Encoding of ATN-B1 CPDLC for Cybersecurity Evaluation

Niklas Karlsson
Department of Computer and Information Science
Linköping University
Linköping, Sweden
nikka560@student.liu.se

Hampus Rosenquist
Department of Computer and Information Science
Linköping University
Linköping, Sweden
hamro777@student.liu.se

***Abstract*—Aeronautical Telecommunications Network Baseline 1 (ATN-B1) was developed to alleviate the ever-increasing congestion of airspace with a more capacity effective and reliable communication protocol. However, ATN-B1 was constructed without security in mind. This paper investigates potential security flaws in the protocols Controller-Pilot Data Link Communication (CPDLC) and VHF Data Link Mode 2 (VDL Mode 2) that are included in ATN-B1. This was done by the construction and use of encoders & decoders for the protocols to replicate the message chain and prove the ability of spoofing.**

I. Introduction

With the high air traffic amounts of today, secure Air Traffic Control (ATC) is of utmost importance. To handle this crowded air space, specifications such as Aeronautical Telecommunications Network Baseline 1 (ATN-B1) has been developed. ATN-B1 consists of several protocols, two of which are relevant to our topic, namely Controller-Pilot Data Link Communication (CPDLC) and VHF Data Link Mode 2 (VDL Mode 2).

The CPDLC protocol complements voice communication over Very High Frequency (VHF), delivering increased efficiency. VDL Mode 2 is the network that CPDLC uses to enable CPDLC communication between aircraft and ATC.

To evaluate and test attacks of said protocols, certain software and hardware are necessary. Inexpensive hardware and an open-source decoder of CPDLC over VDL Mode 2 already exist, but no encoder is available to the public, to the best of our knowledge.

In this paper, we intend to create an encoder of CPDLC messages for VDL Mode 2 communication and evaluate said protocols' cybersecurity. By doing so, future research can focus on evaluations rather than implementations of the communication chain.

II. Background

This section will present a brief introduction to each subject, software, and hardware that has been used in this project.

A. ATN-B1

ATN-B1 is the European version of an internetwork architecture that defines several protocols to enable ground-to-ground and air-to-ground communication over data links in the European airspace. While ATN-B1 was created in the early 1990s, the mandate to use it started as late as February 5, 2020, by the European Union Aviation Safety Agency (EASA).

The idea of the new architecture was to deliver a faster, more reliable, and more capacity-effective mean of communication in the European airspace. A capacity effective communication system was an important aspect to relieve the ever-increasingly congested airspace [1].

To enable this, VDL Mode 2 was specified to be used for radio communication, which can be read more about in the section below.

B. CPDLC

CPDLC is a protocol that defines the format for simple text-based communication between ATCs and pilots. The pilot can use CPDLC to report various information, request clearance, respond to ATC messages and declare or recall an emergency. The ATC can issue crossing constraints, level assignments, lateral deviations, speed- and radio frequency assignments, as well as request information from the pilot. It is also possible to send a free text message that does not conform to any pre-specified format.

CPDLC exists in two types, namely *FANS 1/A+ CPDLC* which is specified in the Future Air Navigation System (FANS), and *ATN-B1 CPDLC* which is specified in ATN-B1. Contrary to what the FANS name suggests, it is the simpler and more outdated version and is not mandated in either European or United States of America's domestic airspace [2]. In this paper, we focus solely on the ATN-B1 version of CPDLC.

C. VDL Mode 2

VDL Mode 2 is the second – and main – version of VHF Data Link which specifies the radio communication over VHF between aircraft and ground stations. It specifies both the modulation and several layers of headers. VDL Mode 2 is the network that ATN-B1 CPDLC is sent over.

The VDL frame structure is presented in figure 1 and the resulting data from the VDL frame structure is channel coded using bit scrambling with a pseudo-random sequence to avoid long sequences of 0:s or 1:s that might otherwise complicate synchronization.

Thereafter, the data is transmitted over VHF on the band 117.975 - 137.000 MHz with a 25 kHz wide channel. The channel is a time-division duplex (TDD) channel; meaning only one radio transmitter can be used at a time. However, in crowded areas, aircraft may be assigned a different frequency after the initial link establishment.

The modulation used is Differential 8 Phase Shift Keying (D8PSK). Using phase shifting of the carrier wave, relative to the previous element sent, to convey data, it transmits at a rate of 31 500 bits per second which translates to 10 500 symbols for D8PSK since three bits represent one symbol, 0-7 [3].

