Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

SITE – Simulador TETRA

Joaz Soares Castro Junior
Matricula: 99/17128

Professor Orientador: Plínio Ricardo Ganime Alves

Junho de 2005

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

SITE – Simulador TETRA

Joaz Soares Castro Junior
Matricula: 99/17128

Professor Orientador: Plínio Ricardo Ganime Alves

Junho de 2005

Agradecimentos

Agradeço a Deus por ter me dado força e perseverança para alcançar meus objetivos.

Agradeço a meus pais que sempre me apoiarem nos momentos de fraqueza. Que sempre me deram a assistência necessária para tornar esse sonho realidade.

Agradeço a Vanessa Martins Farias por ter tido um papel fundamental no meu ingresso a Universidade de Brasília.

Agradeço a minha amada Rayla Guimarães Jacundá, que sempre me apoiou, me incentivou e me trouxe muita alegria.

Um agradecimento especial a meus amigos do LABTECC, Aely Lima, Ewerton Pacheco e João Laurindo por sua amizade e apoio durante todo o processo de elaboração deste trabalho.

Agradeço a todos meus amigos que me ajudaram nessa caminhada.

Agradeço também a meu professor orientador Plínio Ganime que me proporcionou a oportunidade de ingressar na área da comunicação crítica

Abstract

O padrão aberto de comunicação crítica TETRA (TErrestrial Trunked Radio) surgiu na Europa por volta de 1990 e é um padrão de rádio móvel digital criado para suprir as necessidades dos usuários profissionais e de serviços de emergência.

O TETRA é atualmente padronizado pelo ETSI (*European Telecommunications Standard Institute) e* surgiu seguindo a tendência da tecnologia GSM.

Em estudos feitos no LABTECC (LABoratório de TEcnologia em Comunicação Crítica) da Universidade de Brasília, foi criado uma plataforma, denominada SIMUTECC, onde foi comparado dois padrões abertos: TETRA e PROJETO25.

O SITE (SIMulador TETRA) é uma interface gráfica que compõem parte do SIMUTECC. Essa interface foi criada para simular alguns parâmetros relevantes com vistas a sua adoação como padrão de comunicação critica para o Brasil.

I. Abreviaturas / Siglas

ACELP - Algebraic Code Excited Linear Predictive

BER - Bit Error Rate

CRC - Cyclic Redundancy Check

DQPSK - Differential Quaternary Phase Shift Keying

FEC – Forward Error Correction

GSM - Global System for Mobile

HT – Tipicamente Montanhoso

MCC (Mobile Country Code),

MNC (Mobile Network Code).

RCPC - RAte Compatible Punctured Convolutional Codes

SITE - Simulador TETRA

T U – Tipicamente Urbano

TDMA – Time Division Multiple Access

TETRA – Terrestrial Trunked Radio

II. Introdução

Tecnologias de comunicações críticas são sistemas diferenciados capazes de prover a comunicação mesmo em situações caóticas (desabamentos, enchentes, atentados terroristas...), ou seja, possuem a devida robustez para que a comunicação nestas condições continue a ocorrer, mesmo que as demais redes como, por exemplo, a telefonia fixa e celular estejam inoperantes.

Um sistema para comunicação em segurança pública (sistema de comunicação crítica) troncalizado e digital inclui elementos como central de despacho, sistema de controle, estação base de RF e equipamento de controle, unidades veiculares e portáteis, sistema de sinalização e equipamento de interface que, combinados para prover um sistema de rádio móvel, permitem comunicação bidirecional entre as centrais de despacho através das estações base e uma ou mais unidades veiculares ou portáteis.

Interoperabilidade entre órgãos, serviços de dados com a menor ocupação possível dos canais, comunicação segura e de qualidade, rapidez no atendimento à população nas centrais de despacho e quantidade de canais que permitam a realização da comunicação pelos agentes mesmo durante os períodos de maior tráfego são algumas aplicações em segurança pública que justificam a utilização de um sistema de comunicação crítica.

Tendo essas aplicações em vista, a taxa de transmissão, a área de cobertura, o grau de segurança, a capacidade de evolução tecnológica, o acesso imediato ao canal de voz e dados, compatibilidade com o legado caso a transição para o novo sistema seja de forma gradual são fatores que devem ser levados em consideração para a decisão de qual padrão adotar. O objetivo do SITE é apresentar os resultados de estudos de algumas destas variáveis como BER e *fading*.

Para o melhor entendimento deste trabalho, alguns conceitos básicos serão apresentados abaixo.

• **Bit**: algarismo do sistema binário que só pode assumir os valores 0 e 1. Em comunicações digitais, combinações de bits são usadas para representar sons de voz.

• **TETRA**: Trata-se de um padrão aberto para rádio troncalizado que opera em TDMA

• **Comunicação Crítica**: Também conhecida como comunicação emergencial, esse tipo de comunicação é usada pelos órgãos responsáveis pela segurança em geral, bombeiros e por grandes empresas que necessitam de uma comunicação eficaz e segura.

• **Criptografia**: Algoritmo utilizado para codificar voz ou dados com intuito de impedir ou pelo menos dificultar que pessoas não autorizadas tenham acesso aos dados da transmissão.

• **Erros na transmissão**: Os erros na transmissão de bits, ocorrem quando há inversão de bits. Por exemplo, devido a algum motivo o bit 1 passa a ser igual a 0 ou o bit 0 passa a ser 1.

• **Bits de voz**: São bits utilizados para representar a voz de um emissor.

• **Bits de Paridade**: Os bits de paridade indicam se a quantidade de 1's de determinado grupo de bits é par ou ímpar. Se a quantidade é par, o bit de paridade é igual a 0. Caso contrário, esse bit assume o valor 1. No texto, para determinar os bits de paridade, usou-se a operação "ou exclusivo", que tem a tabela verdade dada na Tabela 3. Por exemplo, se tivermos o seguinte grupo de bits {0; 0; 1; 0; 1; 1; 0; 1}, fazendo a operação ou exclusivo temos: 0+0+1+0+1+1+0+1 = 0. Caso o grupo seja { 0; 1; 1; 1} temos: 0 + 1 + 1 + 1 = 1.

• **Tail bits**: Os *tail bits* têm a função de inicializar o código convolucional.

• **Frame**: Um *frame* é um conjunto de bits. No caso do padrão TETRA, um *frame* possui 137 bits.

III. Apresentação visual do Simulador

O simulador SITE foi desenvolvido na linguagem orientada a objeto Borland Delphi 5. Essa linguagem foi utilizada por apresentar componentes gráficos de fácil utilização, grande agilidade para trabalhar com formulários, grande poder de cálculo e principalmente por apresentar uma relação de necessidade de recursos computacionais Vs. Aplicações possíveis bem interessante.

A tela inicial do simulador é mostrada abaixo:

Figura 1: Tela inicial

Uma característica adotada no programa é o botão “Saiba Mais” onde o usuário poderá obter informações adicionais sobre o tópico em avaliação.

IV. Codificação de voz no padrão TETRA

Em um processo de comunicação crítica (também conhecido como comunicação emergencial), é extremamente necessário que a informação transmitida pelo emissor seja recuperada integralmente pelo receptor, com a menor taxa de erros e com o maior sigilo possível. Os erros na transmissão ocorrem por diversos motivos, dentre eles, devido a distorção de fase, interferência eletromagnética e ruídos térmicos e impulsivos. Esses erros podem ser corrigidos com o uso de um algoritmo de codificação específico. No caso do padrão TETRA, tal algoritmo também promove a criptografia da mensagem, tornando-a sigilosa. O código em questão atua sobre os bits de voz, adicionando bits de redundância (como bits de paridade e *tail* bits) e promovendo operações matemáticas complexas que possibilitam a recuperação do sinal na decodificação da mensagem e a segurança da mesma [1].

O codificador de fonte do padrão TETRA, é o ACELP, cuja taxa de codificação é de 4,57 kbps,e produz a cada 30ms um frame 137 bits. Os bits resultantes desse codificador passam pelo processo de controle de erros. De acordo com o padrão há dois modos possíveis para codificação. O primeiro é quando dois frames são codificados juntos (caso utilizado no simulador)[2]. O segundo ocorre quando é necessário enviar sinais de controle no canal de tráfego (frame stealing). Neste caso, cada frame é codificado separadamente.

O algoritmo para controle de erros e criptografia é resumido abaixo:

1. Interfaces na estrutura de controle de erros e criptografia:

A definição de interfaces na estrutura de controle de erros e criptografia é dada na Figura 2.

