

**Relatório do Segundo Trabalho Prático**

|  |  |
| --- | --- |
| Grupo | 13 |
| 38866 | Manuel Dias |

|  |  |
| --- | --- |
| Docente | Artur Ferreira |
|  |  |

Relatório final realizado no âmbito da Unidade Curricular de Comunicações,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2019/2020

Junho de 2020

Índice

[Índice de Figuras 2](#_Toc44048559)

[1. Objetivos 3](#_Toc44048560)

[2. Grupo 1 4](#_Toc44048561)

[2.1 Alínea A 4](#_Toc44048562)

[2.1.1 Códigos de Linha 5](#_Toc44048563)

[2.1.1.1 Ausência de Código de Linha 5](#_Toc44048564)

[2.1.1.2 NRZ Bipolar Absoluto 6](#_Toc44048565)

[2.1.1.3 NRZ Unipolar Mark 6](#_Toc44048566)

[2.1.2 Modulação Digital 7](#_Toc44048567)

[2.1.2.1 Ausência de Modulação Digital 7](#_Toc44048568)

[2.1.2.2 Modulação OOK 8](#_Toc44048569)

[2.1.2.3 Modulação FSK 8](#_Toc44048570)

[2.1.3 Canal Físico 9](#_Toc44048571)

[2.2 Alínea B 10](#_Toc44048572)

[2.3 Alínea C 15](#_Toc44048573)

[2.3.1 Transmissão de um ficheiro de texto 15](#_Toc44048574)

[2.3.1 Transmissão de um ficheiro de imagem 16](#_Toc44048575)

[3. Grupo 2 19](#_Toc44048576)

[4. Grupo 3 20](#_Toc44048577)

[4.1 Alínea (i) 20](#_Toc44048578)

[4.2 Alínea (ii) 21](#_Toc44048579)

# Índice de Figuras

[Figura 1 - Representação da GUI totalmente modificada 4](#_Toc44048580)

[Figura 2 - Representação do código de linha na GUI 5](#_Toc44048581)

[Figura 3 - Representação da modulação digital na GUI 7](#_Toc44048582)

[Figura 4 - Representação do Ruído de Canal na GUI 9](#_Toc44048583)

[Figura 5 - Emissão de dados com NRZ Bipolar Absoluto 10](#_Toc44048584)

[Figura 6 - Receção de dados com NRZ Bipolar Absoluto 10](#_Toc44048585)

[Figura 7 - Ficheiro resultante de uma transmissão NRZ Bipolar Absoluto 10](#_Toc44048586)

[Figura 8 - Emissão de dados com NRZ Unipolar Mark 11](#_Toc44048587)

[Figura 9 - Receção de dados com NRZ Unipolar Mark 11](#_Toc44048588)

[Figura 10 - Ficheiro resultante de uma transmissão NRZ Unipolar Mark 11](#_Toc44048589)

[Figura 11 - Emissão de dados com modulação OOK 12](#_Toc44048590)

[Figura 12 - Receção de dados com modulação OOK 12](#_Toc44048591)

[Figura 13 - Ficheiro resultante de uma transmissão com modulação OOK 12](#_Toc44048592)

[Figura 14 - Emissão de dados com modulação FSK 13](#_Toc44048593)

[Figura 15 - Receção de dados com modulação FSK 13](#_Toc44048594)

[Figura 16 - Ficheiro resultante de uma transmissão com modulação FSK 13](#_Toc44048595)

[Figura 17 - Receção de dados com modulação FSK com valor de SNR alterado 14](#_Toc44048596)

[Figura 18 - Receção de dados com modulação FSK com valor de atenuação do canal alterado 14](#_Toc44048597)

[Figura 19 - Sinal de emissão do ficheiro de texto "a.txt" codificado 15](#_Toc44048598)

[Figura 20 - Sinal de receção do ficheiro de texto "a.txt" codificado 16](#_Toc44048599)

[Figura 21 - Conteúdo do ficheiro original 16](#_Toc44048600)

[Figura 22 - Conteúdo do ficheiro transmitido 16](#_Toc44048601)

[Figura 23 - Sinal de emissão do ficheiro de texto "bird.gif" codificado 17](#_Toc44048602)

[Figura 24 - Sinal de receção do ficheiro de texto "bird.gif" codificado 17](#_Toc44048603)

[Figura 25 - Resultado da tentativa de descodificar o sinal recebido 18](#_Toc44048604)

[Figura 26 - Resultados do script de testes das funções de BI 20](#_Toc44048605)

[Figura 27 - Resultados do script de testes das funções de ISBN 21](#_Toc44048606)

# 1. Objetivos

Este trabalho tem como objetivos:

* Desenvolver programas e aplicações em MATLAB/OCTAVE;
* Estudo e aplicação de conceitos sobre codificação de fonte, cifra, correção de erros e desempenho de sistemas de transmissão digital, ao nível físico;

# 2. Grupo 1

No primeiro grupo do trabalho prático, é pretendido adicionar funcionalidades à GUI do trabalho anterior, adicionando 3 sub-blocos ao bloco CSD: emissor, canal e recetor, e testar as suas funcionalidades.

## 2.1 Alínea A

As funcionalidades adicionadas foram separadas em 3 partes:

1. Suporte de códigos de linha;
2. Suporte de modulações digitais;
3. Suporte de simulação de um canal físico;

Em todas as funcionalidades, existem 2 parâmetros genéricos que podem ser escolhidos pelo utilizador, o tempo de bit em segundos e o número de amostras por cada bit. Este último parâmetro não convém ser um número muito elevado, pois, dependendo do ficheiro a analisar, este vai indicar a resolução do sinal e quanto maior for a resolução, mais memória vai consumir. Estes parâmetros podem ser verificados na Figura 1 no canto inferior direito.

Opcionalmente, foi adicionado um modo janela que mostra 10 bits no gráfico da GUI, quando houver uma simulação e o utilizador carregar num dos 2 botões do sub-bloco de código de linha ou de modulação digital. Esta janela pode ser movível no eixo do tempo de transmissão, deslocando a barra por debaixo do gráfico.

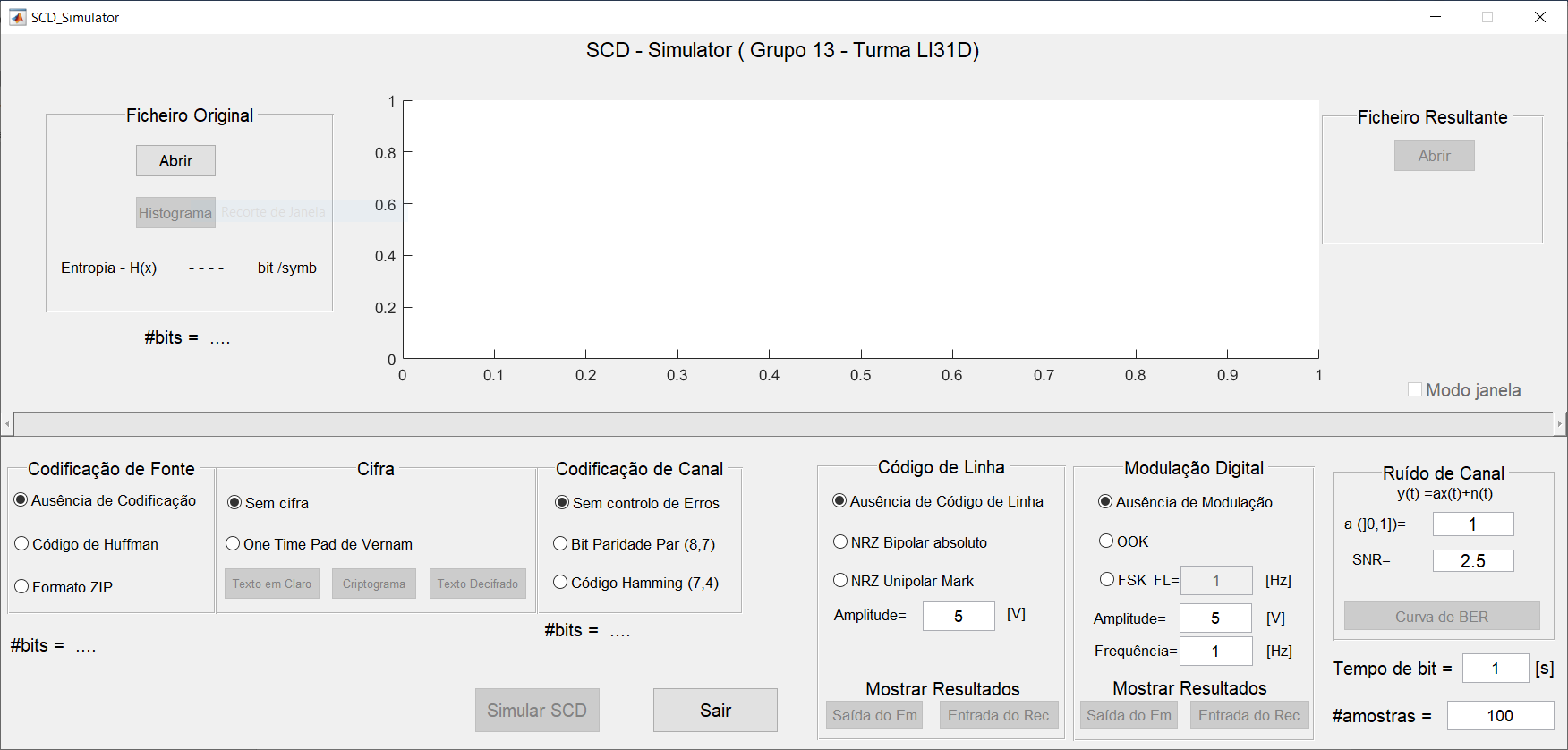


Figura - Representação da GUI totalmente modificada

### 2.1.1 Códigos de Linha

O código de linha está dividido em 3 partes:

1. Ausência de código de linha;
2. NRZ Bipolar Absoluto;
3. NRZ Unipolar Mark;

Esta implementação permite ao utilizador escolher que tipo de código de linha quer e, para além disto, o utilizador pode escolher a amplitude de cada pulso em Volts, como pode ser verificado na Figura 1 e na Figura 2.

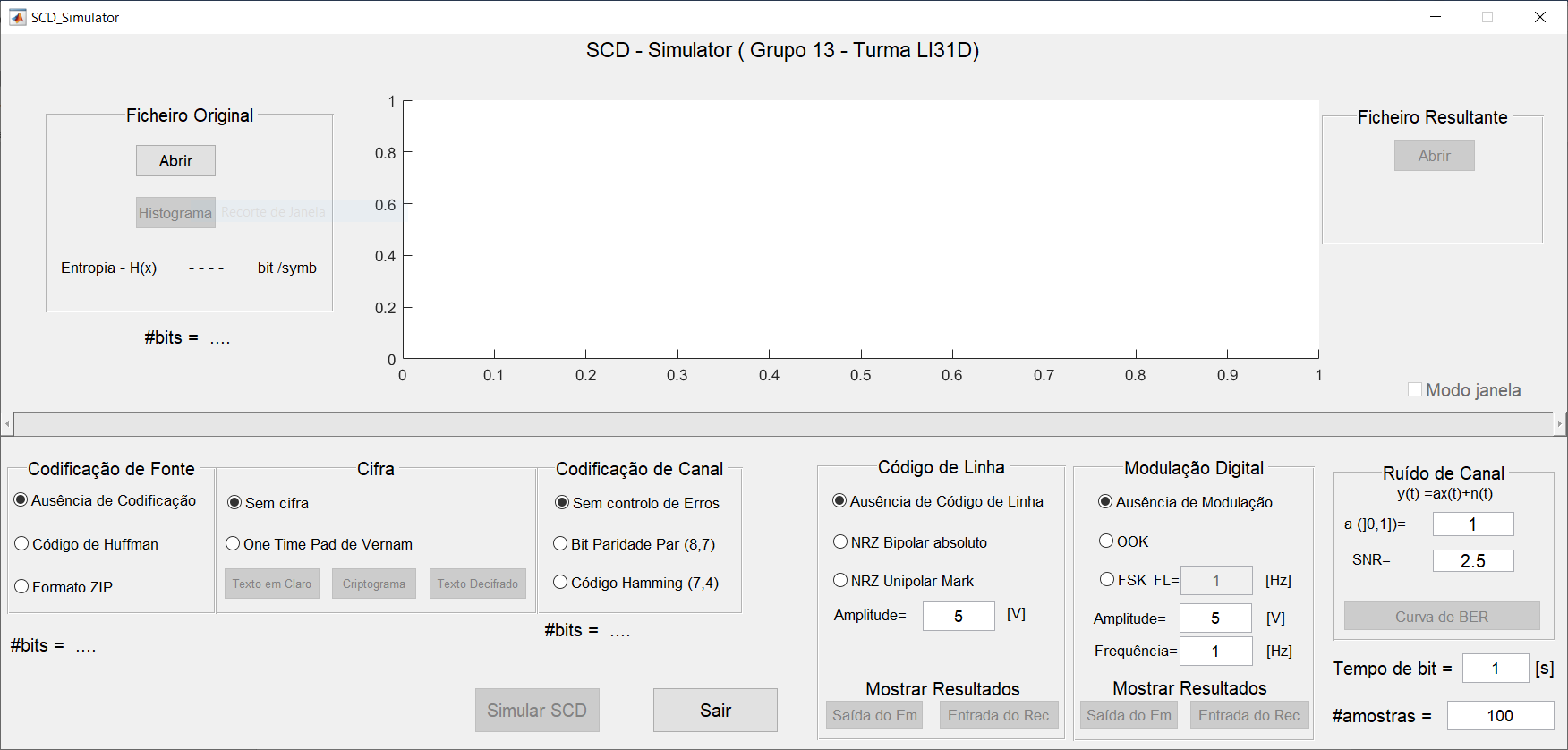


Figura - Representação do código de linha na GUI

#### 2.1.1.1 Ausência de Código de Linha

Na emissão de pulsos digitais não modulados, é necessário dos dados a transmitir a partir de um ficheiro, da amplitude de cada pulso, o número de amostras de cada pulso e o tempo de bit. A partir disto é possível transformar os dados do ficheiro num Stream de pulsos em que cada pulso pode ter entre 2 valores, ou 0, caso o bit seja 0, ou o valor de amplitude se o bit for 1.