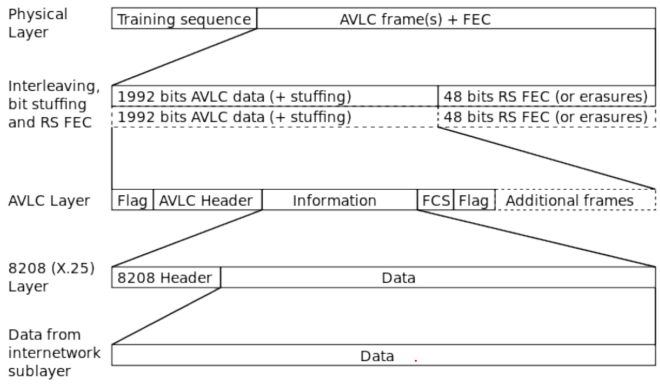

Figure 1. Complete VDL frame structure [3].

D. Hardware

1) HackRF One: is a Software-Defined Radio (SDR) device that can transmit and receive radio signals in the frequency span between 1 MHz to 6 GHz. It can be connected to a standard personal computer through USB and has software for Unix systems to interact with the device. Both hardware and software are open-source [4].

2) RTL-SDR: is an inexpensive radio receiver that can be connected to a computer through USB. It comes in different versions and models but can usually receive frequencies in the span of 500 kHz to 1.75 GHz. Open-source drivers are widely available [5].

E. Software

1) dumpvdl2: is an open-source VDL Mode 2 message decoder and protocol analyzer. It runs on Linux and supports a variety of SDR hardware, including RTL-SDR. It also supports an array of different protocols that use VDL Mode 2, including ATN-B1 CPDLC [6].

2) ATN-B1 CPDLC Encoder: As a part of Anton Magnusson's and Vilhelm Melkstam's bachelor's thesis that is discussed in section III-A, they created an encoder for the CPDLC part of ATN-B1 CPDLC, without VDL Mode 2 encoding due to time constraints. In this report we have continued their work, focusing on the encoding of VDL mode 2 to complete the communication chain of ATN-B1.

III. Related works

In this section, previous work done by others will be introduced to give a better understanding of the background of the subject as well as developed software.

A. Using Software-Defined Radio for ATN B1: A look into CPDLC encoding and VDL Mode 2 transmission

Anton Magnusson and Vilhem Melkstam evaluated CPDLC and VDL Mode 2 for potential vulnerabilities by threat modeling [7]. They concluded that spoofing, replay- and jamming-attacks were possible when transmitting CPDLC messages and could endanger the safety of airplanes using ATN-B1. While they did not construct the VDL Mode 2 part of the message chain, they believe that the vulnerabilities will remain even with VDL Mode 2 encoding since the protocol only regulates the mode of transmission and does not come with any security features that would hinder the attack.

B. Technical details of VDL Mode 2

Stefan Lundström constructed a basic overview of the technical details for VDL Mode 2 [3]. This paper disassembles the layers of VDL Mode 2 and gives an overview of what each layer consists of. The report does not go into detail about every part of each layer, but is nevertheless useful to get an overview of the transmission mode, and where to find a more detailed explanation.

C. Demonstrating ADS-B AND CPDLC Attacks with Software-Defined Radio

Gustafsson and Eskilsson showed the possibility to use HackRF SDR in order to transmit CPDLC and ADS-B messages [8]. By reconstructing their setup, one could transmit CPDLC messages with an HackRF One, and receive them with an RTL-SDR dongle with the help of a software called GNU Radio.

IV. Encoding of VDL Mode 2

Here an in-depth explanation of the encoding for VDL mode 2 will be introduced. This will contain both the hardware and software to properly run the encoder.

The VDL frame is made up of several layers shown in figure 1. It can also be divided into two categories, the data link layer, and the physical layer. Which part belongs to which will become clear below.

A. Data link layer

The encoding of the data link layer consists of constructing an 8208 (X.25) header and formatting the data into an Aviation VHF Link Control (AVLC) data structure [3].

1) 8202 (X.25) header: is also know as the SubNetwork Access Protocol (SNAcP) which in turn is based on the ISO 8208 standard (a.k.a. X.25) [3]. It's format can be seen in figure 2.