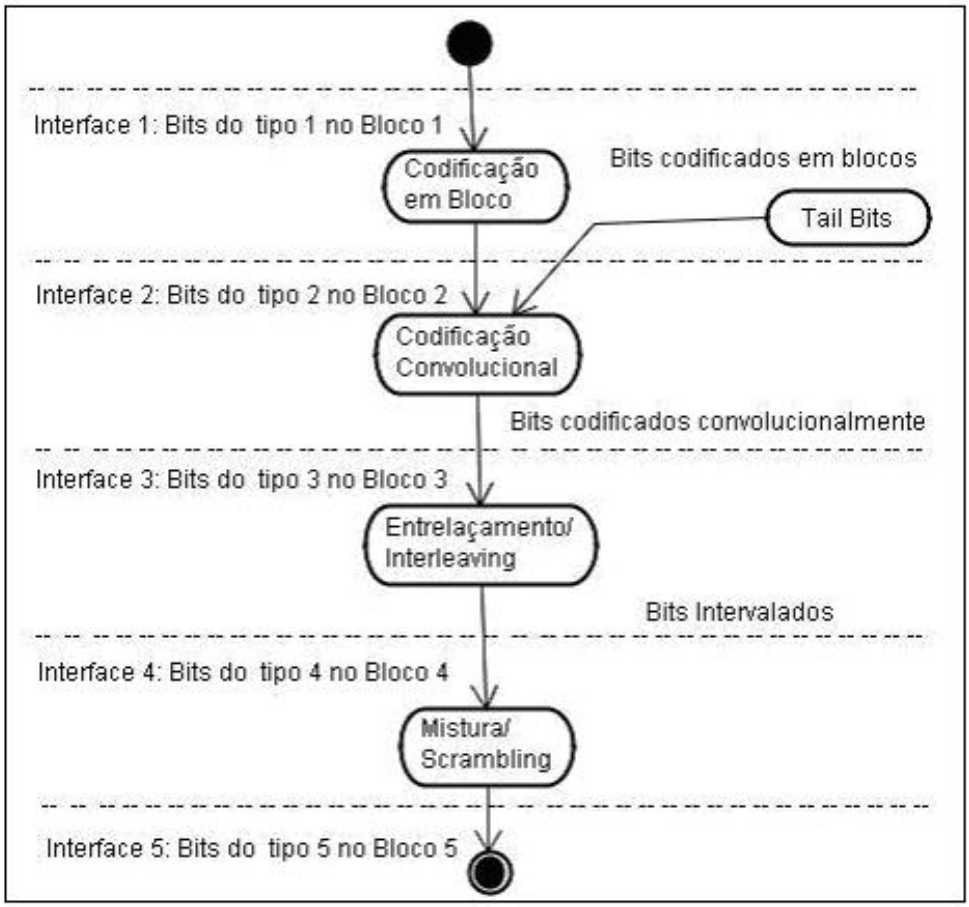


Figura 2: Definição das interfaces na estrutura de controle de erros e criptografia

A janela correspondente em nossa interface gráfica é mostrada na **Figura 3** onde temos a simulação da geração de dois *frames* de voz.

O código, de forma resumida, utilizado para efetuar tal ação também está mostrado abaixo.

1.1 Exemplo de codificação em Delphi para geração dos frames

```
procedure Tfrmtetra.btngerarClick(Sender: TObject);
var i,j: integer;
m1,m2,m3,m4,m5,m6,m7,m8: integer;

//------------------------------GERANDO FRAME 1 DE 137 BITS (MATRIZ A)----------------
begin
      edtframe1.Color:=clWhite;
      btncodificar.Enabled:=true;
      btnlimpar.Enabled:= true;
      figblocos.Visible:=true;
      edtconc.Color:=clWhite;
      if edtframe1.Text='' then
      begin
            Randomize;
            for i:= 1 to 137 do
            begin
             A[i]:=strtoint(numero[random(2)]);
             edtframe1.Text := edtframe1.Text + inttostr(A[i]);
             sgA.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(A[i]);
            end;
      end
//GERANDO NOVA SEQUENCIA PARA OS FRAMES 1 E 2 (NOVAS MATRIZES A, B e M )
      else
      begin
           if MessageDlg('Deseja gerar outra sequencia?',mtConfirmation,[mbYes,mbno],0)=Mryes then
            begin
              edtframe1.Clear;
              edtframe2.Clear;
              edtconc.clear;
              figblocos.Visible:=true;
              Randomize;
              for i:= 1 to 137 do
              begin
                //para frame 1
```

```
    A[i]:=strtoint(numero[random(2)]);
    edtframe1.Text := edtframe1.Text + inttostr(A[i]);
    sgA.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(A[i]);
    //para frame 2
    B[i]:=strtoint(numero[random(2)]);
    edtframe2.Text := edtframe2.Text + inttostr(B[i]);
    sgB.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(B[i]);
  end;
  //para matriz m (concatenada)  e sgRCPC (rcpc)
  for i:=1 to 274 do
  begin
    sgM.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(M[i]);
    sgRCPC.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(M[i]);   //para matriz sgRCPC
    rcpc[i]:=M[i]; //matriz para ser utilizada no rcpc (usar nas sensibilidades)
    edtconc.Text:=edtconc.Text + inttostr(M[i]); //mandando nova matriz para o edit
concatenação
  end;
    edtcrc.Clear;
    edtcrc.Color:=clMenu;
    edtrcpc1.Clear;
    edtrcpc1.Color:=clMenu;
    edtclass0.Clear;
    edtclass0.Color:=clMenu;
    edtclass1.Clear;
    edtclass1.Color:=clMenu;
    edtclass2.Clear;
    edtclass2.Color:=clMenu;
    edtrcpc2.Clear;
    edtrcpc2.Color:=clMenu;
    edtconc2.Clear;
    edtconc2.Color:=clMenu;
    edtinter.Clear;
    edtinter.Color:=clMenu;
    edtscramb.Clear;
    edtscramb.Color:=clMenu;
```

```
            btnlimpar.Enabled:=false;
        end;
    end;
//-----------------------GERANDO FRAME 2 (MATRIZ B)---------------------------------------
    edtframe2.Color:=clWhite;
    if edtframe2.Text='' then
    begin
        for i:= 1 to 137 do
        begin
            B[i]:=strtoint(numero[random(2)]);
            edtframe2.Text := edtframe2.Text + inttostr(B[i]);
            sgB.Cells[i-1,0]:=inttostr(i)+ ':  ' + inttostr(B[i]);
        end;
    end;
```

Note que para a geração desse código, utilizamos algumas bibliotecas e algumas variáveis. Essas são listadas abaixo como foram implementadas no algoritmo do programa.

Bibliotecas utilizadas: Windows, Messages,shellapi, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, ExtCtrls, Menus, Grids, math, ComCtrls, StdCtrls, Buttons, jpeg, Tabnotbk,cStatistics, TeeProcs, TeEngine, Chart, mxgraph, Series;

Variáveis utilizadas:

```
numero : array[0..1]of char = ('0','1');  //para gerar bit aleatorio para o frame
A,B:array[1..137] of Longint; // para matriz dos frames A e B
M: array [1..274] of integer; // matriz gerada na saida do crc
rcpc: array[1..286] of integer; // matriz resultante do CRC e entrada do RCPC 2

M2   : array ARRAY[1..432] of integer;            // matriz de saída do RCPC
M3   : array ARRAY[1..432] of integer;           // matriz de saída do Interleaving
M4   : array ARRAY[1..432] of integer;          // matriz de saída do Scrambling

msg0 : array[1..102] of integer ;     //matriz dos bits de sensibilidade 0 (provenientes de M1)
msg1 : array[1..112] of integer;      //matriz dos bits de sensibilidade 1 (provenientes de M1)
msg2 : array[1..72] of integer;       //matriz dos bits de sensibilidade 2 resultante do CRC
V1   :   array[1..336] of integer;    // matriz de saída do 16 state RCPC mother code (sensibilidade 1)
```

```
V2   : array[1..216] of integer;      //matriz de saída do 16-state RCPC mother code (sensibilidade 2)

C1   : array [1..168] of integer;     //matriz resultante do RCPC (sensibilidade 1)
C2   : array [1..162] of integer;     //matriz resultante do RCPC (sensibilidade 2)
h1   : array [1..168] of integer;     //matriz de valores intermediários usados (sensibilidade 1)
h2   : array [1..162] of integer;     //matriz de valores intermediários usados (sensibilidade 2)

P1   : array [1..3] of integer;       //puncturing coefficients do puncturing mother code de taxa 2/3
 P2   : array [1..9] of integer;      // puncturing coefficients do puncturing mother code de taxa 8/18

e    : array [1..30] of integer;      // para definição dos bits do extended colour code
C    : array [1..32] of integer;      //       idem o de cima
P    : array [-31..432] of integer;   //bits da sequência de scrambling

 j,k,i : integer;                     //números usados no processo
 g10,g11,g12,g13,g14 : integer;
 g20,g21,g22,g23,g24 : integer;
 g30,g31,g32,g33,g34 : integer;
```