Esta função, *Em\_Aus*, retorna 3 vetores:

1. Vetor de pulsos;
2. Vetor de tempo;
3. Vetor de energia de cada bit;

Estes vetores são essenciais para representar graficamente os pulsos, no caso do vetor de pulsos e do tempo, como também para simular e calcular o ruído de canal, no caso do vetor de energia de cada bit.

Na receção de pulsos digitais não modulados, é feito o procedimento inverso, escrevendo, no final, num ficheiro cujo nome é a concatenação de “CLA\_” com o nome do ficheiro passado na emissão.

#### 2.1.1.2 NRZ Bipolar Absoluto

Na emissão de pulsos do tipo NRZ Bipolar Absoluto, com a técnica de bit-suffing com blocos de 6 bits, é necessário o mesmo número de parâmetros que na Ausência de Código de Linha. No entanto, antes de transformar os bits em pulsos, os bits são postos em blocos de 6 ou 7 bits. Para tal, foi feita a função *Generate\_Bip\_Block* que gera blocos de N bits, neste caso 6, se houver transições de bits dentro desse bloco, caso contrário adiciona um bit inverso ao primeiro bit do bloco para o recetor detetar uma transição. A transição destes blocos para pulsos é feita quase da mesma maneira que na funcionalidade de ausência. A diferença está quando um bit no bloco for 0, o pulso não terá o valor 0, mas sim o valor negativo da amplitude.

Na receção de pulsos do tipo NRZ Bipolar Absoluto, com a técnica de bit-suffing com blocos de 6 bits, é feito o procedimento inverso, no entanto, a cada 6 pulsos, se não detetar uma transição, então o bit a seguir é ignorado, pois este é o bit de deteção de transição. No final é gravado os dados num ficheiro cujo nome é a concatenação de “NRZBA\_” com o nome do ficheiro recebido no emissor.

#### 2.1.1.3 NRZ Unipolar Mark

Na emissão de pulsos do tipo NRZ Unipolar Mark, com a técnica de bit-suffing com blocos de 6 bits, é necessário o mesmo número de parâmetros que na Ausência de Código de Linha. No entanto, antes de transformar os bits em pulsos, os bits são postos em blocos de 6 ou 7 bits. Para tal, foi feita a função *Generate\_Uni\_Block* que gera blocos de N bits, neste caso 6, se houver transições de bits dentro desse bloco, caso contrário adiciona um bit inverso ao primeiro bit do bloco para o recetor detetar uma transição. Para fazer a transição dos blocos para pulsos, é preciso definir primeiro, o valor inicial, mark, do sinal que foi 0. Este valor, inicialmente, tanto pode ser 0 como 1, mas tem de ser o mesmo entre o emissor e o recetor. Depois, durante o ciclo da transformação, cada vez que é detetado o bit 1, o valor de mark altera para , para fazer a transição de sinal. Este sinal está compreendido entre a amplitude escolhida pelo utilizador e 0.

Na receção de pulsos do tipo NRZ Unipolar Mark, com a técnica de bit-suffing com blocos de 6 bits, é feito o procedimento inverso, no entanto, a cada 6 pulsos, se não detetar uma transição, então o bit a seguir é ignorado, pois este é o bit de deteção de transição. No final é gravado os dados num ficheiro cujo nome é a concatenação de “NRZUM\_” com o nome do ficheiro recebido no emissor.

### 2.1.2 Modulação Digital

A modulação digital está dividida em 3 partes:

1. Ausência de modulação;
2. OOK;
3. FSK;

Esta implementação permite ao utilizador escolher que tipo de modulação quer e, para além disto, o utilizador pode escolher a amplitude de sinal em Volts como a frequência em Hertz, como pode ser verificado na Figura 1 e na Figura 3. Caso o utilizador escolha a opção FSK, a frequência à frente da opção indica a frequência quando um bit for 0.

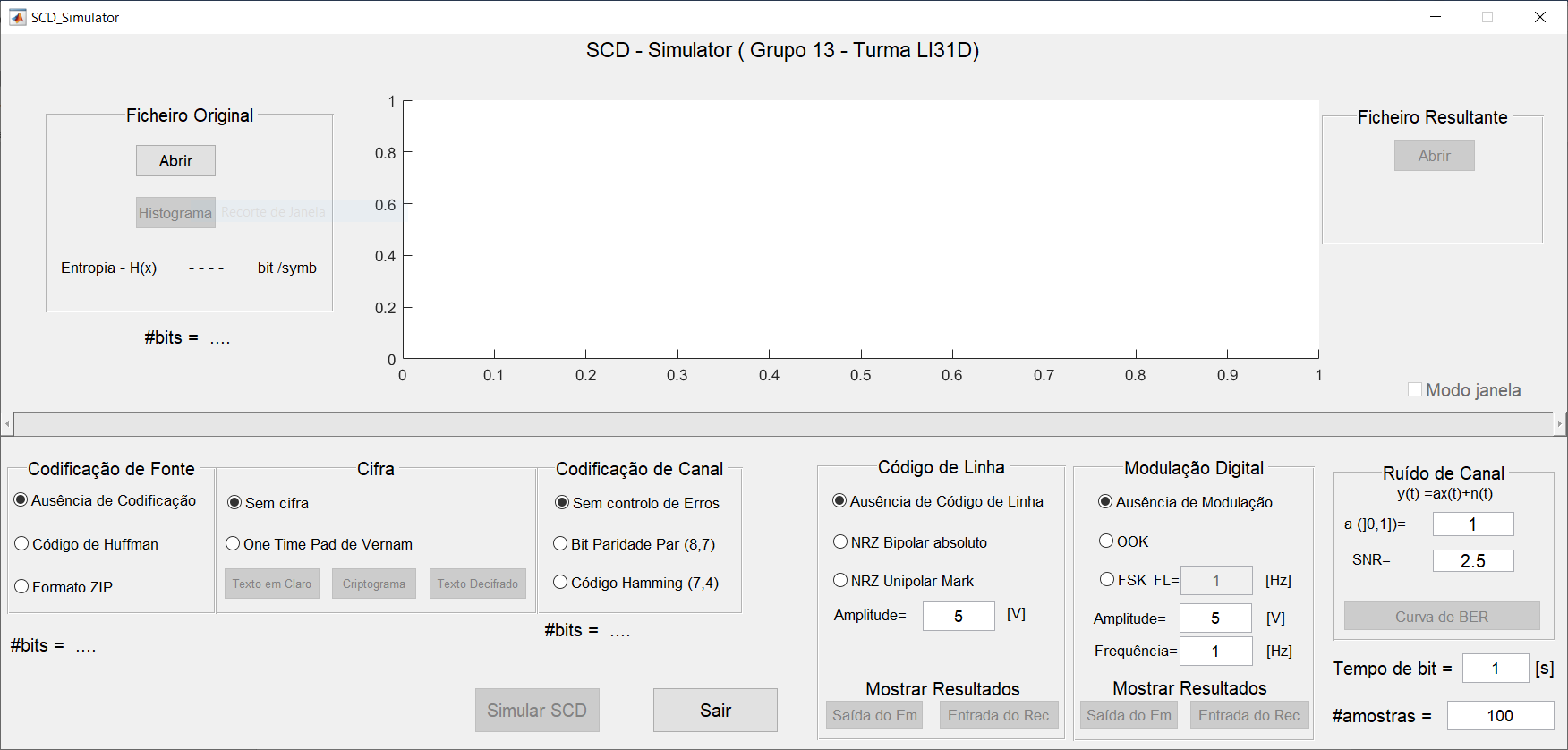


Figura - Representação da modulação digital na GUI

#### 2.1.2.1 Ausência de Modulação Digital

Neste modo, como não existe modulação, os dados à saída do emissor são iguais aos dados à saída do emissor de código de linha.

Na receção, é menos simples, pois o sinal pode vir com ruído. Para resolver isto, é calculado a energia de bit recebido em cada pulso. Se o código de linha escolhido for o NRZ Bipolar, então é verificado se esta energia é positiva ou negativa, devolvendo a amplitude de um pulso negativo caso a energia seja positiva. Caso contrário o pulso devolvido vem com amplitude negativa. Nos outros tipos de código de linha, é comparada esta energia com a energia de um pulso sem ruído e caso que a energia calculada com ruído for superior a metade da energia de um pulso sem ruído, então devolve um pulso com amplitude escolhida pelo utilizador, caso contrário o pulso terá uma amplitude de 0.

#### 2.1.2.2 Modulação OOK

Para fazer a emissão da modulação digital do tipo OOK, é necessário receber o vetor de dados a modular, a amplitude usada no código de linha, o tempo de bit, a amplitude de modulação escolhida pelo utilizador, a frequência e o número de amostras. Depois, num ciclo, é verificado pulso a pulso, se a amplitude do mesmo corresponde à amplitude do código de linha. Se for igual, então é gerado um sinal sinusoidal, caso contrário, o sinal é 0 no tempo de bit corrente.

Para a receção de sinais com modulação digital do tipo OOK é necessário calcular a energia absoluta do sinal, pois a partir disto é possível verificar se a amplitude de cada sinal de bit é igual ou superior à amplitude escolhida pelo utilizador. Se for superior ou igual, então o sinal representa um valor positivo, gerando um pulso com uma amplitude escolhida no código de linha. Caso contrário, o pulso terá ou uma amplitude com valor 0 ou com um valor negativo, dependendo do código de linha escolhido.

#### 2.1.2.3 Modulação FSK

Para fazer a emissão da modulação digital do tipo FSK, é necessário receber o vetor de dados a modular, a amplitude usada no código de linha, o tempo de bit, a amplitude de modulação escolhida pelo utilizador, a frequência que representa o bit 1, a frequência que representa o bit 0 e o número de amostras. Depois, num ciclo, é verificado pulso a pulso, se a amplitude do mesmo corresponde à amplitude do código de linha. Se for igual, então é gerado um sinal sinusoidal com a frequência com bit 1, caso contrário, é gerado com a frequência com bit 0.

Para a receção de sinais com modulação digital do tipo FSK é necessário calcular a frequência fundamental do sinal, pois a partir disto é possível verificar a frequência de cada sinal de bit. Se esta for superior ou igual à frequência do bit 1 ao quadrado, então o sinal representa um bit 1, gerando um pulso com uma amplitude escolhida no código de linha. Caso contrário, o pulso terá ou uma amplitude com valor 0 ou com um valor negativo, dependendo do código de linha escolhido.

### 2.1.3 Canal Físico

A simulação do canal físico é feita seguindo a expressão . Isto significa que é necessário que o utilizador insira um valor de atenuação do canal, α, e a relação sinal-ruído, SNR, para adicionar ruído ao canal. Isto pode ser verificado na Figura 1 e 4.