The header contains three fields in four octets. The first octet contains a version number that always takes on the value of 01. The second octet is the semantics field and can take one of four

SNAcP Header Format

Octet	Value	Meaning
1	01	Version number
2		Semantics of address:
	00	Not a multicast address
	01	All ESs
	02	All ISs
	03	Broadcast
3,4		OSI checksum as defined in ISO 8473

Figure 2. SNAcP header format [9].

values, and is determined by the layer above; the internetwork sublayer (CPDLC in our case) [9].

The last two octets consist of an OSI checksum that is defined in ISO 8473 and is constructed as follows [10]:

1: $C_0 \leftarrow C_1 \leftarrow 0$

2: Process each octet of the header sequentially from $i = 1$ to L by

$$C_0 \leftarrow C_0 \leftarrow O_i$$
$$C_1 \leftarrow C_1 \leftarrow C_0$$

3: Calculate:

$$X \leftarrow (L-8)C_0 - C_1 \mod 255$$
$$Y \leftarrow (L-7)(-C_0) + C_1 \mod 255$$

4: If $X = 0$, then $X \leftarrow 255$

5: If $Y = 0$, then $Y \leftarrow 255$

6: Place the values of X and Y in octets 3 and 4 respectively.

2) AVLC layer: is a slightly modified version of the High-level Data Link Control (HDLC) protocol which is presented in ISO 13239 and is formatted as shown in figure 3 [11]. In the original HDLC frame format, there is only one address field with a length of one octet. In AVLC both destination and source addresses exist and are four octets long each [3].

Figure 3. AVLC frame structure [3].

The flag field is always set to 01111110. While the control field is specified in figure 4 where the first row for I format is relevant for our case. $N(S)$ equals the transmitting send sequence number which is calculated by $N(S) = N(S)_{prev} + 1 \mod 7$, where $N(S)_{prev}$ is taken as input data.

$N(R)$ is the transmitting receive sequence number and is calculated by $N(R) = N(S)_{prev} + 2 \mod 7$.

P/F is the poll bit and is always set to 0 in our case, to present that the transmission is from a primary station and not a secondary- or combined station response frame transmission.

Following the control field is the actual data in its entirety from the above layer as seen in figure 1. Thereafter, a two-octet long frame checking sequence (FCS) is calculated and included per the algorithm presented in figure 5.

Control field format for	Control field bits*							
	1	2	3	4	5	6	7	8
Information transfer command/ response (I format)	0	N(S)			P/F	N(R)		
Supervisory commands/ responses (S format)	1	0	S	S	P/F	N(R)		
Unnumbered commands/ responses (U format)	1	1	M	M	P/F	M	M	M

Figure 4. Control field formats from ISO 13239.

The 16-bit FCS shall be the ones complement of the sum (modulo 2) of

a) the remainder of

$$x^k (x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$$

divided (modulo 2) by the generator polynomial

$$x^{16} + x^{12} + x^5 + 1,$$

where k is the number of bits being protected by the FCS and

b) the remainder of the division (modulo 2) by the generator polynomial

$$x^{16} + x^{12} + x^5 + 1$$

of the product of x^{16} by the content of the k bits being protected.

Figure 5. 16-bit frame checking sequence (FCS) from ISO 13239.

Last but not least, a final octet containing the flag represents the end of the frame.

It should also be noted that in case more than one AVLC frame is on queue to be transmitted, they may be included in a sequence inside the Information field, only separated with one flag [11].

The data link layer is now completed and is passed on to the encoding of the physical layer as demonstrated in figure 1.

B. Physical layer

1) Interleaving, bit stuffing, and RS FEC: In this part of the physical layer, the incoming AVLC data is separated into 249 octet blocks (1992 bits), each accompanied by a calculated Reed Solomon (RS) Forward Error Correction (FEC) field as protection against data corruption. If the incoming AVLC data is smaller than 249 octets (either before or after a potential split), the block is padded with zeros to fill out to 249 octets before generating the RS FEC field. The RS FEC is calculated using the python library *reedsolo*'s function RSCodec, with the input RSCodec(48, 2040) – representing a 48 bit (6 octets) long RS FEC for a total length of 2040 bits (255 octets) when the RS FEC is included. However, depending on the length of the message, not all RS FEC octets should be transmitted [3]. Namely:

- data ≤ 2 octets: no error correction
- $3 \leq$ data ≤ 30: the first two RS FEC octets
- $31 \leq$ data ≤ 67: the first four RS FEC octets
- $67 <$ data: all 6 RS FEC octets

However, all FEC octets should always be generated, and later removed according to the list above. After that, any

potential bit stuffing that was previously added is removed. Then, if multiple blocks were created, they are added after each other in sequence and passed on to the next layer as described in figure 1 [3].