2 Codificação dos frames de voz

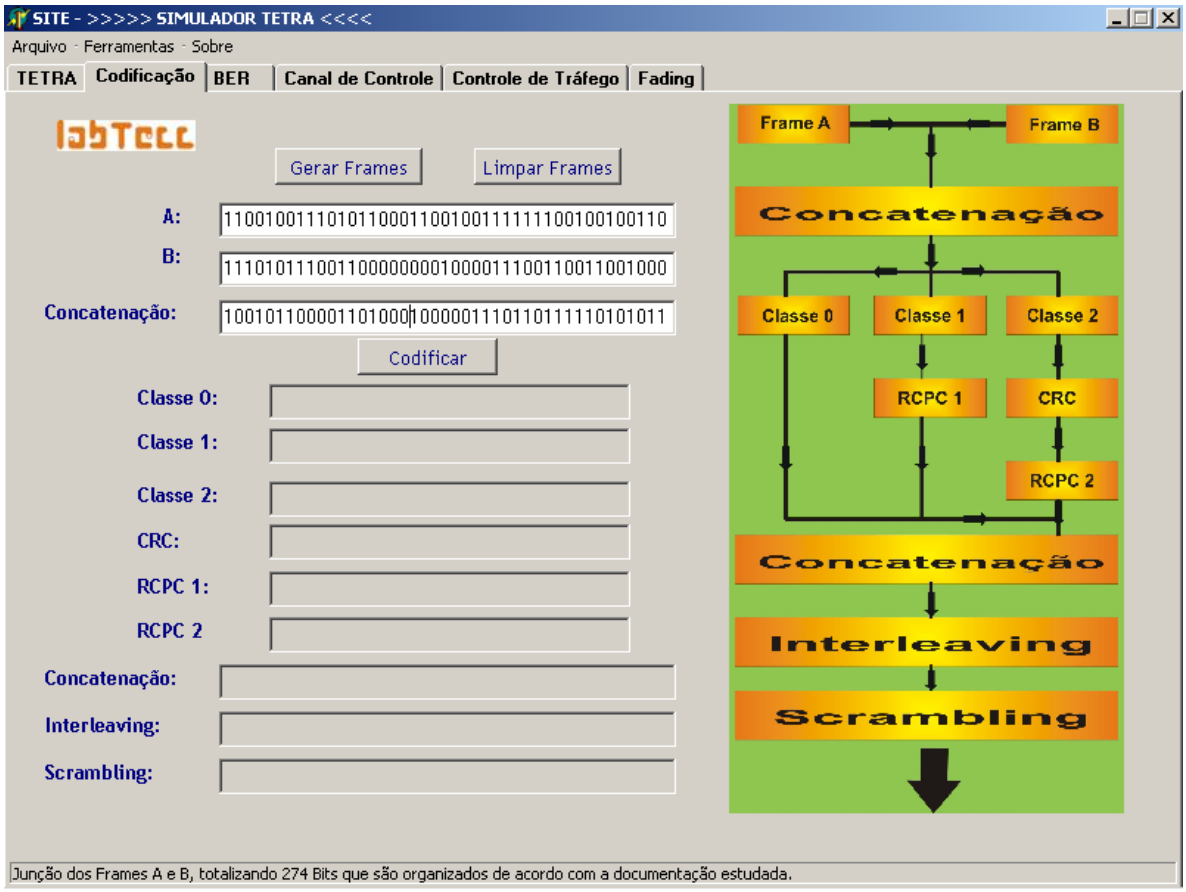


Figura 3 : Simulação da Geração do Frame de voz

Os bits de informação, que correspondem à entrada do canal codificador são referidos como bits do tipo 1, que definem a interface 1 na Figura 2. Na Figura 4 é mostrado o esquema utilizado no programa para elaboração do algoritmo para geração dos *frames* [3] e para o processo de correção de erros. Ele é bem semelhante ao mostrado na Figura 3, só que com mais detalhes. Em seguida será descrito todo o processo.

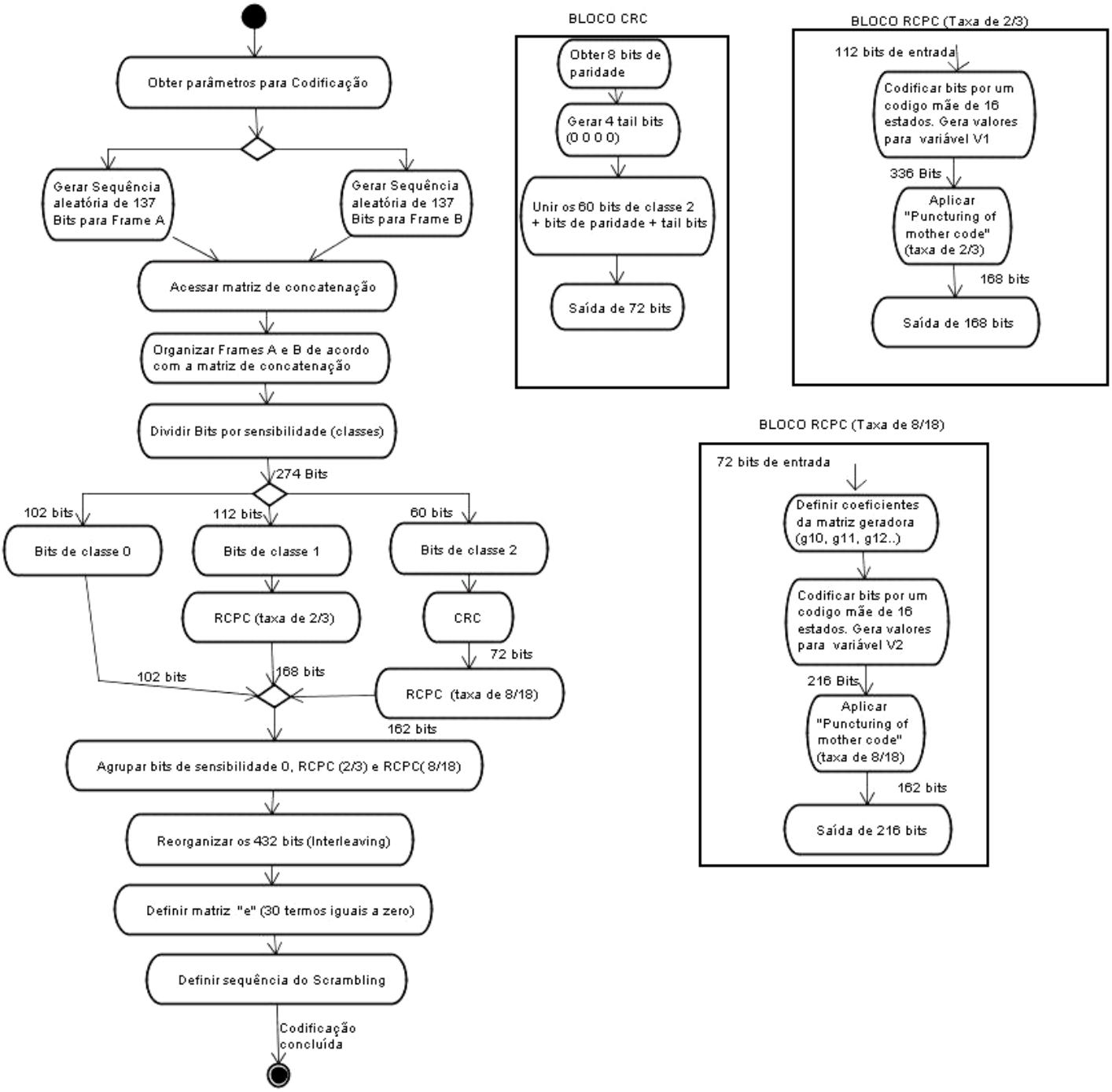


Figura 4: Diagrama de geração de frame e dos códigos corretores

- Os bits do tipo 1 serão ordenados em 3 classes, de acordo com a sua sensibilidade.

- As três classes mais os bits de paridade e os tail bits são os bits do tipo 2, e definem a interface 2 na figura 2.

- Os bits do tipo 2 serão codificados por um código convolucional, que retorna bits codificados convolucionalmente. Somente os bits das duas classes mais sensíveis serão codificados nesta etapa.

- Os bits codificados convolucionalmente junto com os bits da classe desprotegida correspondem aos bits do tipo 3, e eles definem a interface 3 na figura 2.

- Os bits do tipo 3 serão intervalados, o que define a interface 4 na figura 2.

- Os bits do tipo 4 serão misturados, e isso define a interface 5 na figura 2.