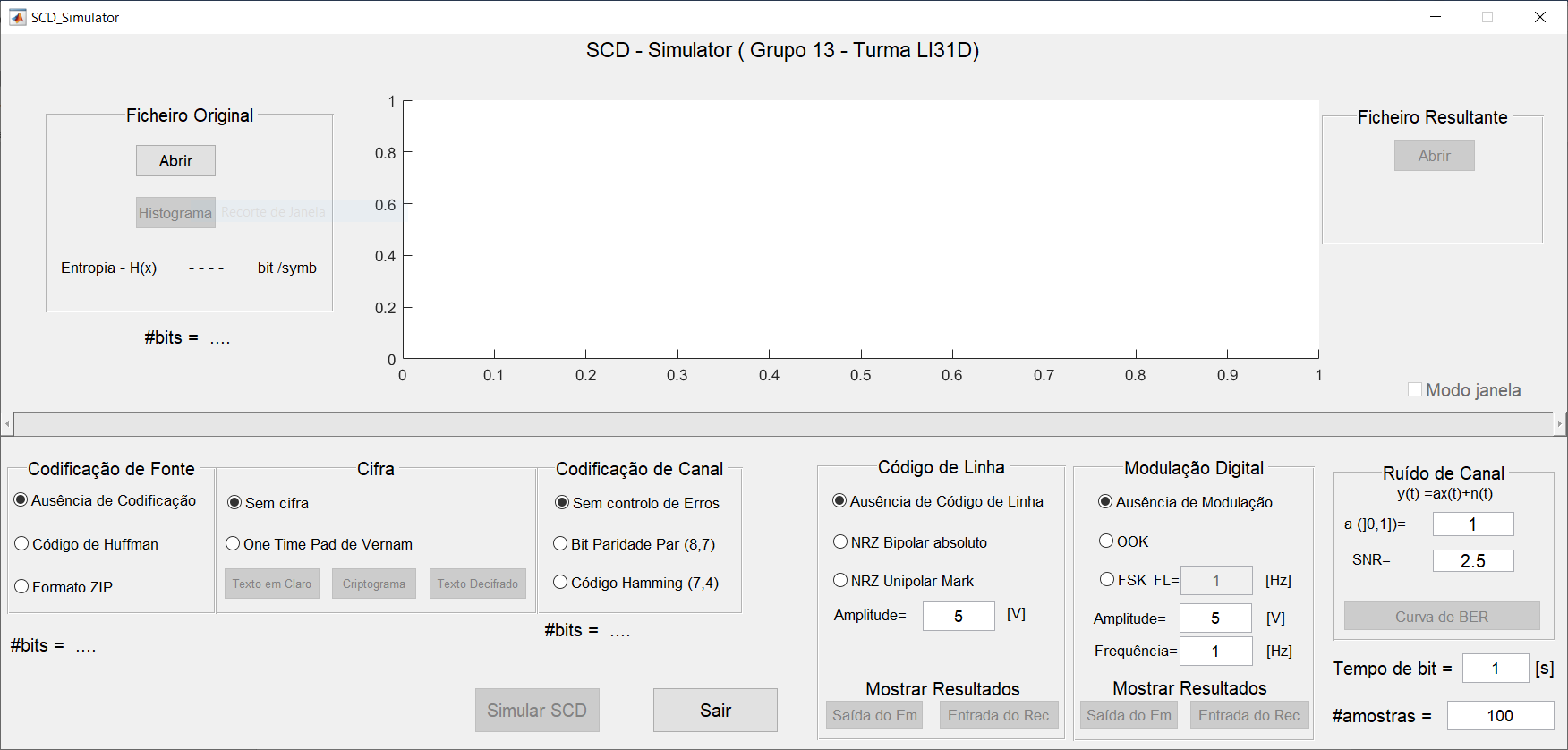


Figura - Representação do Ruído de Canal na GUI

Para calcular a adição de ruído, é preciso saber o SNR e a energia de sinal. A partir destes dois fatores, é possível calcular a energia do ruído e depois utiliza-se um gerador de números aleatórios normalmente distribuídos para simular o ruído. Por último é devolvido ao sinal de saída, os sinais de entrada atenuados com o ruído.

## 2.2 Alínea B

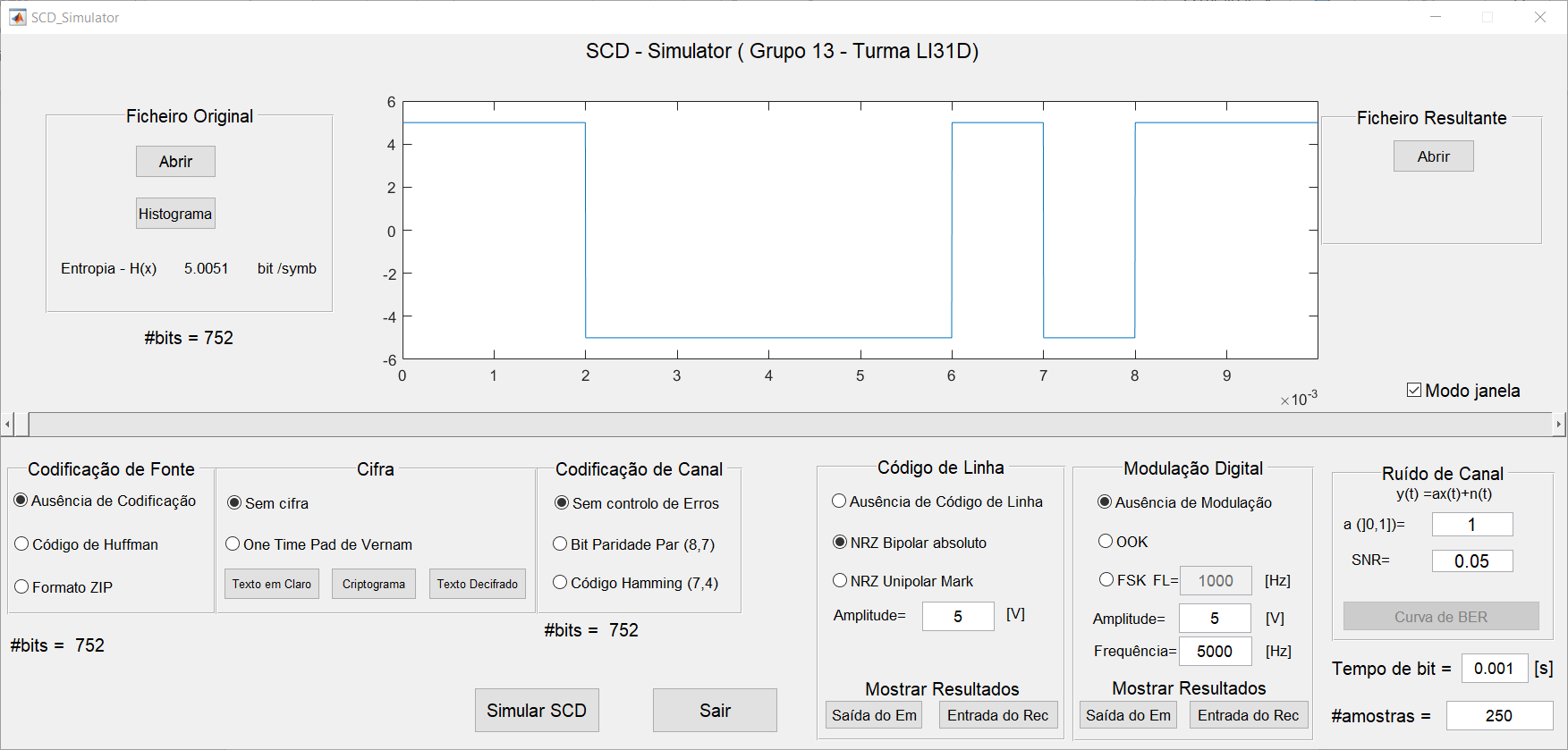


Figura - Emissão de dados com NRZ Bipolar Absoluto

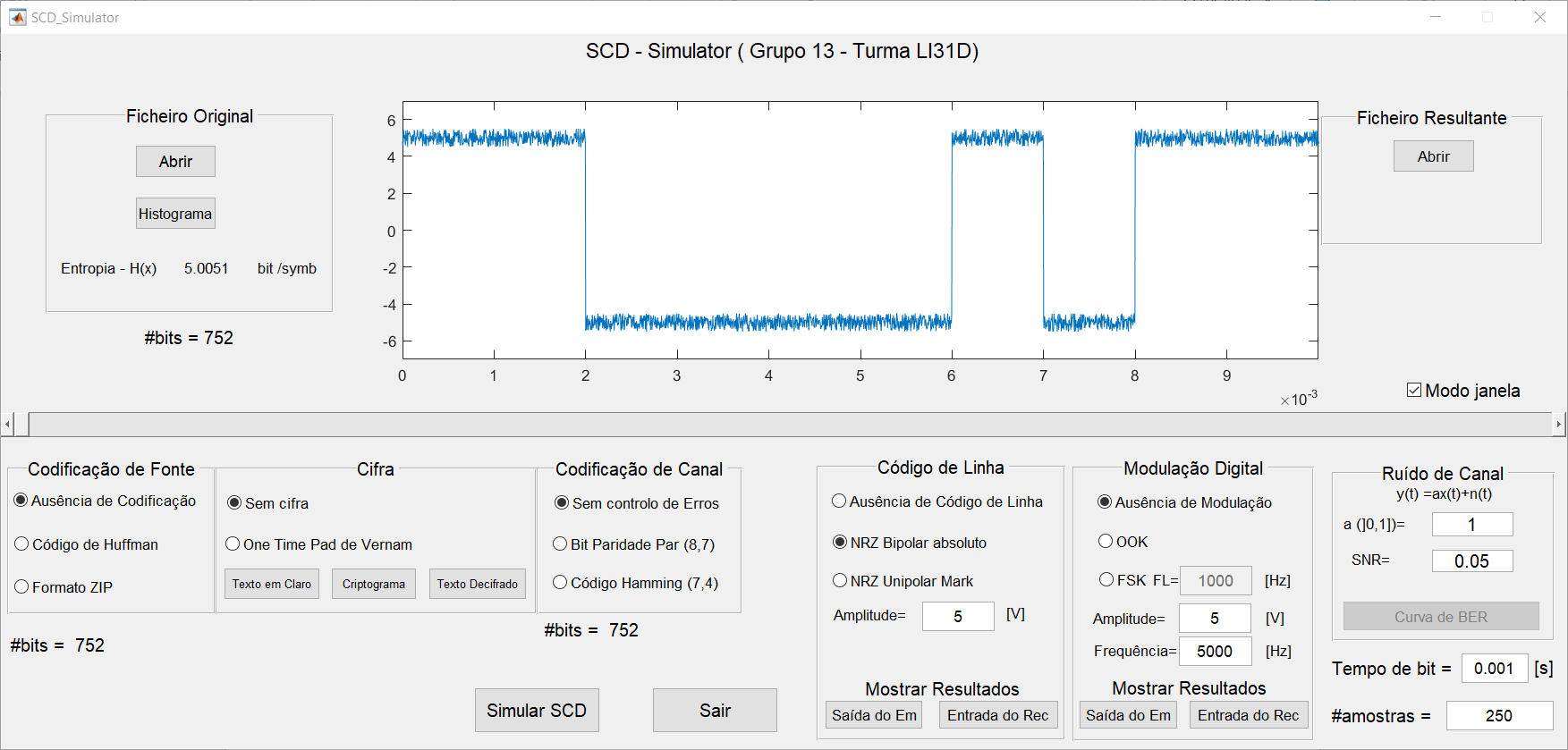


Figura - Receção de dados com NRZ Bipolar Absoluto

Na Figura 5 é possível verificar a emissão de dados com código de linha NRZ Bipolar com uma amplitude de 5 a -5 Volts e um tempo de bit de 1 milissegundo. Na Figura 6 é possível verificar a receção dos mesmos dados, mas com uma adição de ruído na amplitude, no entanto, é visivelmente possível perceber os bits a 0 e a 1. Na Figura 7 está representado o ficheiro resultante da transmissão.