2) Training sequence: Since VDL Mode 2's D8PSK modulation requires synchronization, a training sequence is generated based on a unique predetermined synchronization word, among other fields. The structure is presented in figure 6. The transmission length is taken as input and must specify the number of bits following the header FEC, while not including the RS FEC and padded bits in the AVLC frame. The input is transformed into a 17-bit format with the LSB first [3].

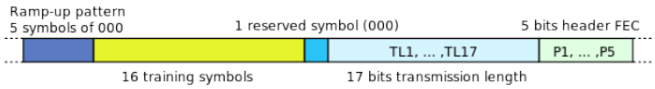

Figure 6. Training sequence structure. [3].

The five-bit FEC header is calculated as described in figure 7.

$$H = \begin{bmatrix} 0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1 \\ 0&0&1&1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1 \\ 1&1&0&0&0&1&1&1&0&0&1&1&0&0&0&0&1&1&1&1 \\ 1&1&0&1&1&0&1&1&0&1&0&1&0&0&1&1&0&0&1&1 \\ 0&1&1&0&1&0&0&1&1&1&1&0&0&1&0&1&0&1&0&1 \end{bmatrix}$$

The parity bits are calculated using the following equation:

$$[P_1, \ldots, P_5] = [R_1, \ldots, R_3, TL_1, \ldots, TL_{17}] H^T$$

Figure 7. Training sequence's FEC header calculation [3].

3) Channel coding: is applied on all bits following the training sequence and is done by using a pseudo-random (PN) sequence. As displayed in figure 8, the PN sequence is created by a 15-stage generator with a predetermined initial bit pattern [3].

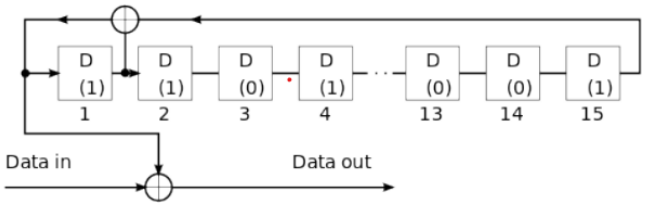

Figure 8. PN generator for channel coding [3].

4) D8PSK modulation: For the last step, GNU Radio is used to generate a python script that performs the D8PSK modulation. The GNU Radio schematic can be seen in figure 9. Input and output were handled with files, which can be seen in the schematics as *File Source* [12] and *File Sink* [13]. To match the VDL Mode 2 specifications, every symbol was represented by ten samples, and the sample rate was set to 105 kHz. Leading to a symbol rate of 10.5 k symbols per second.

The actual modulation is done by the *Constellation Modulator* [14] with appropriate settings for specifically D8PSK modulation, being

- Differential encoding = Yes
- Samples/Symbol = 10
- Excess bandwith = 1.06
- Constellation object =
  - Symbol map = [1,0,7,6,5,4,3,2]
  - Constellation points = [0.383+0.924j, 0.924+0.383j, 0.924-0.383j, 0.383-0.924j, -0.383-0.924j, -0.924-0.383j, -0.924+0.383j, -0.383+0.924j]
  - Rotational symmetry = 8
  - Dimensionality = 1
  - Normalization type = Amplitude
  - Soft decisions precision = 8
  - Soft decisions LUT = None

Apart from the modulation scheme presented in figure 9, a schematic with modulation, demodulation, and a graphical interface (GUI) was created as can be seen in figure 10. This was used to test the implementation. The rendered GUI can be seen in figure 11 in the appendix.

Thereafter, the open-source program hackrf [15] was used to communicate the generated signal with the HackRF One hardware over a local free frequency, such as 868 MHz here in Sweden.

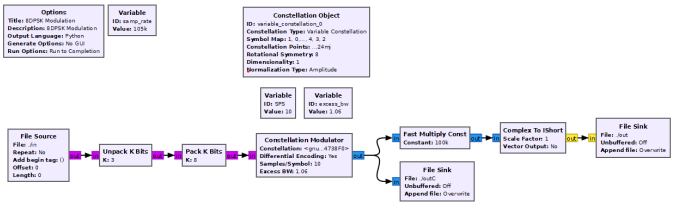

Figure 9. GNU Radio schema for D8PSK modulation.