- Os bits do tipo 5 são, então mapeados em blocos multiplexados e transmitidos para o receptor.

Através da Figura 5 é possível verificar de forma mais ilustrativa o procedimento descrito acima. Note que na parte direita da Figura 5 é mostrado um diagrama de todo o processo, semelhante a Figura 2. Já do lado esquerdo tem-se a simulação dos valores.

Um outro detalhe a ser observado são os *tips*. Sempre que o usuário clica em algum campo ou passa o mouse por cima deste, mensagens são exibidas informando ao usuário como proceder para obter mais informações. Neste caso, é mostrado o número de bits por etapa.

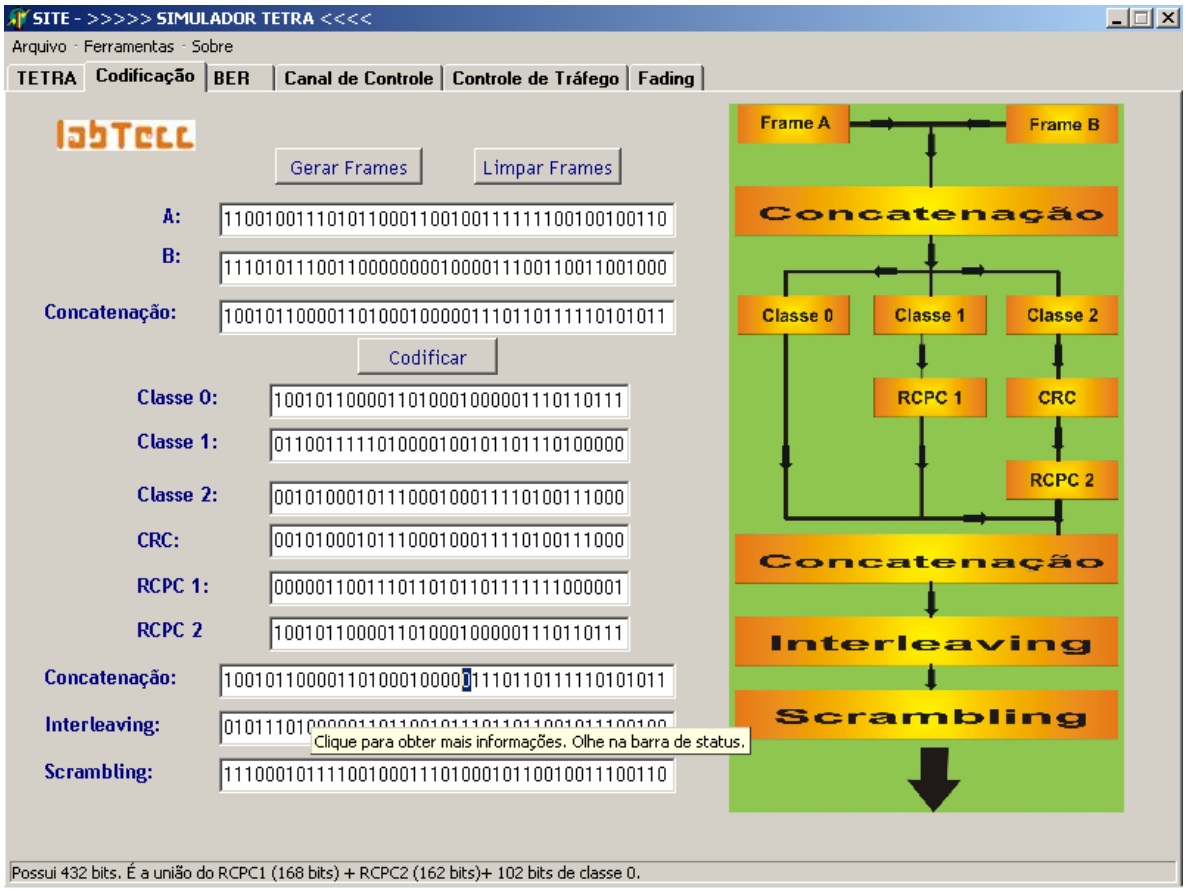


Figura 5 : Codificação dos Frames

Utilizando o diagrama da Figura 2 segue abaixo uma explicação da codificação do *Block Enconding*. Nessa interface, os bits de cada frame serão separados em três classes, de acordo

com sua sensibilidade, a qual determina a importância de cada bit na informação digital. A separação é feita de acordo com a Tabela 1, para um frame (137 bits):

Classe 0 (51 termos)	35, 36, 37, 38, 39, 40, 41, 42, 43, 47, 48, 56, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 74, 75, 83, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 101, 102, 110, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 128, 129 e 137
Classe 1 (56 termos)	58,85,112,54,81,108,135,50,77,104,131,45,72,99,126, 55,82,109,136, 5, 13, 34, 8,16, 17, 22, 23, 24, 25, 26, 6, 14, 7, 15,60, 87, 114, 46, 73, 100, 127, 44, 71, 98, 125, 33, 49, 76, 103, 130, 59, 86, 113, 57, 84 e 111
Classe 2 (30 termos)	18, 19, 20, 21, 31, 32, 53, 80, 107, 134, 1, 2, 3, 4, 9, 10, 11, 12, 27, 28, 29, 30, 52, 79, 106, 133, 51, 78, 105 e 132

Tabela 1: Separação dos bits em três classes

Uma observação a ser feita é que os números, contidos na Tabela 1, indicam as posições de cada bit no frame. As posições que eles ocupam na tabela indicam a sua importância na informação, a qual é crescente do primeiro para o último bit. A importância também é crescente de acordo com o número da classe.

No padrão TETRA, dois frames de voz são codificados juntos. Dessa forma, é necessário que os mesmos sejam misturados segundo um padrão, que é dado na Tabela 2. Para fins didáticos, adote um frame com nome A e o outro com B. Durante o texto, para facilitar as operações com os bits, será usada a linguagem matricial.

Durante a codificação em blocos, somente os bits da classe 2 serão codificados. Como temos dois frames, 60 bits serão codificados juntos. Chamemos a matriz desses bits de M. Da Tabela 2, temos:

M = [A[18] , B[18] , A[19] , B[19] , A[20] , B[20] , A[21] , B[21] , A[31] , B[31] , A[32] , B[32] , A[53], B[53] , A[80] , B[80] , A[107] , B[107] , A[134] , B[134] , A[1] , B[1] , A[2] , B[2] , A[3] , B[3] , A[4] , B[4] , A[9] , B[9] , A[10] , B[10], A[11] , B[11] , A[12] , B[12] , A[27] , B[27] , A[28] , B[28] , A[29] , B[29] , A[30] , B[30] ,A[52] , B[52], A[79] , B[79] , A[106] , B[106] , A[133] , B[133] , A[51] , B[51] , A[78] , B[78] , A[105] , B[105] , A[132] , B [132]]

O código usado na codificação de **M** é o CRC. Esse código possui uma matriz geradora (G). O Código CRC consiste da multiplicação de **M** por **G** e posteriormente da soma de todos os bits de paridade com todos os bits da matriz **M,** resultando em um oitavo bit de paridade. Ao final desse código, teremos uma matriz similar a **M**, mas com 8 bits de paridade adicionados. A matriz resultante é mostrada abaixo:

M1=[A[18] , B[18] , A[19] , B[19] , A[20] , B[20] , A[21] , B[21] , A[31] , B[31] , A[32] , B[32] , A[53], B[53] , A[80] , B[80] , A[107] , B[107] , A[134] , B[134] , A[1] , B[1] , A[2] , B[2] , A[3] , B[3] , A[4] , B[4] , A[9] , B[9] , A[10] , B[10], A[11] , B[11] , A[12] , B[12] , A[27] , B[27] , A[28] , B[28] , A[29] , B[29] , A[30] , B[30] ,A[52] , B[52], A[79] , B[79] , A[106] , B[106] , A[133] , B[133] , A[51] , B[51] , A[78] , B[78] , A[105] , B[105] , A[132] , B [132] , b1 , b2 , b3 , b4 , b5 , b6 , b7 , b8]

A matriz **M1** continua similar a **M** devido à característica da matriz geradora, que é mostrada logo abaixo. Verifique que da coluna 1 à coluna 60, tem-se uma matriz identidade. Observe que é devido a isso que os bits de 0 a 60 continuam iguais aos de **M**. Note também que a operação **"soma"** equivale à operação binária "**ou exclusivo"**, que tem como tabela verdade a Tabela 3

O código CRC consiste, na verdade, da adição de bits de paridade na classe mais sensível, o que está relacionado com o controle de erros na mensagem transmitida.

Note que os números em colchetes indicam a posição de cada bit em seu respectivo frame.

Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame
1	B35,A	2	B35,B	3	B36,A	4	B36,B	5	B37,A
6	B37,B	7	B38,A	8	B38,B	9	B39,A	10	B39,B
11	B40,A	12	B40,B	13	B41,A	14	B41,B	15	B42,A
16	B42,B	17	B43,A	18	B43,B	19	B47,A	20	B47,B
21	B48,A	22	B48,B	23	B56,A	24	B56,B	25	B61,A
26	B61,B	27	B62,A	28	B62,B	29	B63,A	30	B63,B
31	B64,A	32	B64,B	33	B65,A	34	B65,B	35	B66,A
36	B66,B	37	B67,A	38	B67,B	39	B68,A	40	B68,B
41	B69,A	42	B69,B	43	B70,A	44	B70,B	45	B74,A
46	B74,B	47	B75,A	48	B75,B	49	B83,A	50	B83,B
51	B88,A	52	B88,B	53	B89,A	54	B89,B	55	B90,A
56	B90,B	57	B91,A	58	B91,B	59	B92,A	60	B92,B
61	B93,A	62	B93,B	63	B94,A	64	B94,B	65	B95,A
66	B95,B	67	B96,A	68	B96,B	69	B97,A	70	B97,B
71	B101,A	72	B101,B	73	B102,A	74	B102,B	75	B110,A
76	B110,B	77	B115,A	78	B115,B	79	B116,A	80	B116,B
81	B117,A	82	B117,B	83	B118,A	84	B118,B	85	B119,A
86	B119,B	87	B120,A	88	B120,B	89	B121,A	90	B121,B
91	B122,A	92	B122,B	93	B123,A	94	B123,B	95	B124,A
96	B124,B	97	B128,A	98	B128,B	99	B129,A	100	B129,B
101	B137,A	102	B137,B	103	B58,A	104	B58,B	105	B85,A
106	B85,B	107	B112,A	108	B112,B	109	B54,A	110	B54,B
111	B81,A	112	B81,B	113	B108,A	114	B108,B	115	B135,A
116	B135,B	117	B50,A	118	B50,B	119	B77,A	120	B77,B
121	B104,A	122	B104,B	123	B131,A	124	B131,B	125	B45,A
126	B45,B	127	B72,A	128	B72,B	129	B99,A	130	B99,B
131	B126,A	132	B126,B	133	B55,A	134	B55,B	135	B82,A
136	B82,B	137	B109,A	138	B109,B	139	B136,A	140	B136,B
141	B5,A	142	B5,B	143	B13,A	144	B13,B	145	B34,A
146	B34,B	147	B8,A	148	B8,B	149	B16,A	150	B16,B
151	B17,A	152	B17,B	153	B22,A	154	B22,B	155	B23,A
156	B23,B	157	B24,A	158	B24,B	159	B25,A	160	B25,B
161	B26,A	162	B26,B	163	B6,A	164	B6,B	165	B14,A
166	B14,B	167	B7,A	168	B7,B	169	B15,A	170	B15,B
171	B60,A	172	B60,B	173	B87,A	174	B87,B	175	B114,A
176	B114,B	177	B46,A	178	B46,B	179	B73,A	180	B73,B
181	B100,A	182	B100,B	183	B127,A	184	B127,B	185	B44,A
186	B44,B	187	B71,A	188	B71,B	189	B98,A	190	B98,B
191	B125,A	192	B125,B	193	B33,A	194	B33,B	195	B49,A
196	B49,B	197	B76,A	198	B76,B	199	B103,A	200	B103,B
201	B130,A	202	B130,B	203	B59,A	204	B59,B	205	B86,A
206	B86,B	207	B113,A	208	B113,B	209	B57,A	210	B57,B
211	B84,A	212	B84,B	213	B111,A	214	B111,B	215	B18,A
216	B18,B	217	B19,A	218	B19,B	219	B20,A	220	B20,B
221	B21,A	222	B21,B	223	B31,A	224	B31,B	225	B32,A

(continued)

Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame	Bit no.	Bit number in speech frame
226	B32,B	227	B53,A	228	B53,B	229	B80,A	230	B80,B
231	B107,A	232	B107,B	233	B134,A	234	B134,B	235	B1,A
236	B1,B	237	B2,A	238	B2,B	239	B3,A	240	B3,B
241	B4,A	242	B4,B	243	B9,A	244	B9,B	245	B10,A
246	B10,B	247	B11,A	248	B11,B	249	B12,A	250	B12,B
251	B27,A	252	B27,B	253	B28,A	254	B28,B	255	B29,A
256	B29,B	257	B30,A	258	B30,B	259	B52,A	260	B52,B
261	B79,A	262	B79,B	263	B106,A	264	B106,B	265	B133,A
266	B133,B	267	B51,A	268	B51,B	269	B78,A	270	B78,B
271	B105,A	272	B105,B	273	B132,A	274	B132,B	275	parity b1

Tabela 2: Concatenação dos frames A e B e reposicionamento dos bits.

Note que a matriz geradora só gera 7 bits de paridade. O oitavo bit de paridade corresponde à soma de todos os bits de M com os 7 bits de paridade obtidos com o auxílio da matriz geradora.

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	0

Tabela 3 : Tabela verdade da operação "ou exclusivo".

No Apêndice A será mostrado como foi implementado a Tabela 2 no programa SITE, e em seguida, no Apêndice B será mostrado a matriz geradora do código CRC

3 Codificação Convolucional: Convolutional Encoding

Ainda tomando por base a Figura 2 foi feito um procedimento semelhante para o segundo bloco, a Codificação covolucional.

Nessa interface, os bits das classes 0 e 1 serão codificados segundo o código RCPC.

Os bits de classe 2 são deixados desprotegidos. A codificação desta etapa ocorre em dois passos:

- codificação por um código-mãe;
- pontuação do código-mãe.

Os bits de classes 0 e 1 serão codificados separadamente, mas o algoritmo é o mesmo para ambos. Abaixo está a apresentação geral, e mais adiante os dois casos encontram-se especificados.

3.1 Codificação pelo código-mãe:

A entrada para o código-mãe é uma matriz $C_2[k]$ de K_2 bits. Cada bit gerará, ao final desse código, 3 outros bits de acordo com a seguinte regra:

$$V(3(k-1)+i) = \sum_{j=0}^{4} C_2(k-j)g_{i,j} \qquad i=1,2,3, k=1,2,\ldots,K_2 \qquad [1]$$

Sendo que $C_2(k-j) = 0$ para $k<1$. No estudo de cada caso a equação acima será bem analisada.

Os polinômios geradores desse código são:

$$\begin{aligned} G_1(D) &= 1 + D + D^2 + D^3 + D^4 \\ G_2(D) &= 1 + D + D^3 + D^4 \\ G_3(D) &= 1 + D^2 + D^4 \end{aligned} \qquad [2]$$

Note que os coeficientes $g_{i,j}$ são os coeficientes do polinômio gerador, que são utilizados na equação acima.

Dos polinômios geradores, temos:

$$g_{1,0} = g_{1,1} = g_{1,2} = g_{1,3} = g_{1,4} = 1;$$
$$g_{2,0} = g_{2,1} = 1,\ g_{2,2} = 0,\ g_{2,3} = g_{2,4} = 1; \quad \textbf{[3]}$$

A saída desta etapa é a matriz V[k], sendo que k = 1, 2, ... , $3K_2$.

3.2 Pontuação do código-mãe:

Nesta etapa, será selecionada uma certa quantidade de bits daqueles obtidos na etapa anterior, segundo uma regra. A entrada para este código consiste da matriz V[k] de $3K_2$ bits, e a saída consistirá da matriz C_3[k] de K_3 bits. Os números K_2 e K_3 dependem da classe que está sendo codificada. Na pontuação, para os bits de classe 2 será aplicada a taxa 8/18, e para os bits de classe 1 a taxa 2/3. Isso quer dizer que a saída consistirá de K_3 = (18/8) x K_2 bits para a classe 2 e de (3/2) x K_2 bits para a classe 1. A regra referida anteriormente é a seguinte:

$$C_3(j) = V(k) \qquad j = 1,2,\ldots,K_3$$
$$k = Period * \left((j-1)\operatorname{div} t\right) + P\left(j - t\left((j-1)\operatorname{div} t\right)\right) \quad \textbf{[4]}$$

Observe que as constantes Período, t e P(t) são características de cada classe, e serão especificadas mais à frente.

3.3 Codificação dos bits de classe 2:

3.3.1 Código-mãe

A entrada desta etapa consiste de 72 bits (Matriz **M10[k] = C_2[k]** , k=0..72). Totalizando, ao final, 216 bits (Matriz **V[k]** , k = 1, 2,..., 216).