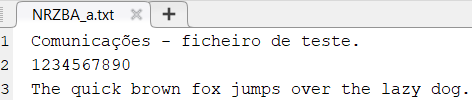


Figura - Ficheiro resultante de uma transmissão NRZ Bipolar Absoluto

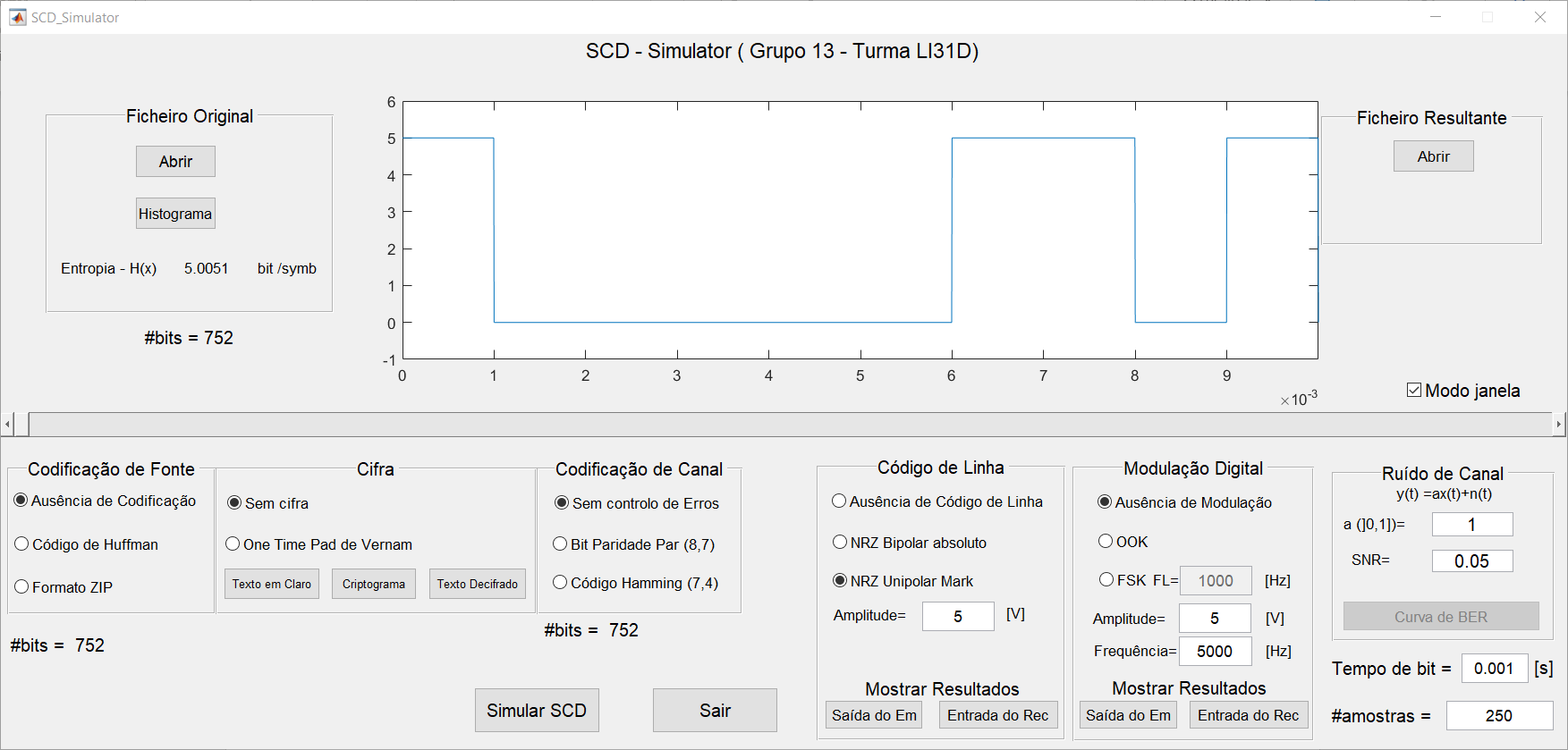


Figura - Emissão de dados com NRZ Unipolar Mark

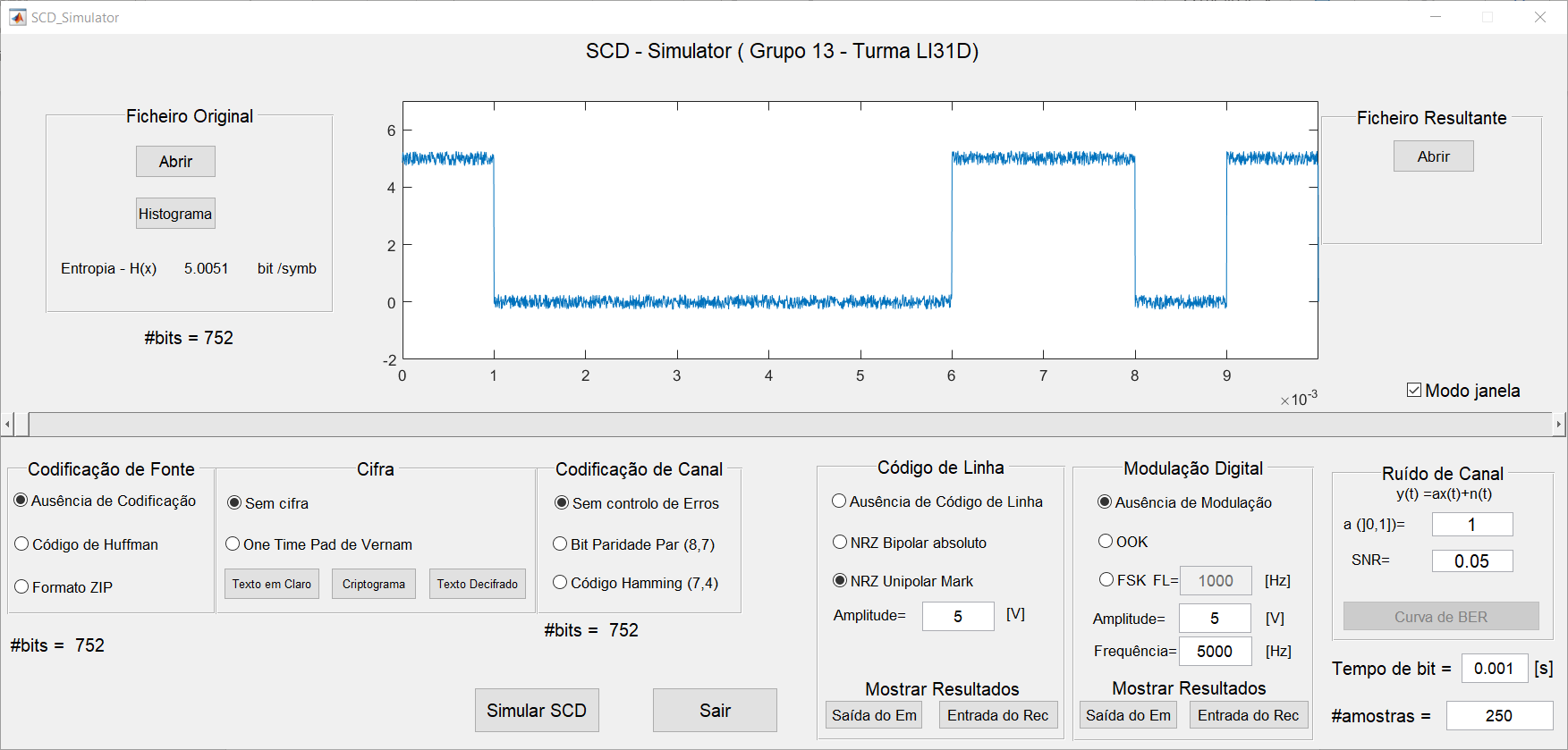


Figura - Receção de dados com NRZ Unipolar Mark

Na Figura 8 é possível verificar a emissão de dados com código de linha NRZ Unipolar com uma amplitude de 5 a 0 Volts e um tempo de bit de 1 milissegundo. Na Figura 9 é possível verificar a receção dos mesmos dados, mas com uma adição de ruído na amplitude, no entanto, é visivelmente possível perceber os bits a 0 e a 1. Na Figura 10 está representado o ficheiro resultante da transmissão.