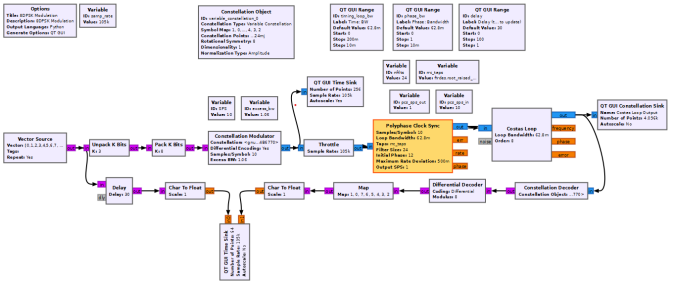

Figure 10. GNU Radio schema for D8PSK modulation and demodulation, with GUI.

V. Evaluation of CPDLC security patch

To secure the protocol from potential attacks that CPDLC has been proven vulnerable to [16], a secure key exchange patch has been suggested for CPDLC [17]. This security patch proposes an elliptic curve digital signature technique (ECQV) to secure the integrity of messages and authentication of identities. By securing these two properties, the existing vulnerabilities would no longer work, potentially securing the communication protocol. This patch comes with both a

computational and storage overhead, which may affect its usefulness since ATN-B1 has been constructed with speed in mind as delayed instructions may impact the possibility of properly executing an instruction by the pilot, endangering the aircraft.

As the patch consists of two messages, one from the ground which is 896 bits long, and one from the aircraft which is either 800 or 1088 bits long, one can calculate the worst-case scenario of the overhead. The worst-case scenario would be the longest key exchange followed by an empty message. Comparing this scenario with that of just sending the empty message we can see the largest overhead difference the security patch can achieve. The above-mentioned lengths are before the VDL Mode 2 encoding.

$$M_{keys} = M_{ground} + M_{aircraft} = \\ 1120 + 1336 + 224 = 2680 \text{ bits}$$

As seen in the equation above the largest difference created from the patch would result in 2456 bits after encoding the messages. Calculating a theoretical speed that does not take into account any computational time for parsing or encoding of the messages, 2546 bits would take 78 ms at the expected transfer rate of 31 500 bits/s. As such, it seems like the security patch is a reasonable extension to the protocol as the added overhead does not take up a significant time to send.

VI. Results

This section will present the resulting status of the encoder implementation, including any possible limitations or room for improvement.

All stages of encoding have been implemented. However, after putting all the pieces together, it is unfortunately not in working condition as this project comes to an end.

Testing the individual stages of encoding to a high degree of certainty of its correct functionality has been very difficult. The instructions presented in section IV make out the lion's share of the available information. Meaning, no access to any examples of correct output for different inputs to evaluate the implementations. Some testing was of course done to evaluate the different parts of the encoding, but for certain parts, it was hard to conclude with high confidence that it was correctly implemented.

The only concrete test available was testing the entire encoding chain against a decoder, like dumpvdl2. This of course is an integration test of sorts Big-Bang testing and all the consequences that come with that strategy.

Gathering and interpreting all the necessary information to implement the encoding turned out to be a large part of the project. Certain parts were vaguely defined and left room for interpretation, posing a further threat to the functionality of the final product.

As time runs out, we have not been able to identify what is malfunctioning. However, we do believe the implementation is in good progress towards completing the encoding chain of ATN-B1 CPDLC.

As for the result from the evaluation of the CPDLC security patch, it was concluded that the overhead introduced from the proposed security patch was within reason, regarding transmission data size and transfer rate.

VII. Conclusions

This section will conclude what the results mean for both the protocols' cybersecurity and future research on the subject.

While it proved hard to complete the communication chain for ATN-B1 CPDLC within the limited scope of this project, a foundation was created that enables further development that may take large advantage of the work already done. This means researchers are one step closer to accessing a solution that enables practical evaluation of the ATN-B1 CPDLC protocol independently and in turn, contributes to the protocols' cybersecurity.

Having access to a complete communication chain for independent researchers may enable testing proposed security patches, such as [17] presented earlier, in practice and stress test the implementations.

Therefore, we believe our work will contribute, at the very least as a proof of concept, to the protocols' cybersecurity and future research on the subject.

References

[1] UniversalAvionics, "Understanding Data Comm Systems with Domestic and Oceanic FANS 1/A+ and ATN B1 Services." November 2020.

[2] GlobalAerospaceDesignCorporation, "Controller-Pilot Data Link Communications (CPDLC)." February 2021.