M10=[A[18] , B[18] , A[19] , B[19] , A[20] , B[20] , A[21] , B[21] , A[31] , B[31] , A[32], B[32] , A[53], B[53] , A[80] , B[80] , A[107] , B[107] , A[134] , B[134] , A[1] , B[1], A[2] , B[2] , A[3] , B[3] , A[4] , B[4] , A[9] , B[9] , A[10] , B[10], A[11] , B[11] , A[12], B[12] , A[27] , B[27] , A[28] , B[28] ,

A[29] , B[29] , A[30] , B[30] ,A[52] , B[52], A[79], B[79] , A[106] , B[106] , A[133] , B[133] , A[51] , B[51] , A[78] , B[78] , A[105] , B[105], A[132] , B [132] , b1 , b2 , b3 , b4 , b5 , b6 , b7 , b8, 0, 0, 0, 0]

Note que foram adicionados quatro bits iguais a zero ao final da matriz M1, resultando na matriz M10. Esses quatro bits são chamados de tail bits, e têm a função de inicializar o código convolucional.

Foi adotado i = 1, 2 e 3 , e alguns valores de k para a construção do algorímo, como mostrado abaixo.

- Para i = 1:

$$V[3(k-1)+1] = V[3k-2] \quad \textbf{[5]}$$

Para k=1,2,3,4,...,72 então:

$$V[k = 1] = V[1] = C_2[1] \quad \textbf{[6]}$$

$V[k = 2] = V[4] = C_2[2] + C_2[1]$

$V[k = 3] = V[7] = C_2[3] + C_2[2] + C_2[1]$

$V[k = 4] = V[10] = C_2[4] + C_2[3] + C_2[2] + C_2[1]$

$V[k = 5] = V[13] = C_2[5] + C_2[4] + C_2[3] + C_2[2] + C_2[1]$

$V[k = 6] = V[16] = C_2[6] + C_2[5] + C_2[4] + C_2[3] + C_2[2]$

$V[k = 7] = V[19] = C_2[7] + C_2[6] + C_2[5] + C_2[4] + C_2[3]$

.... ….

$V[k = k] = V[3k-2] = C_2[k]*g_{1,0} + C_2[k-1]*g_{1,1} + C_2[k-2]*g_{1,2} + C_2[k-3]*g_{1,3} + C_2[k-4]*g_{1,4}$

.... ….

$V[k = 71] = V[211] = C_2[71] + C_2[70] + C_2[69] + C_2[68] + C_2[67]$

$V[k = 72] = V[214] = C_2[70] + C_2[69] + C_2[68] + C_2[67] + C_2[66]$

- Para i = 2:

$$V(3(k-1)+2) = V(3k-1) \quad \textbf{[7]}$$

Para k=1,2,3,4,...,72 então:

$$V[k = 1] = V[2] = C_2[1] \quad \textbf{[8]}$$

$V[k = 2] = V[5] = C_2[2] + C_2[1]$

$V[k = 3] = V[8] = C_2[3] + C_2[2]$

$V[k = 4] = V[11] = C_2[4] + C_2[3] + C_2[1]$

$V[k = 5] = V[14] = C_2[5] + C_2[4] + C_2[2] + C_2[1]$

$V[k = 6] = V[17] = C_2[6] + C_2[5] + C_2[3] + C_2[2]$

$V[k = 7] = V[20] = C_2[7] + C_2[6] + C_2[4] + C_2[3]$

.... …. ....

$V[k = k] = V[3k-1] = C_2[k]*g_{2,0} + C_2[k-1]*g_{2,1} + C_2[k-2]*g_{2,2} + C_2[k-3]*g_{2,3} + C_2[k-4]*g_{2,4}$

.... …. ....

$V[k = 71] = V[212] = C_2[71] + C_2[70] + C_2[68] + C_2[67]$

$V[k = 72] = V[215] = C_2[70] + C_2[69] + C_2[67] + C_2[66]$

- Para i = 3:

$$V[3(k-1)+3] = V[3k] \quad \textbf{[9]}$$

Para k=1,2,3,4,...,72 então:

$$V[k = 1] = V[1] = C_2[1] \quad \textbf{[10]}$$

$V[k = 2] = V[4] = C_2[2]$

$V[k = 3] = V[7] = C_2[3] + C_2[1]$

$V[k = 4] = V[10] = C_2[4] + C_2[2]$

$V[k = 5] = V[13] = C_2[5] + C_2[3] + C_2[1]$

$V[k = 6] = V[16] = C_2[6] + C_2[4] + C_2[2]$

$V[k = 7] = V[19] = C_2[7] + C_2[5] + C_2[3]$

.... ….

$V[k = k] = V[3k-2] = C_2[k]*g_{3,0} + C_2[k-1]*g_{3,1} + C_2[k-2]*g_{3,2} + C_2[k-3]*g_{3,3} + C_2[k-4]*g_{3,4}$

.... **....**

$V[k = 71] = V[213] = C_2[71] + C_2[69] + C_2[67]$

$V[k = 72] = V[216] = C_2[70] + C_2[68] + C_2[66]$

Note que para cada um dos 3 valores de i foram computados 72 diferentes valores para a matriz V[k]. Isso completa o código-mãe, e possibilita que o código continue.

3.3.2 Pontuação do Código-Mãe

A entrada desta etapa consiste de 216 bits (Matriz **V[k]** , k=0, 1, ..., 216). Totalizando (18/8)x72 = 162 bits (Matriz $\mathbf{C_3[k]}$, k = 1, 2, ..., 162).

As constantes utilizadas na equação para pontuação são:

t = 9;

P(1) = 1, P(2) = 2, P(3) = 3, P(4) = 4, P(5) = 5,
P(6) = 7, P(7) = 8, P(8) = 10, P(9) = 11 e

[11]

Period = 12.

Substituindo na equação as constantes e utilizando alguns valores para j, tem-se:

$$k[j] = 12*((j-1)\text{div}9) + P(j-9((j-1)\text{div}9)),\ j = 1, 2, ..., 162 \quad \textbf{[12]}$$

A operação div9 retorna o valor da divisão inteira de um número por 9. Por exemplo:

1div9 = 0;
9div9 = 1;
10div9 = 1;
18div9 = 2.

Para j = 1, 2, 3, ..., 162, obtem-se:

k[j=1] = P[1] = 1
k[j=2] = P[2] = 2
k[j=3] = P[3] = 3
k[j=4] = P[4] = 4
k[j=5] = P[5] = 5
k[j=6] = P[6] = 7
k[j=7] = P[7] = 8
k[j=8] = P[8] = 10 **[13]**
k[j=9] = P[9] = 11
k[j=10] = 12+P[1] =13
k[j=11] = 12+P[2] =14

... ...

k[j=19] = 24+P[1] = 25

k[j=20] = 24+P[2] =26

... ...

k[j=162] = 204+P[9] =215

Com os valores de k[j] obtidos, pode-se então selecionar os 162 bits da matriz V[k] anterior, segundo a regra abaixo:

$C_3[j] = V[k]$ **[14]**

Isso completa a pontuação do código-mãe para a classe 2.

3.4 Codificação dos bits de classe 1:

3.4.1 Código-mãe

A entrada desta etapa consiste de 112 bits (Matriz **M11[k] = C_2[k]**, k=1, 2,..., 112). Totalizando 336 bits (Matriz **V[k]** , k = 1, 2, ..., 112).

M11=[A[58] , B[58] , A[85] , B[85] , A[112] , B[112] , A[54] , B[54] , A[81] , B[81], A[108] , B[108] , A[135], B[135] , A[50] , B[50] , A[77] , B[77] , A[104] , B[104], A[131], B[131] , A[45] , B[45] , A[72] , B[72] , A[99] , B[99] , A[126] , B[126] , A[55] , B[55], A[82] , B[82] , A[109] , B[109] , A[136] , B[136] , A[5] , B[5] , A[13] , B[13] , A[34], B[34] ,A[8] , B[8], A[16] , B[16] , A[17] , B[17] , A[22] , B[22] , A[23] , B[23] , A[24], B[24] , A[25] , B[25] , A[26] , B [26] , A[6], B[6], A[14], B[14], A[7], B[7], A[15], B[15], A[60], B[60], A[87], B[87], A[114], B[114], A[46], B[46], A[73], B[73], A[100], B[100], A[127], B[127], A[44], B[44], A[71], B[71], A[98], B[98], A[125], B[125], A[33], B[33], A[49], B[49], A[76], B[76], A[103], B[103], A[130], B[130], A[59], B[59], A[86], B[86], A[113], B[113], A[57], B[57], A[84], B[84], A[111], B[111]].