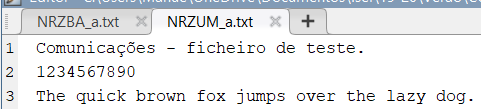


Figura - Ficheiro resultante de uma transmissão NRZ Unipolar Mark

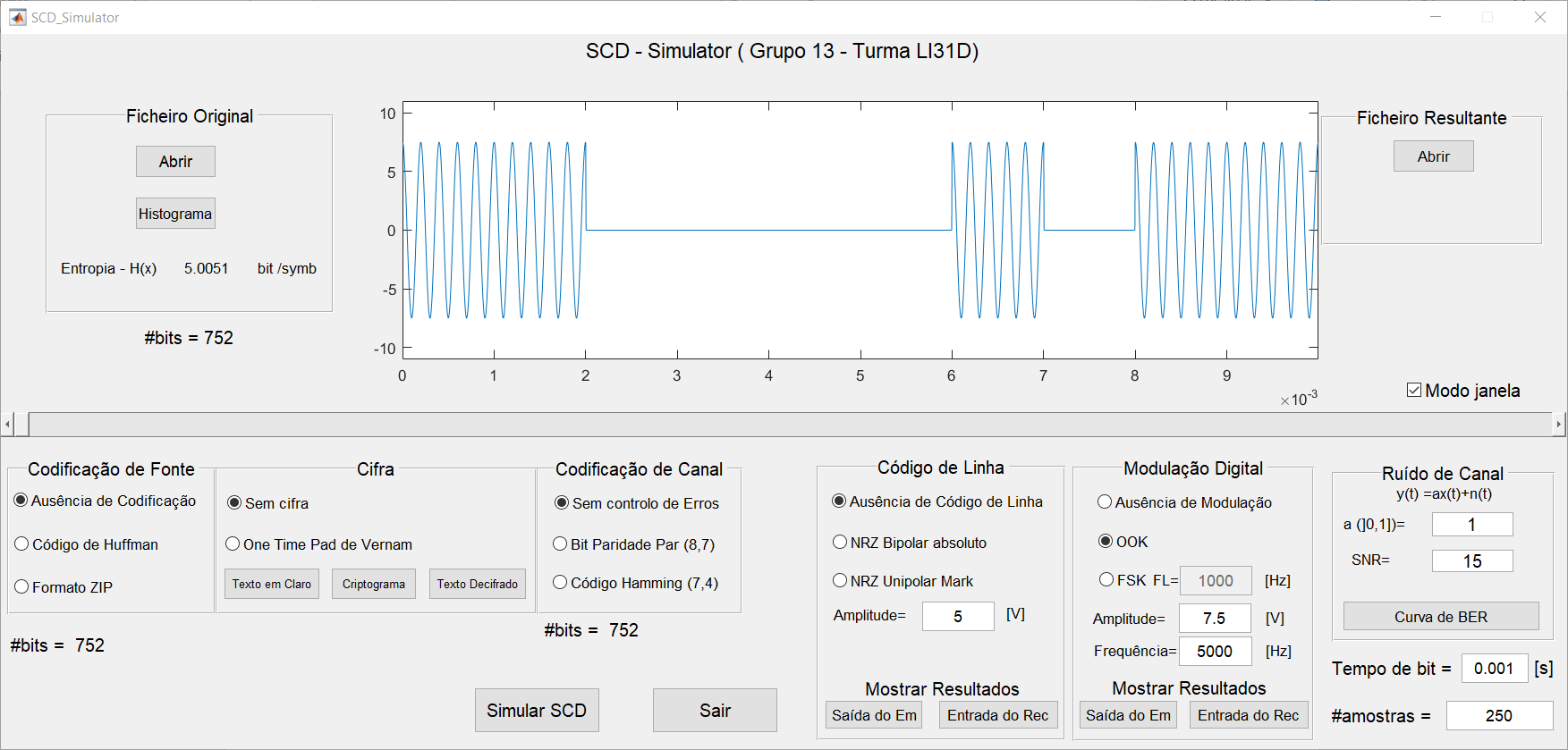


Figura - Emissão de dados com modulação OOK

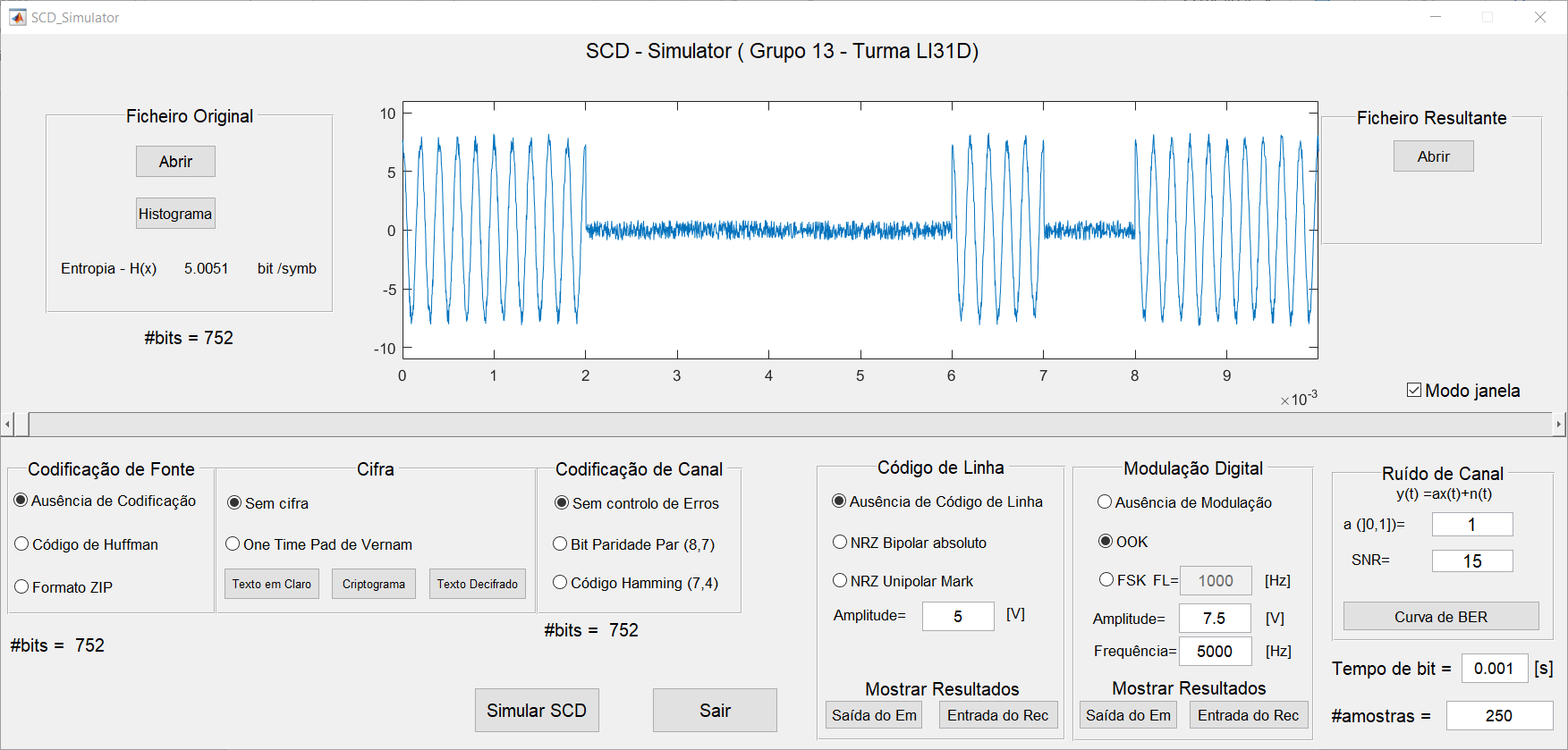


Figura - Receção de dados com modulação OOK

Na Figura 11 é possível verificar a emissão de dados com modulação OOK com uma amplitude de 7,5 a -7,5 Volts, com um tempo de bit de 1 milissegundo e uma frequência de 5KHz. Na Figura 12 é possível verificar a receção dos mesmos dados, mas com uma adição de ruído na amplitude, no entanto, é visivelmente possível perceber os bits a 0 e a 1. Na Figura 13 está representado o ficheiro resultante da transmissão.

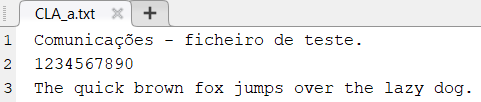


Figura - Ficheiro resultante de uma transmissão com modulação OOK

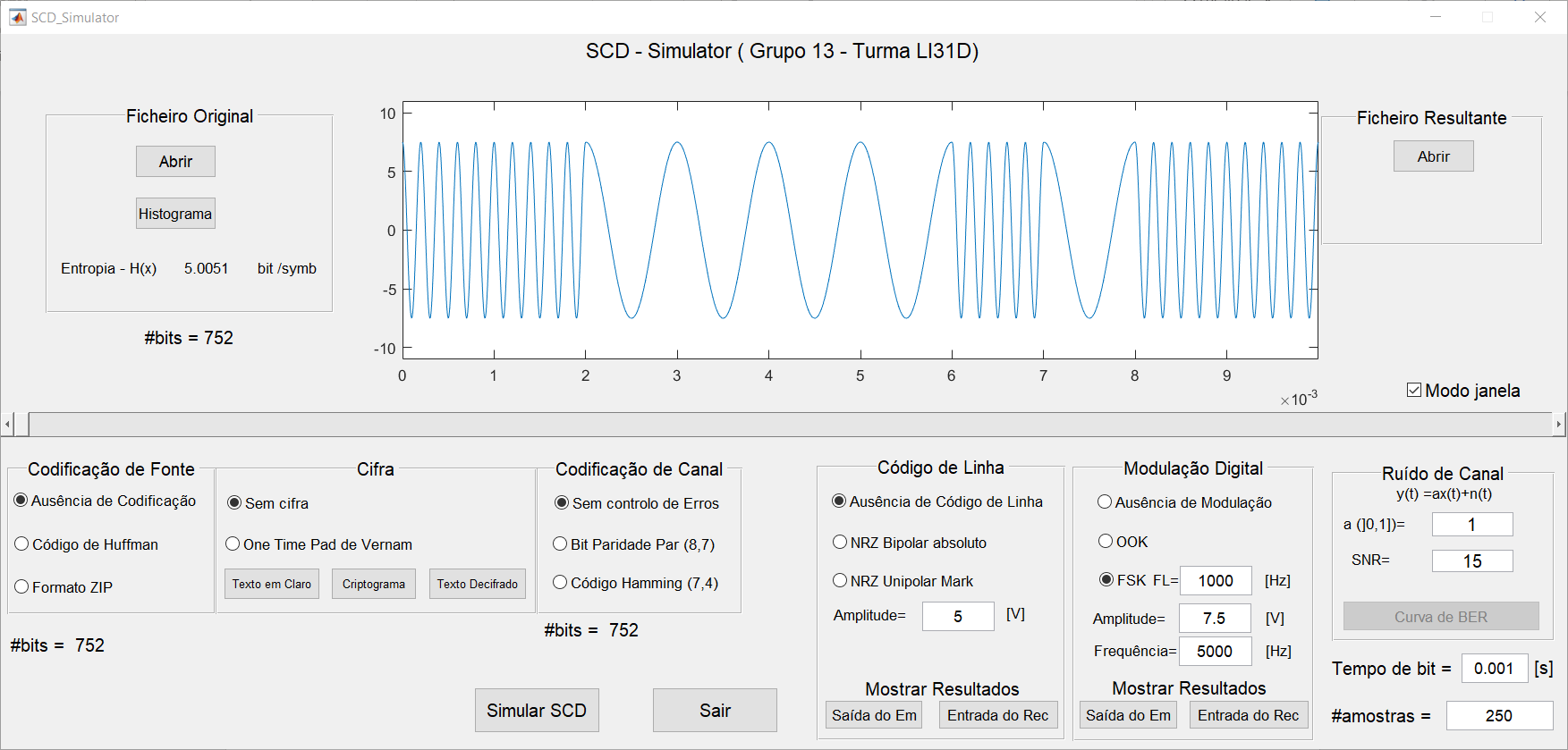


Figura - Emissão de dados com modulação FSK

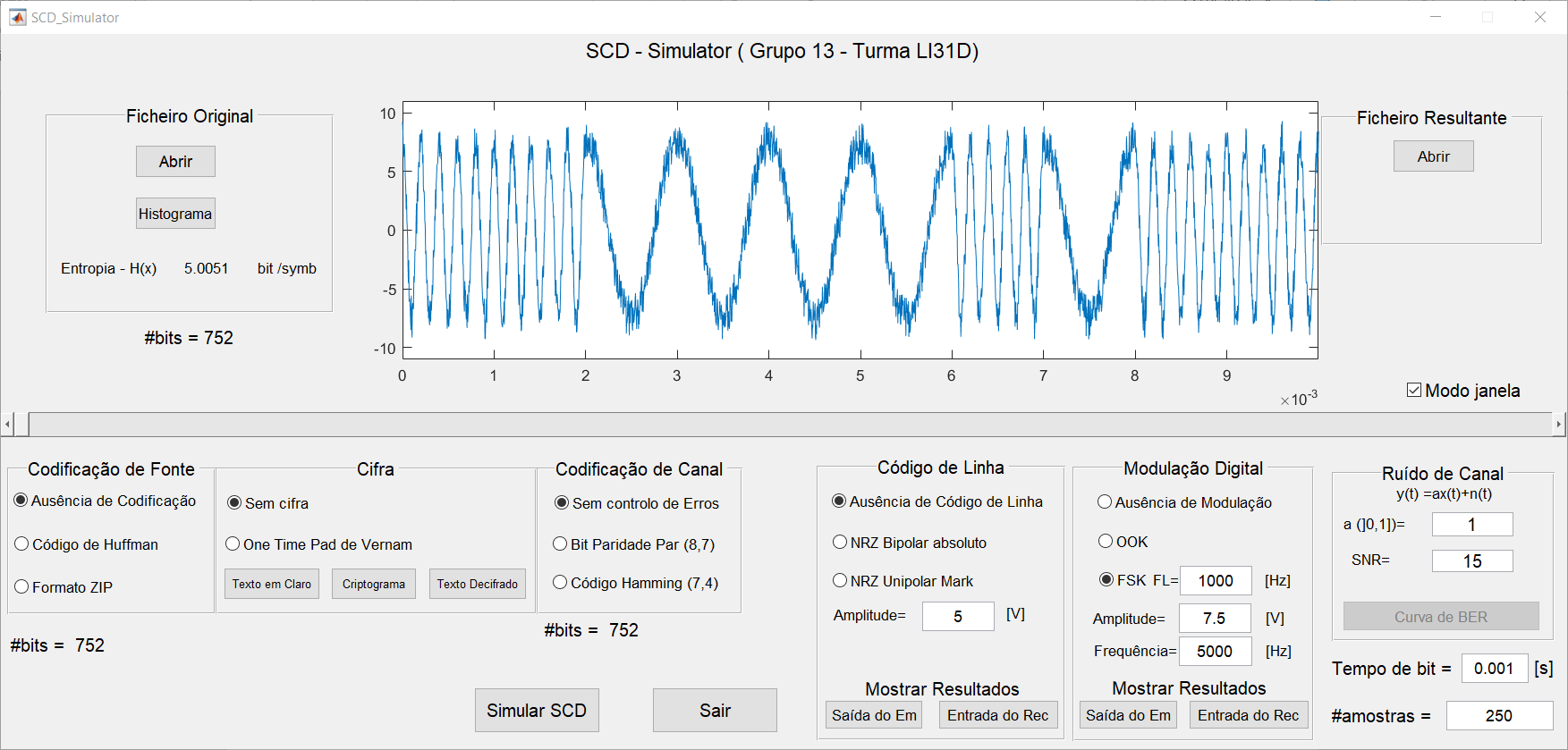


Figura - Receção de dados com modulação FSK

Na Figura 14 é possível verificar a emissão de dados com modulação OOK com uma amplitude de 7,5 a -7,5 Volts, com um tempo de bit de 1 milissegundo e uma frequência de 5KHz com bits a 1 e 1KHz com bits a 0. Na Figura 15 é possível verificar a receção dos mesmos dados, mas com uma adição de ruído na amplitude, no entanto, é visivelmente possível perceber os bits a 0 e a 1. Na Figura 16 está representado o ficheiro resultante da transmissão.