[3] S. Lundström, "Technical details of VDL Mode 2." 2016.

[4] GreatScottGadgets, "HackRF One," 2021.

[5] RTL-SDR.com, "ABOUT RTL-SDR," 2013.

[6] T. Lemiech, "Github: szpajder/dumpvdl2," 2021.

[7] V. Melkstam and A. Magnusson, "Using Software-Defined Radio for ATN B1: A look into CPDLC encoding and VDL Mode 2 transmission," bachelor's thesis, March 2022.

[8] S. Eskilsson, H. Gustafsson, S. Khan, and A. Gurtov, "Demonstrating ADS-B AND CPDLC Attacks with Software-Defined Radio," in *2020 Integrated Communications Navigation and Surveillance Conference (ICNS)*, pp. 1B2–1–1B2–9, 2020.

[9] R. Hagens, N. Hall, and M. Rose, "Use of the Internet as a Subnetwork for Experimentation with the OSI Network Layer (RFC 1070)," February 1989.

[10] ISO/IEC-8473, "Information technology — Protocol for providing the connectionless-mode network service: Protocol specification," November 1998.

[11] ISO/IEC-13239, "Information technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures," 2002.

[12] GNURadioWiki, "File Source," 2022.

[13] GNURadioWiki, "File Sink," 2022.

[14] GNURadioWiki, "Constellation Modulator," 2021.

[15] GreatScottGadgets, "Github: greatscottgadgets/hackrf," 2021.

[16] A. Lehto, I. Sestorp, S. Khan, and A. Gurtov, "Controller Pilot Data Link Communication Security: A Practical Study," in *2021 Integrated Communications Navigation and Surveillance Conference (ICNS)*, pp. 1–11, 2021.

[17] S. Khan, A. Braeken, P. Kumar, and A. Gurtov, "Securing Ground–Air Communication: A Robust Protocol for Controller-Pilot Data Link Communication." March 2022.

APPENDIX

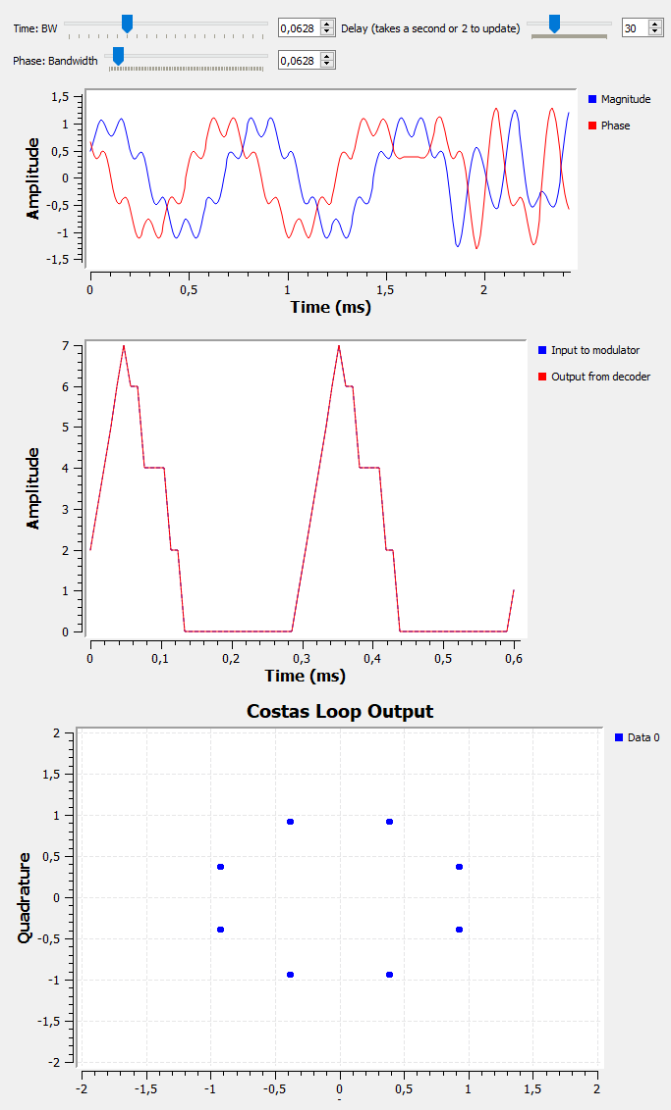

Figure 11. GNU Radio generated GUI produced by schema in figure 10.