a carrier sense medium access technique. The project was divided into two main modules, which were written and first tested separately, then as a unified system. A hardware prototype was also designed. All of this was done using the ITU-R M.1371-5 recommendation as reference.

In relation to the medium access module, simulations were carried out in order to verify that the strategy for accessing the shared medium was correct for the purpose of the project. These tests allowed to affirm that carrier sense TDMA was a good choice. A GNU Radio block was created in order to implement the channel sensing mechanism, including the computation of a dynamic threshold for channel noise level. The evaluation of the developed medium access turned out in great results, proving that the channel was only considered free when no other station was transmitting. In addition, if a computer with a fast enough processor is used, channel occupation is detected and its use is avoided.

As for message generation, all messages used by class B CS AIS stations were implemented. The building of frames was done using correct GPS data as payload. Messages were received by other simulated AIS stations and it was possible to correctly decode them. Aside from their content, the reporting intervals of messages were also coherent with the ITU-R M.1371-5 recommendation.

The design of the hardware prototype was simple. It was highly based on devices or parts to which the authors already had access to. However, the total price of the implemented station was of about half the price of class B AIS stations in Uruguay. The main goals of the project were then achieved, obtaining a low cost working transceiver station using SDR technology.

An out-of-tree GNU Radio module called *gr-itais* was developed. Its code is available on Gitlab [12], incorporating both the resulting blocks of this project and pieces of other projects such as *gr-ais* and *gr-aistx*. The code for the simplified simulations ran is also available on our repository.

6.1 Future Work

We would like to clarify that this article describes the development of a prototype. Several functionalities that are not covered here must be implemented for this station to be used in real life scenarios, as well as more detailed tests need to be carried out.

The developed transmitter sends both dynamic and static information to other stations successfully and automatically. A carrier sense, time division based medium access mechanism was implemented. It considers a UTC time reference in order to work properly in relation to other stations.

An instance of *gr-ais* can be run in the background to allow for message reception, although the interpretation of said messages is not implemented. The ITU-R M.1371-5 recommendation defines three modes of operation: autonomous, assigned and interrogated. Given that received messages are not interpreted, the latter modes of operation could not be done. Our system can not send safety data when requested by other stations, because it can't understand said requests. If message interpretation is implemented in the future, it could be a first step towards implementing the more complex

medium access mechanism, self-organized TDMA. Said strategy requires stations to listen to each other's slot reservations.

It was possible to sense the slots of interest once the `Transmitter` block had categorized them as candidates. However, transmission was carried out about five slots late. This is one of the most important aspects of the project in which more work should be done. The transmission delay needs to be optimized. It is also worth noting that, once the message is transmitted, the transmission has an average duration of 40 ms. This is longer than the defined slot duration, also resulting in issues.

Finally, no graphic interface was designed for the GNU Radio blocks created. This was due to the fact that most of the development was done using scripts for defining flowgraphs. It would be useful to have a graphic representation of the implemented blocks in the future.

REFERENCES

[1] Swedish Maritime Administration. 2019. AIS: How a Swedish Innovation Became a Global Standard.

[2] ais-dotnet github. 2019. AIS.Net.Receiver. https://github.com/ais-dotnet/Ais.Net.Receiver

[3] Peter Antypas. 2015. MAIANA. https://github.com/peterantypas/maiana

[4] Astruder. 2013. *dAISy*. https://github.com/astuder/dAISy

[5] Bistromath. 2020. gr-ais. https://github.com/bistromath/gr-ais/tree/master

[6] Bmagistro. 2020. gr-aistx for GNU Radio 3.8. https://github.com/bmagistro/gr-aistx.git

[7] Remi Challamel, Thibaud Calmettes, and Charlotte Neyret Gigot. 2012. A European Hybrid High Performance Satellite-AIS System. In *2012 6th Advanced Satellite Multimedia Systems Conference (ASMS) and 12th Signal Processing for Space Communications Workshop (SPSC)*. 246–252. https://doi.org/10.1109/ASMS-SPSC.2012.6333084