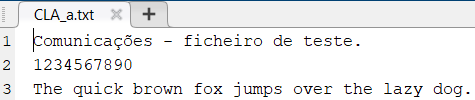


Figura - Ficheiro resultante de uma transmissão com modulação FSK

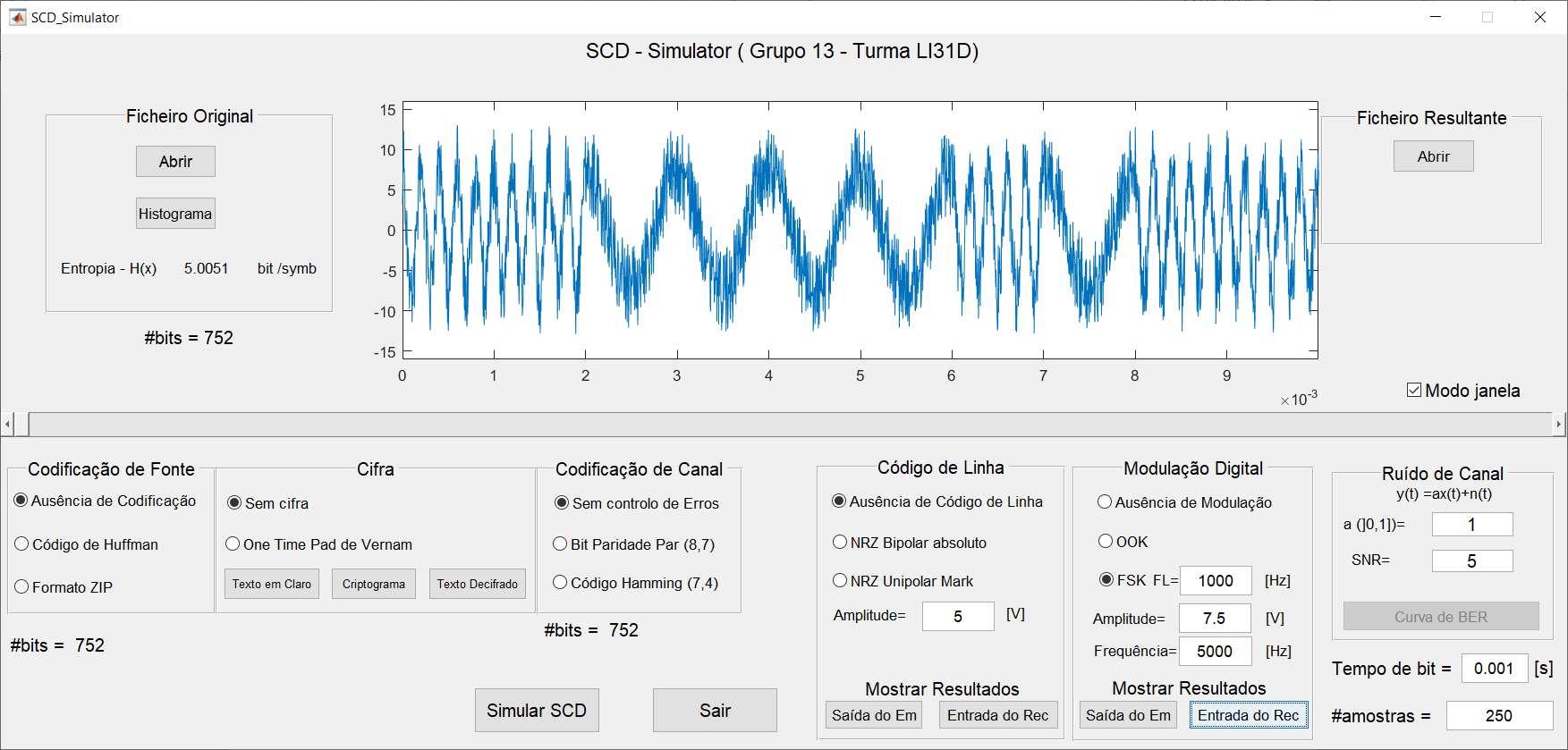


Figura - Receção de dados com modulação FSK com valor de SNR alterado

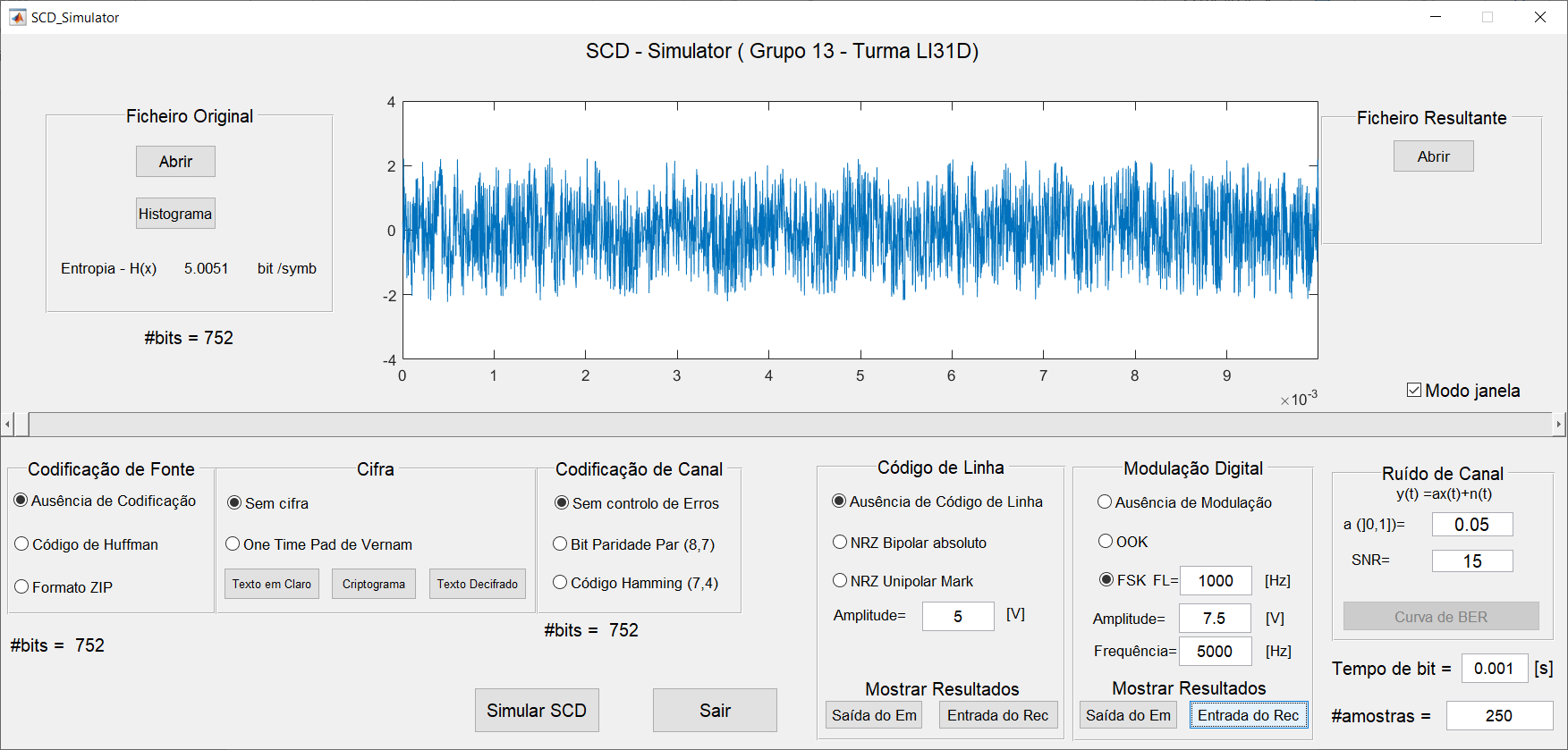


Figura - Receção de dados com modulação FSK com valor de atenuação do canal alterado

As Figuras 17 e 18 representação a funcionalidade dos parâmetros do ruído de canal com o mesmo sinal usado da experiência das Figuras 15 e 16. Na Figura 17, é possível verificar que a amplitude do ruído adicionada é maior, baixando a relação de sinal-ruído, no entanto ainda é possível distinguir os bits 0 e 1. Na Figura 18, o valor da atenuação foi alterado para um valor muito baixo que é impossível distinguir os bits 0 dos bits 1.

## 2.3 Alínea C

### 2.3.1 Transmissão de um ficheiro de texto

A primeira experiência da simulação é uma transmissão de ficheiro de texto é aplicada ao ficheiro “a.txt”. Este vai sofrer uma codificação com código de Huffman, com a cifra One Time Pad de Vernam e com o código de Hamming (7, 4). Como explicado no trabalho anterior, cada uma destas funcionalidades lê de um ficheiro e escreve num novo, ou seja, na função de aplicar o código de linha, o nome do ficheiro a passar é diferente do que no início da codificação. Neste caso, o ficheiro é “H74C\_OTPC\_HufC\_a.txt”. A emissão dos dados deste ficheiro está representada, uma parte, na Figura 19. Na Figura 20, é demonstrado o mesmo sinal no momento da receção após passar por um canal ruidoso. É possível verificar que a olho é muito difícil distinguir os bits 0 dos bits 1.