Adotando o mesmo procedimento feito na etapa 2.3.1, então: i = 1, 2 e 3 , e alguns valores de k. Então:

- Para i = 1:

V[3(k-1)+1] = V[3k-2] **[15]**

Para k=1,2,3,4,...,112 então:

$V[k = 1] = V[1] = C_2[1]$

$V[k = 2] = V[4] = C_2[2] + C_2[1]$

$V[k = 3] = V[7] = C_2[3] + C_2[2] + C_2[1]$

$V[k = 4] = V[10] = C_2[4] + C_2[3] + C_2[2] + C_2[1]$ **[16]**

$V[k = 5] = V[13] = C_2[5] + C_2[4] + C_2[3] + C_2[2] + C_2[1]$

$V[k = 6] = V[16] = C_2[6] + C_2[5] + C_2[4] + C_2[3] + C_2[2]$

$V[k = 7] = V[19] = C_2[7] + C_2[6] + C_2[5] + C_2[4] + C_2[3]$

.... ….

$V[k = k] = V[3k-2] = C_2[k]*g_{1,0} + C_2[k-1]*g_{1,1} + C_2[k-2]*g_{1,2} + C_2[k-3]*g_{1,3} + C_2[k-4]*g_{1,4}$

.... ….

$V[k = 111] = V[331] = C_2[111] + C_2[110] + C_2[109] + C_2[108] + C_2[107]$

$V[k = 112] = V[334] = C_2[112] + C_2[111] + C_2[110] + C_2[109] + C_2[108]$

- Para i = 2:

$V(3(k-1)+2) = V(3k-1)$ **[17]**

Para k=1,2,3,4,...,112 então:

$V[k = 1] = V[2] = C_2[1]$ **[18]**

$V[k = 2] = V[5] = C_2[2] + C_2[1]$

$V[k = 3] = V[8] = C_2[3] + C_2[2]$

$V[k = 4] = V[11] = C_2[4] + C_2[3] + C_2[1]$

$V[k = 5] = V[14] = C_2[5] + C_2[4] + C_2[2] + C_2[1]$

$V[k = 6] = V[17] = C_2[6] + C_2[5] + C_2[3] + C_2[2]$

$V[k = 7] = V[20] = C_2[7] + C_2[6] + C_2[4] + C_2[3]$

.... ….

$V[k = k] = V[3k-1] = C_2[k]*g_{2,0} + C_2[k-1]*g_{2,1} + C_2[k-2]*g_{2,2} + C_2[k-3]*g_{2,3} + C_2[k-4]*g_{2,4}$

.... ….

$V[k = 111] = V[332] = C_2[111] + C_2[110] + C_2[108] + C_2[107]$

$V[k = 112] = V[335] = C_2[112] + C_2[111] + C_2[109] + C_2[108]$

- Para i = 3:

$$V[3(k-1)+3] = V[3k] \quad \textbf{[19]}$$

Para k=1,2,3,4,...,112 então:

$V[k = 1] = V[1] = C_2[1]$

$V[k = 2] = V[4] = C_2[2]$

$V[k = 3] = V[7] = C_2[3] + C_2[1]$

$V[k = 4] = V[10] = C_2[4] + C_2[2]$

$V[k = 5] = V[13] = C_2[5] + C_2[3] + C_2[1]$

$V[k = 6] = V[16] = C_2[6] + C_2[4] + C_2[2]$

$V[k = 7] = V[19] = C_2[7] + C_2[5] + C_2[3]$

.... ….

$V[k = k] = V[3k-2] = C_2[k]*g_{3,0} + C_2[k-1]*g_{3,1} + C_2[k-2]*g_{3,2} + C_2[k-3]*g_{3,3} + C_2[k-4]*g_{3,4}$

.... ….

$V[k = 111] = V[333] = C_2[111] + C_2[109] + C_2[107]$

$V[k = 112] = V[336] = C_2[112] + C_2[110] + C_2[108]$

Note que para cada um dos 3 valores de i foram computados 72 diferentes valores para a matriz V[k]. Isso completa o código-mãe, e possibilita que o código continue.

3.4.2 Pontuação do Código-Mãe

A entrada desta etapa consiste de 336 bits (Matriz **V[k]** , k = 1, 2, ..., 336). Totalizando (3/2)x112 = 168 bits (Matriz $\mathbf{C_3[k]}$, k = 1, 2, ..., 168).

As constantes utilizadas na equação para pontuação são:

$t = 3;$

$P(1) = 1, P(2) = 2, P(3) = 4$ e **[20]**

Period = 6.

Substituindo na equação as constantes e utilizando alguns valores para j, tem-se:

$$k[j] = 6*((j-1)\text{div}3)+P[j-3*((j-1)\text{div}3)] \quad \textbf{[21]}$$

$$k[j=1] = P[1] = 1 \quad \textbf{[22]}$$

k[j=2] = P[2] = 2
k[j=3] = P[3] = 4
k[j=4] = 6+P[1] = 7
k[j=5] = 6+P[2] = 8
k[j=6] = 6+P[3] = 9
k[j=7] = 12+P[1] = 13
k[j=8] = 12+P[2] = 14
k[j=9] = 12+P[3] = 16
... ...
k[j=167] = 330+P[2] = 332
k[j=168] = 330+P[3] = 333

Com os valores de k[j] obtidos, pode-se então selecionar os 168 bits da matriz V[k] anterior, segundo a regra abaixo:

$$C_3[j] = V[k] \quad \textbf{[23]}$$

Isso completa a pontuação do código-mãe para a classe 1.

O procedimento adotado para implementar esse algoritmo é descrito no Apêndice C:

4 Interleaving

Nesta etapa, os bits das classes 0, 1 e 2 serão concatenados em uma única matriz. Essa matriz será construída da seguinte forma:

$C_4[k] = [M_0, M_1, M_2]$ **[24]**

Os bits das classes 1 e 2 passaram pelo código RCPC e pelo CRC, e as quantidades de bits de ambas as classes foram alteradas. A classe 0 permanece com a mesma quantidade de bits iniciais. A quantidade de bits da matriz $C_4[k]$ é igual à soma dos bits das três classes. As matrizes M_0, M_1 e M_2 indicam as matrizes resultantes das 3 classes de bits após passarem pela codificação.

A matriz $C_4[k]$ possui 432 elementos, os quais serão reordenados segundo a seguinte regra:

$$C_4(i*24+j) = C_3(j*18+i) \quad \textbf{[25]}$$

Os valores **i** e **j** devem variar de forma que a expressão (18*j + i) varie de 1 a 432. Isso é possível fazendo j variar de 0 a 24 e i de 0 a 17, como mostrado na Tabela 4::

i	j	j*18 + i	i*24+j
1	0	1	24
2	0	2	48
3	0	3	72
4	0	4	96
5	0	5	120
6	0	6	144
7	0	7	168
8	0	8	192
9	0	9	216
10	0	10	240
11	0	11	264
12	0	12	288

13	0	13	312
14	0	14	336
15	0	15	360
16	0	16	384
17	0	17	408
0	1	18	1
1	1	19	25
2	1	20	49
3	1	21	73
4	1	22	97
5	1	23	121
6	1	24	145
7	1	25	169
8	1	26	193
9	1	27	217
10	1	28	241
11	1	29	265
12	1	30	289
13	1	31	313
14	1	32	337
i	j	j*18 + i	i*24+j
15	1	33	361
16	1	34	385
17	1	35	409
0	2	36	2
...	...	...	...
0	23	414	23
1	23	415	47
2	23	416	71
3	23	417	95
4	23	418	119
5	23	419	143
6	23	420	167
7	23	421	191
8	23	422	215
9	23	423	239
10	23	424	263
11	23	425	287
12	23	426	311
13	23	427	335

14	23	428	359
15	23	429	383
16	23	430	407
17	23	431	431
0	24	432	24

Tabela 4: Interleaving

Para visualizar o código em Delphi utilizado para implementar o *interleaving* vide Apêndice D

5 *Scrambling (Mistura)*

Nesta etapa, os K_4 bits provenientes do *Interleaving*, $b_4(1),b_4(2),..., b_4(K_4)$, serão transformados em K_5 bits do tipo 5, $b_5(1),b_5(2),..., b_5(K_4)$, com $K_5 = K_4$, de acordo com a seguinte regra:

$b_4[k] = b_4[k] + p[k]$ **[26]**

Sendo que a soma equivale à operação "ou exclusivo", e p[k] é o k-ésimo bit da seqüência de scrambling.

A seqüência de scrambling será gerada pelos 30 bits do "extended colour code" e[1], e[2], ..., e[30] (adotamos para o software o valor 0 para a matriz e).

O k-ésimo bit da seqüência de scrambling é dado pela seguinte equação:

$$p(k) = \sum_{i=1}^{32} c_i p(k-i)$$ **[27]**

Com a seguinte inicialização:

$p(k) = e(1-k)$ para $k = -29, -28, ... 0$; e

$p(k) = 1$ para $k = -31, -30$ **[28]**

Sendo que $c_i = 1$ para i = 0, 1, 2, 4, 5, 7, 8, 10, 11, 12, 16, 22, 23, 26 e 32, e $c_i = 0$ para todos os outros valores.