[8] Febus Reidj G. Cruz, Ryan Christopher M. Gania, Bryx William C. Garcia, and Jared Christian R. Nob. 2018. Implementing Automatic Identification System Transmitter on Software Defined Radio. In *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*. 1–4. https://doi.org/10.1109/HNICEM.2018.8666288

[9] Prefectura Nacional Naval de la República Oriental del Uruguay. 2004. Disposición Marítima Nº 94/04. https://www.armada.mil.uy/ContenidosPDFs/Prena/Dirme/disposiciones_maritimas/disposicion_maritima_94.pdf

[10] Domschl. 2020. RaspberryNtpServer. https://github.com/domschl/RaspberryNtpServer

[11] Riadh Essaadali, Chokri Jebali, Khaled Grati, and Ammar Kouki. 2015. AIS Data Exchange Protocol Study and Embedded Software Development for Maritime Navigation. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. 1594–1599. https://doi.org/10.1109/CCECE.2015.7129519

[12] Romina García and Máximo Pirri. 2024. gr-itais. https://gitlab.fing.edu.uy/rominag/itais

[13] Maciej Gucma. 2008. Low Cost AIS System for Safe Navigation. *Journal of Konbin* 6 (01 2008), 235–246. https://doi.org/10.2478/v10040-008-0070-2

[14] Kazuhiko Hasegawa, Kojiro Hata, Kazuhisa Niwa, and Junji Fukuto. 2008. Transmission Evaluation of Ship-borne Automatic Identification System (AIS) in Congested Waterways. In *2008 8th International Conference on ITS Telecommunications*. 18–23. https://doi.org/10.1109/ITST.2008.4740219

[15] Xu Hu, Bin Lin, Ping Wang, Hongguang Lyu, and Tie-Shan Li. 2021. A Collision Feedback based Multiple Access Control Protocol for Very High Frequency Data Exchange System in E-navigation. *Journal of Navigation* 74 (03 2021), 1–16. https://doi.org/10.1017/S0373463321000126

[16] jvder github. 2021. AIS-catcher. https://github.com/jvde-github/AIS-catcher

[17] Nicolás Molina Padrón. 2015. *Diseño e Implementación de un Prototipo Hardware para un Banco de Pruebas del Estándar AIS*. Thesis. https://accedacris.ulpgc.es/bitstream/10553/20972/1/0728673_00000_0000.pdf

[18] Richard Munoz. 2017. The Evolution of Communication through the Centuries. https://www.mobilecon2012.com/the-evolution-of-communication-through-the-centuries/

[19] Andy Norris. 2006. Automatic Identification Systems – the Effects of Class B on the Use of Class A Systems. *The Journal of Navigation* 59, 2 (2006), 335–347. https://doi.org/10.1017/S0373463306003766

[20] Máximo Pirri, Romina García, Gonzalo Belcredi, and Claudina Rattaro. [n. d.]. *Implementación de un Transmisor AIS de bajo Costo basado en SDR*. http://grupomontevideo.org/jji/2023/EjeCienciasExactas/23tecnologiasdelainformacionycomunicacion/Pirri_Fernandez_Maximo_255/Pirri_Fernandez_Maximo_255.pdf

[21] AGG Software. [n. d.]. AIS Online Decoder. https://www.aggsoft.com/ais-decoder.htm

[22] Trendmicro. 2020. gr-aistx. https://github.com/trendmicro/ais

[23] International Telecommunications Union. 1997. Recommendation ITU-R M.1313*: Technical Characteristics of Maritime Radionavigation Radars. https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1313-1-200005-W!!PDF-E.pdf

[24] International Telecommunications Union. 2005. Recommendation ITU-R M.1371-5 (02/2014): Technical Characteristics for an Automatic Identification System using Time Division Multiple Access in the VHF Maritime Mobile Frequency Band. https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1371-5-201402-I!!PDF-E.pdf

[25] Recursos Acuáticos y Prefectura Naval. 2014. Sistema de Identificación Automática de Amplio Alcance se Implementa en Pesca Artesanal. https://www.gub.uy/presidencia/comunicacion/noticias/sistema-identificacion-automatica-amplio-alcance-se-implementa-pesca