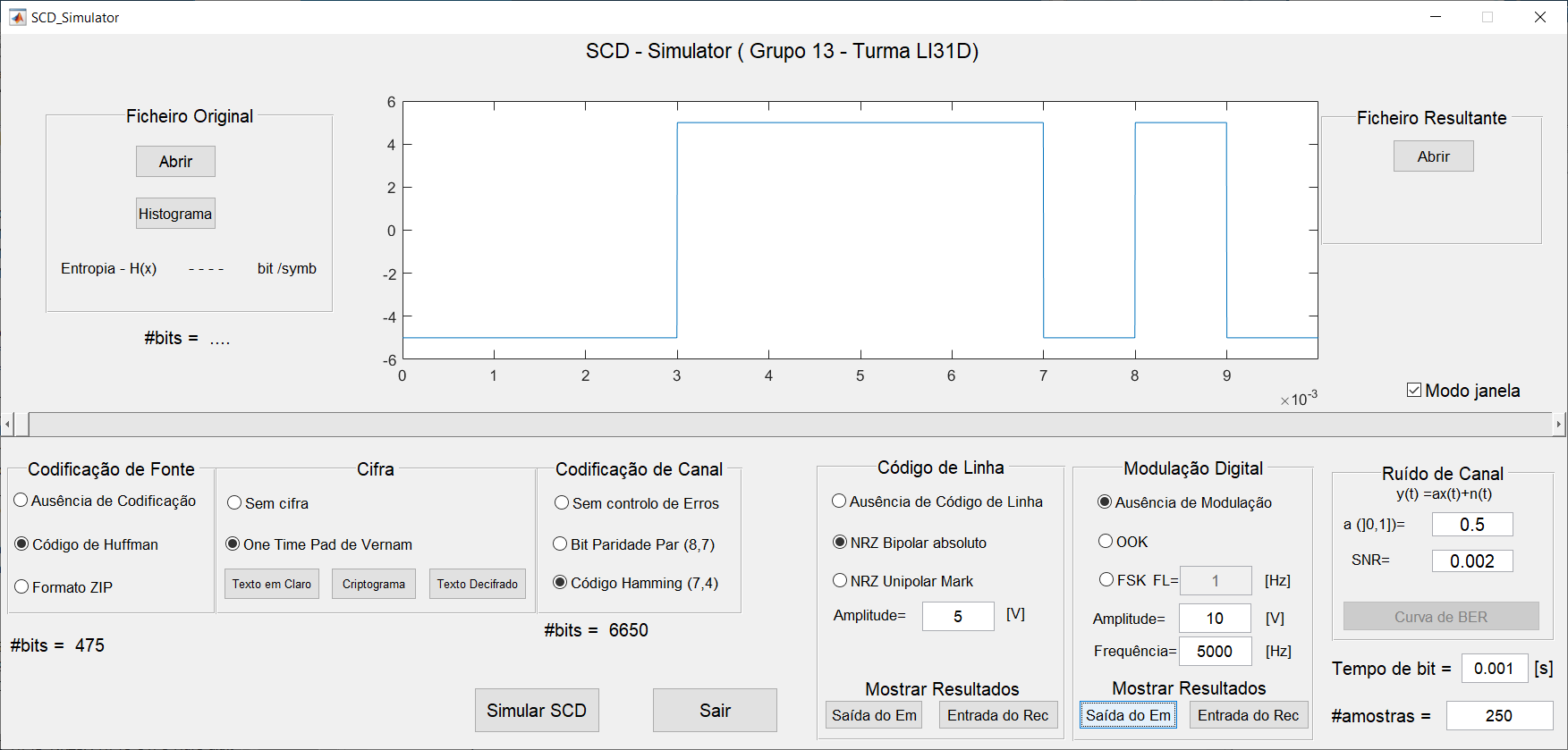


Figura - Sinal de emissão do ficheiro de texto "a.txt" codificado

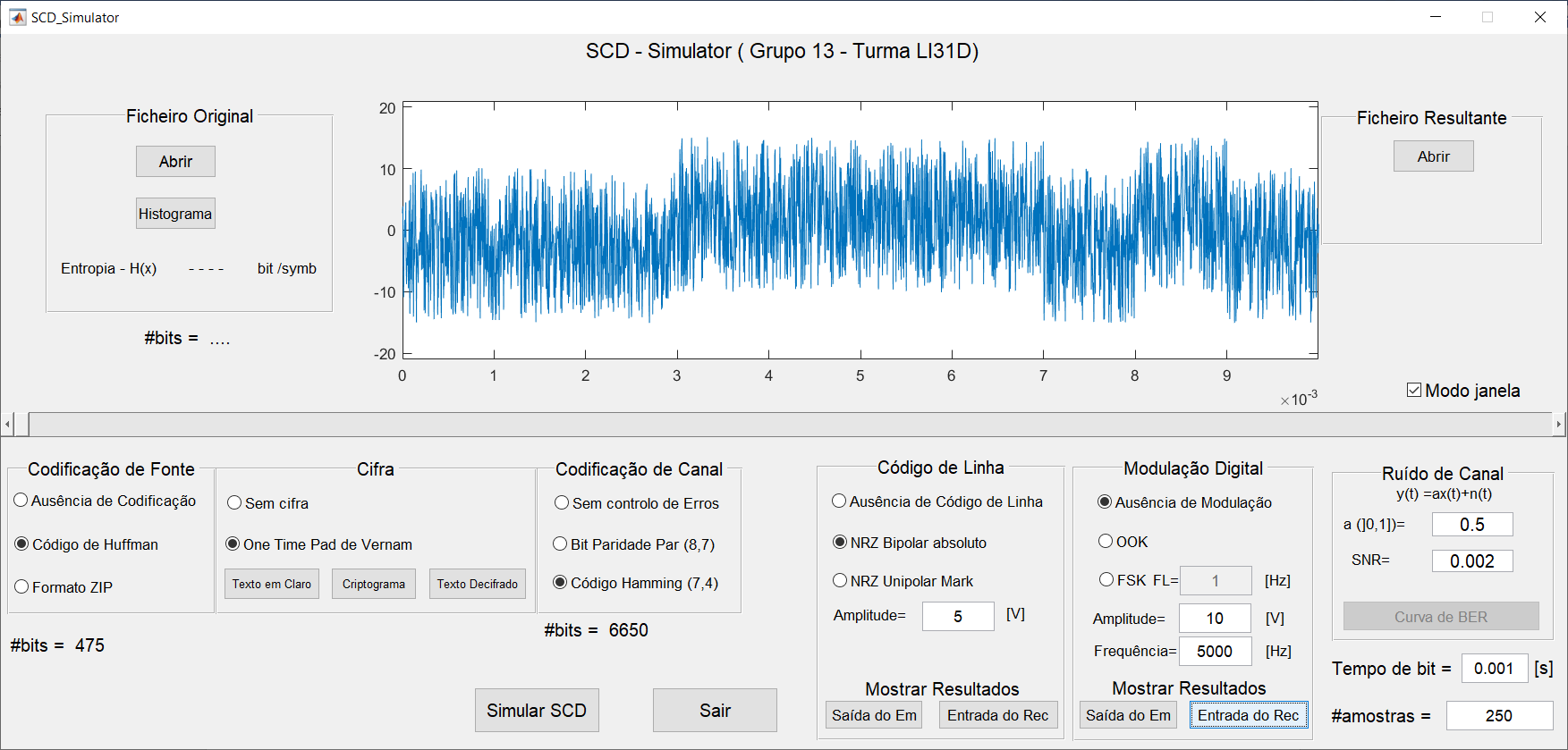


Figura - Sinal de receção do ficheiro de texto "a.txt" codificado

Foi verificado e comparado o ficheiro original com o ficheiro transmitido, e conclui-se que o ficheiro foi transmitido com sucesso e sem erros. Na Figura 21 está representada o conteúdo do ficheiro original e na Figura 22 o ficheiro transmitido e como se pode verificar, o conteúdo é o mesmo.

|  |  |
| --- | --- |
| Figura 21 - Conteúdo do ficheiro original | Figura 22 - Conteúdo do ficheiro transmitido |

### 2.3.1 Transmissão de um ficheiro de imagem

A segunda experiência da simulação é uma transmissão de ficheiro de imagem é aplicada ao ficheiro “bird.gif”, pois esta imagem não ocupa muito espaço. Este vai sofrer uma codificação com código de Huffman, com a cifra One Time Pad de Vernam e com o código de Hamming (7, 4). A emissão dos dados deste ficheiro está representada, uma parte, na Figura 23 e em comparação com a experiência anterior, o número de amostras teve que ser menor, para ocupar menos espaço na memória. Na Figura 24, é demonstrado o mesmo sinal no momento da receção após passar por um canal ruidoso. É possível verificar que a olho é muito difícil distinguir os bits 0 dos bits 1.

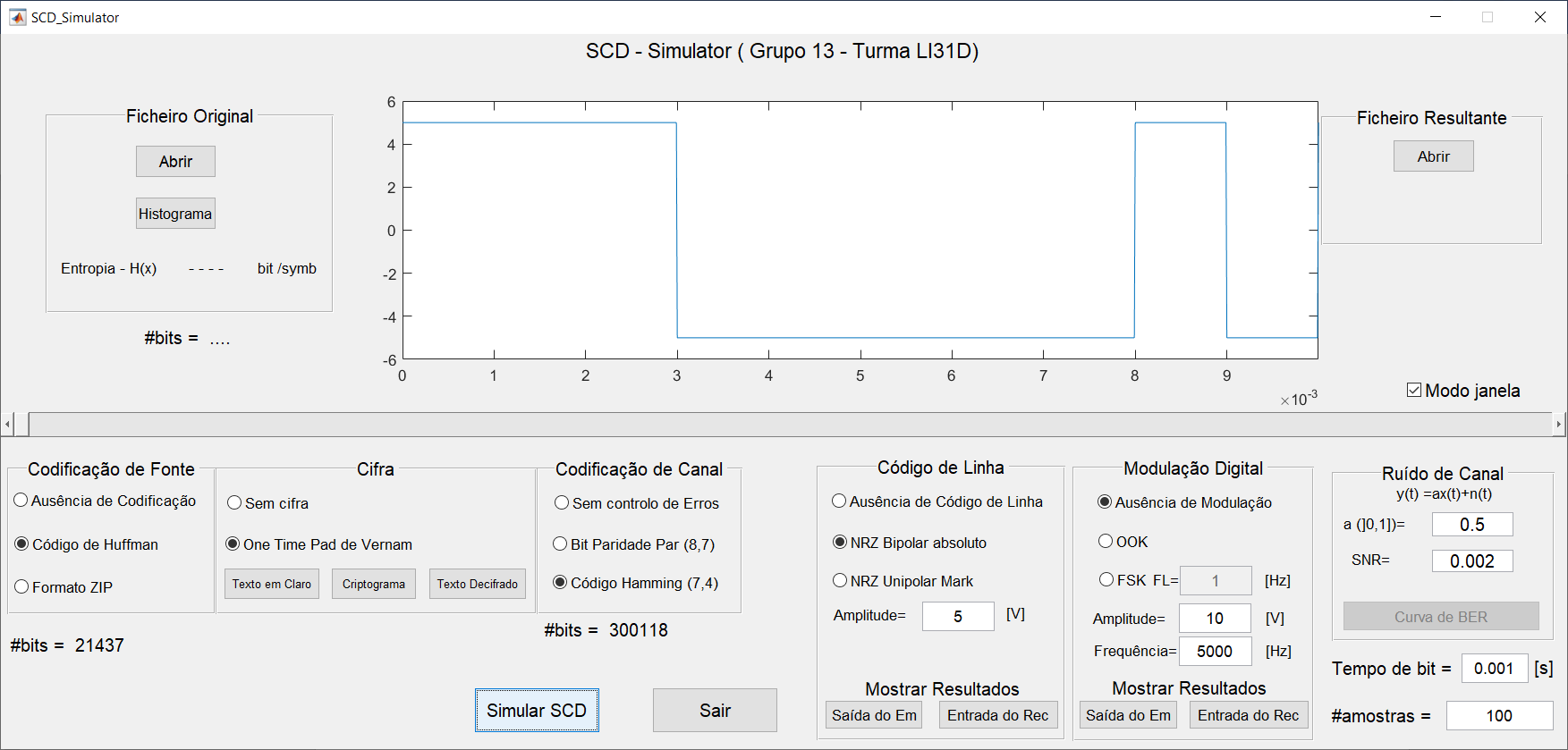


Figura - Sinal de emissão do ficheiro de texto "bird.gif" codificado

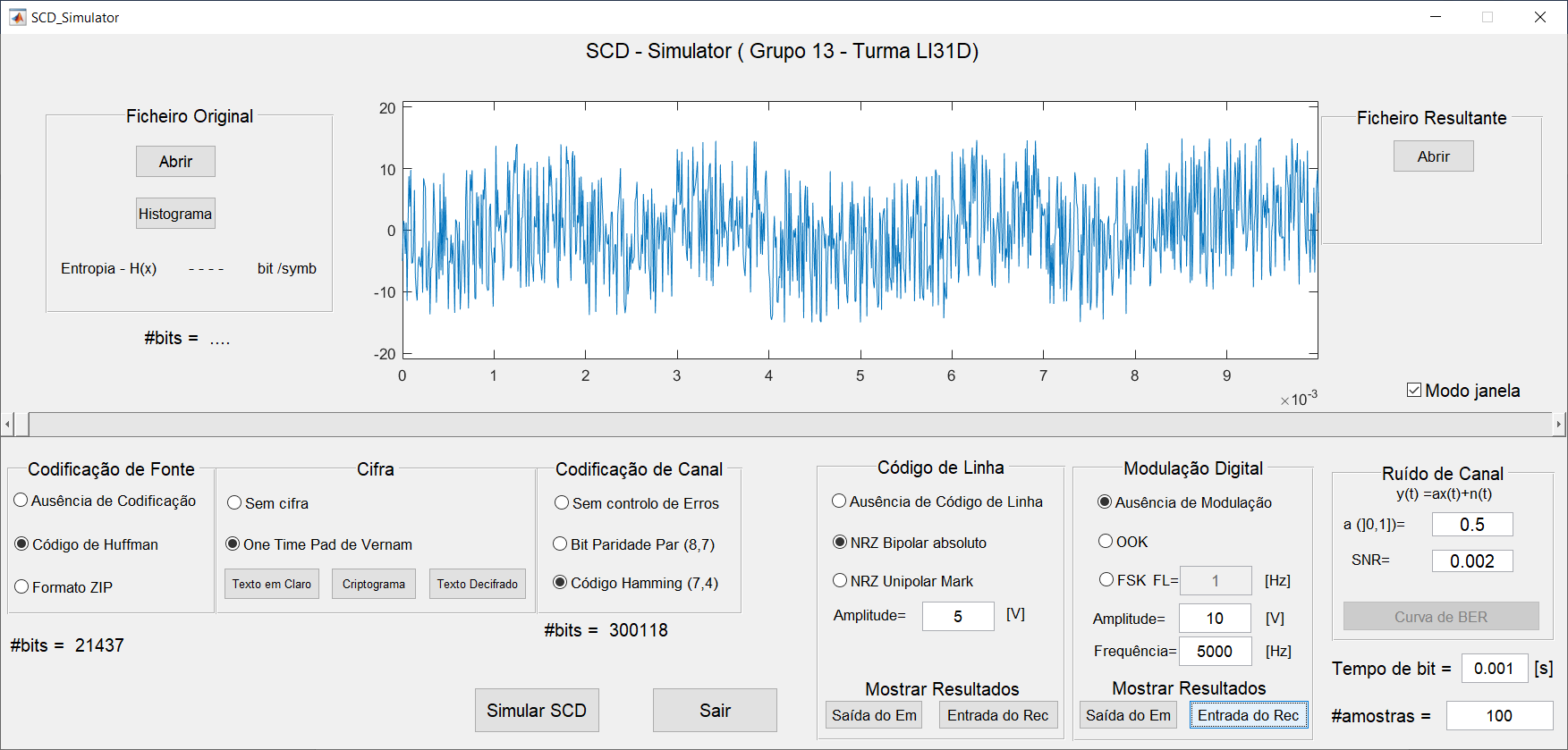


Figura - Sinal de receção do ficheiro de texto "bird.gif" codificado

Como este ficheiro é maior que o ficheiro de texto, implica que seja necessário um número maior de bits a enviar e também um maior tempo de transmissão. Com isto, a probabilidade de haver 1 bit errado durante a transmissão é maior e, por acaso, nesta transmissão houve um erro que influenciou na tentativa de descodificação de Huffman encontrou um símbolo que não constava no dicionário que por consequência gerou um erro, como pode ser verificado na Figura 25, não sendo possível gerar um ficheiro resultante.

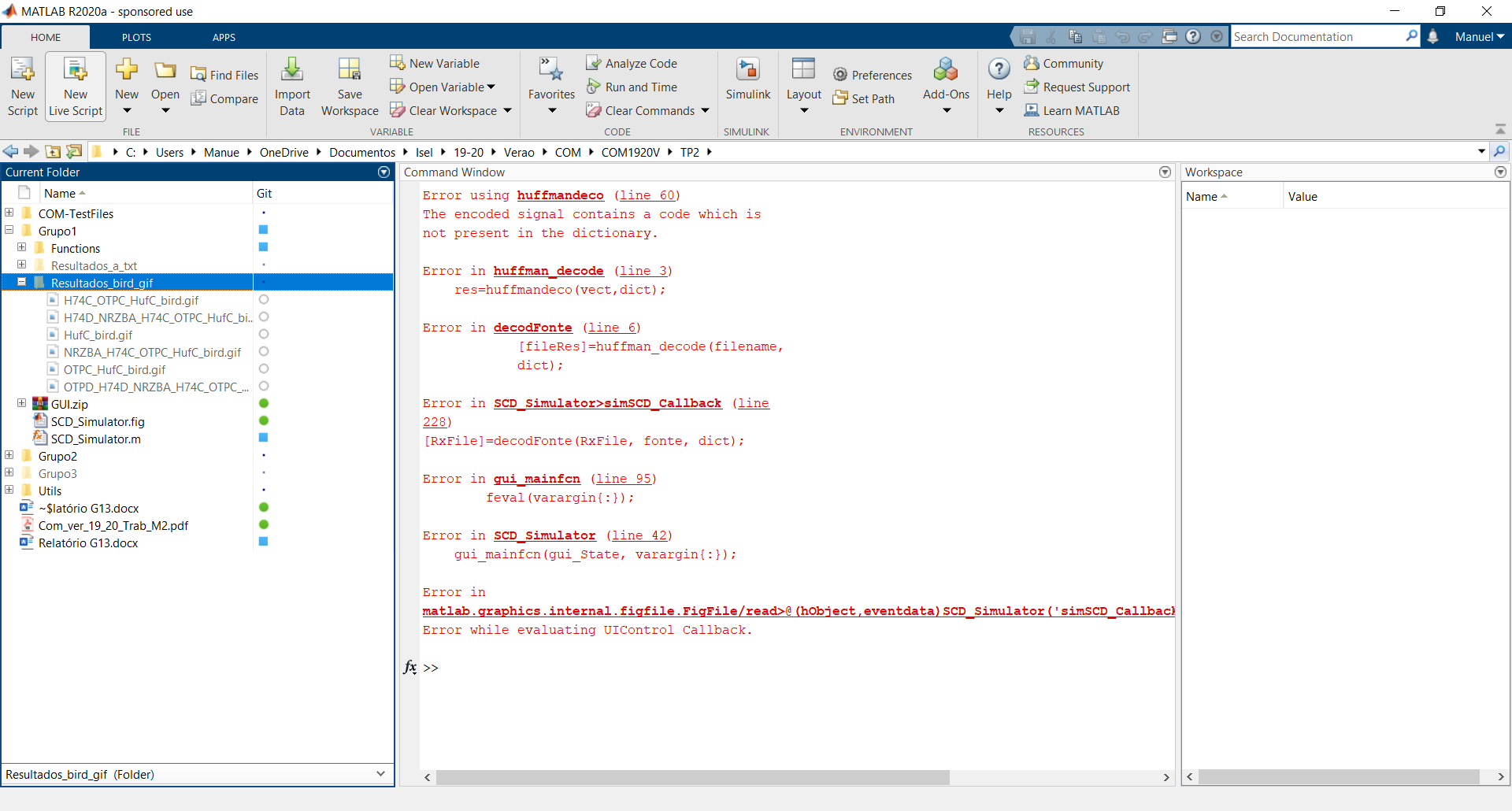


Figura - Resultado da tentativa de descodificar o sinal recebido

# 3. Grupo 2

# 4. Grupo 3

Neste grupo é pretendido fazer funções que contenham algoritmos de verificação de erros, contextualizados na validação do dígito de controlo do Bilhete de Identidade e do ISBN. Apesar de se poder usar qualquer linguagem de programação, foi utilizado à mesma o MATLAB.

## 4.1 Alínea (i)

Para fazer o algoritmo de cálculo do dígito de controlo, foi criada a função *BI\_generate\_control\_digit*. Esta recebe como parâmetro de entrada uma string com 8 caracteres, sendo estes todos números. Para obter o dígito de controlo, foi utilizada a fórmula , onde x é calculado a partir de um ciclo e o t foi calculado usando esta fórmula .

Para fazer o algoritmo de verificação, foi criada a função *BI\_check\_digit\_control*, que tem a capacidade de verificar se o dígito de controlo está correto e se todos os algarismos do BI estão bem e na ordem correta. Esta função recebe uma string com 9 caracteres, sendo estes todos números. Para fazer a verificação é usado o algoritmo do enunciado, recorrendo a uma instrução cíclica e no final é verificado se o resto da divisão do resultado com 11 é 0. Se for 0, então o número do BI é válido, caso contrário é inválido. A Figura 26 demonstra a funcionalidade das duas funções através de um script de teste.

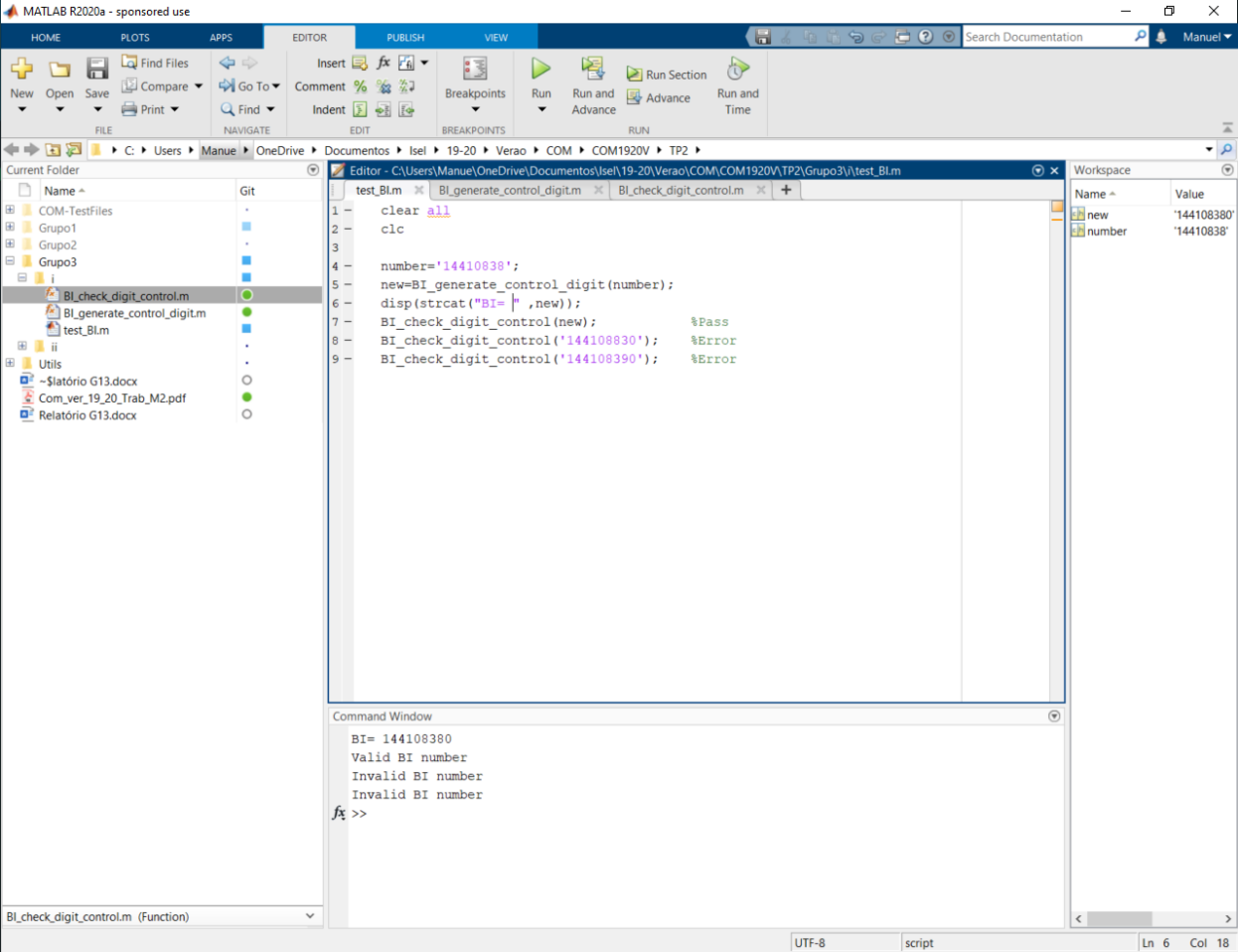


Figura - Resultados do script de testes das funções de BI

## 4.2 Alínea (ii)

Para fazer o algoritmo de cálculo do dígito de controlo, foi criada a função *ISBN\_generate\_control\_digit*. Esta recebe como parâmetro de entrada uma string com 12 caracteres, sendo estes todos números. Para obter o dígito de controlo, foi utilizada a fórmula , onde x é calculado a partir de um ciclo onde é verificado a paridade de cada algarismo e se o algarismo for par, soma-se o seu valor ao resultado, caso contrário é o triplo.

Para fazer o algoritmo de verificação, foi criada a função *ISBN\_check\_digit\_control*, que tem a capacidade de verificar se o dígito de controlo está correto e se todos os algarismos do ISBN estão bem e na ordem correta. Esta função recebe uma string com 13 caracteres, sendo estes todos números. Para fazer a verificação é usado o mesmo algoritmo que a função de geração do dígito de controlo, recorrendo à instrução cíclica. Depois é utilizada a fórmula , onde r é o resultado da instrução cíclica. Se o valor de d for diferente que 0, quer dizer que o valor de ISBN é inválido, caso contrário então é válido. A Figura 27 demonstra um script de teste onde é gerado o dígito de controlo e são feitas várias verificações a diferentes ISBN, que incluem um ISBN correto, uma troca de ordem entre 2 dígitos e um dígito errado.

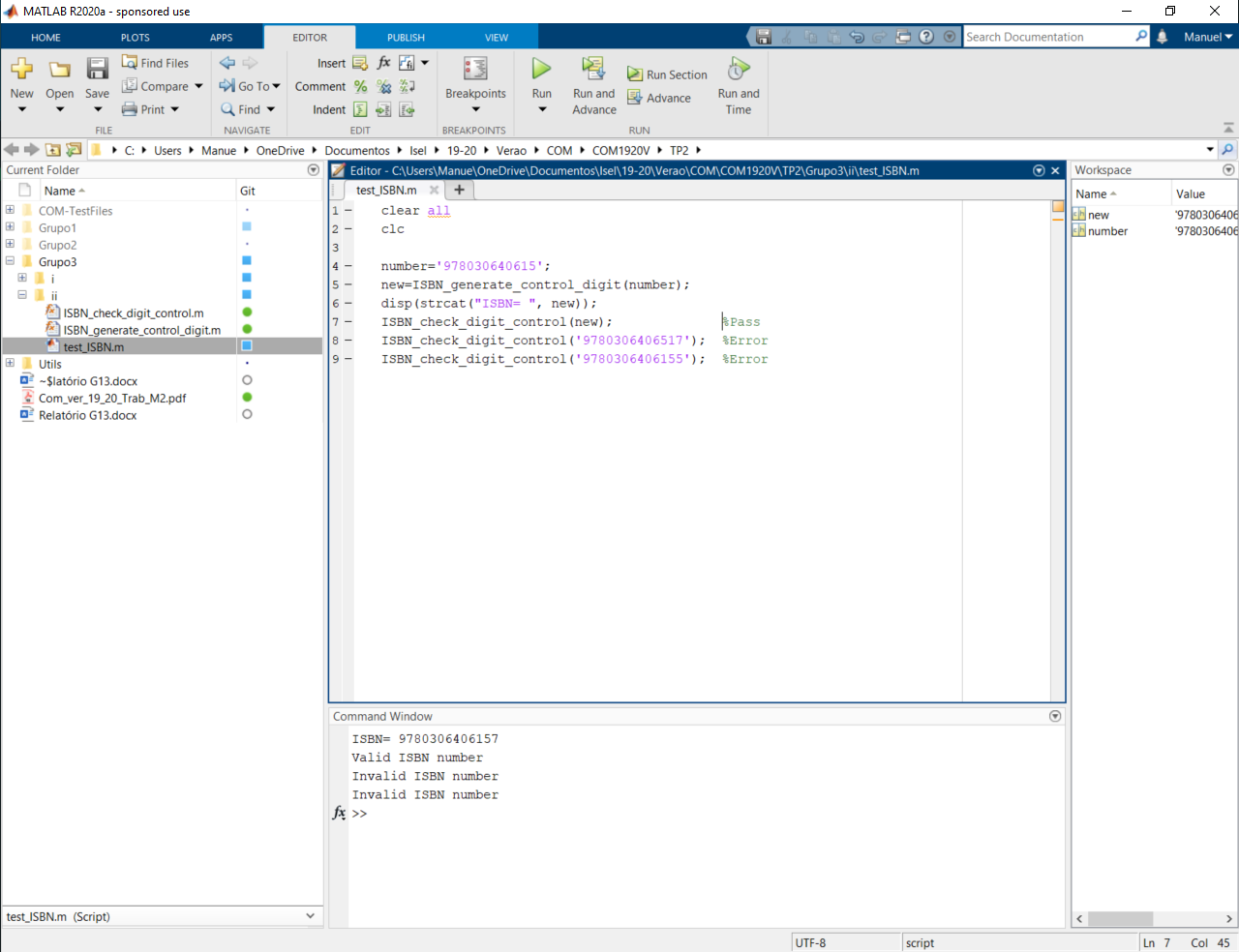


Figura - Resultados do script de testes das funções de ISBN