Substituindo alguns valores para k, pode-se analisar alguns valores da seqüência de scrambling:

$p[k=1] = c_1*p[0] + c_2*p[-1] + c_4*p[-3] + c_5*p[-4] + c_7*p[-6] + c_8*p[-7] + c_{10}*p[-9] + c_{11}*p[-10] + c_{12}*p[-11] + c_{16}*p[-15] + c_{22}*p[-21] + c_{23}*p[-22] + c_{26}*p[-25] + c_{32}*p[-31];$ **[29]**

$p[k=2] = c_1*p[1] + c_2*p[0] + c_4*p[-2] + c_5*p[-3] + c_7*p[-5] + c_8*p[-6] + c_{10}*p[-8] + c_{11}*p[-9] + c_{12}*p[-10] + c_{16}*p[-14] + c_{22}*p[-20] + c_{23}*p[-21] + c_{26}*p[-24] + c_{32}*p[-30];$ **[30]**

$p[k=3] = c_1*p[2] + c_2*p[1] + c_4*p[-1] + c_5*p[-2] + c_7*p[-4] + c_8*p[-5] + c_{10}*p[-7] + c_{11}*p[-8] + c_{12}*p[-9] + c_{16}*p[-13] + c_{22}*p[-19] + c_{23}*p[-20] + c_{26}*p[-23] + c_{32}*p[-29];$ **[31]**

$p[k] = c_1*p[k-1] + c_2*p[k-2] + c_4*p[k-4] + c_5*p[k-5] + c_7*p[k-7] + c_8*p[k-8] + c_{10}*p[k-10] + c_{11}*p[k-11] + c_{12}*p[k--12] + c_{16}*p[k-16] + c_{22}*p[k-22] + c_{23}*p[k-23] + c_{26}*p[k-26] + c_{32}*p[k-32].$ **[32]**

Sendo que os valores p[-31], p[-30], ..., p[0] são dados pelo “extended colour code”, como explicado acima. Esse código varia de acordo com o país e com a rede em que o sistema está inserido. Essa variação é mostrada abaixo:

10 bits Mobile Country Code (MCC) e(1) - e(10) e(1) = msb of MCC	14 bits Mobile Network Code (MNC) e(11) - e(24) e(11) = msb of MNC	6 bits Colour Code e(25) - e(30) e(25) = msb of Colour Code

Tabela 5: ***Extended colour code***

Cada país possui um MCC (Mobile Country Code), e cada rede possui um MNC (Mobile Network Code). No caso do Brasil e para a rede utilizada no LABTECC, os valores de MCC e MNC são, respectivamente, 724 e 25. Logo abaixo, encontram-se os valores de MCC e MNC de alguns países e redes. Os 6 bits do *Colour Code* adotados foram iguais a zero.

Concatenando os bits para formar o “extended colour code”, tem-se:

$(724)_{10} = (1011010100)_2$

$(24)_{10} = (00000000011001)_2$

e = [1;0;1;1;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0;0;1;]

Abaixo, é mostrado na Tabela 6, a relação de alguns códigos (MCC) para alguns países e redes de trabalho.

Em seguida, Figura 6 é mostrada fazendo referencia a uma funcionalidade do SITE que permite apagar todo o processo já realizado para a iniciar um novo, caso seja interesse do usuário.

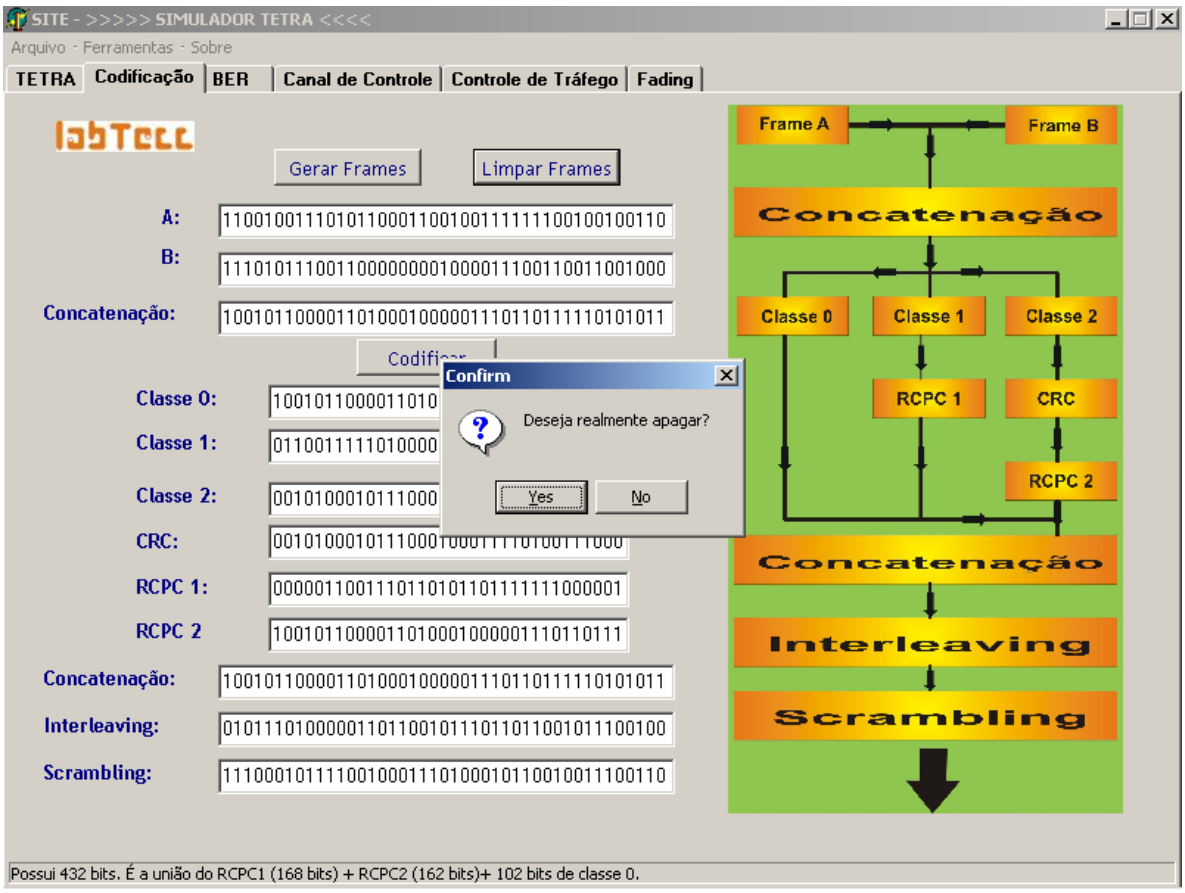


Figura 6 : Opção de limpeza de dados

Country or Geographical Area *Networks*	*MCC + MNC codes^*	
Bangladesh		
GramenPhone	470	01
Aktel	470	02
Mobile 2000	470	03
Belarus		
MDC Velcom	257	01
MTS	257	02
Belgium		
Proximus	206	01
Mobistar	206	10
Base	206	20
Belize		
Belize Telecommunications Ltd., GSM 1900	702	67
International Telecommunications Ltd. (INTELCO)	702	68
Benin		
Libercom	616	01
Telecel	616	02
Spacetel Benin	616	03
Bhutan		
B-Mobile of Bhutan Telecom	402	17
Bosnia and Herzegovina		
Eronet Mobile Communications Ltd.	218	03
MOBI'S (Mobilna Srpske)	218	05
GSMBIH	218	90
Botswana		
Mascom Wireless (Pty) Ltd.	652	01
Orange Botswana (Pty) Ltd.	652	02
Brazil		
Telet	724	00
CRT Cellular	724	01
Global Telecom	724	02
CTMR Cel	724	03
BCP	724	04
Telesc Cel	724	05
Tess	724	06
Sercontel Cel	724	07
Maxitel MG	724	08
Telepar Cel	724	09
ATL Algar	724	10
Telems Cel	724	11
Americel	724	12
Telesp Cel	724	13
Maxitel BA	724	14
CTBC Cel	724	15
BSE	724	16
Ceterp Cel	724	17
Norte Brasil Tel	724	18
Telemig Cel	724	19
Telerj Cel	724	21
Telest Cel	724	23
Telebrasilia Cel	724	25
Telegoias Cel	724	27
Telemat Cel	724	29
Teleacre Cel	724	31
Teleron Cel	724	33

Tabela 6: Códigos MCC- MNC

No Apêndice E o leitor poderá verficar como esta etapa foi implementada.

V. BER - Bit Error Rate

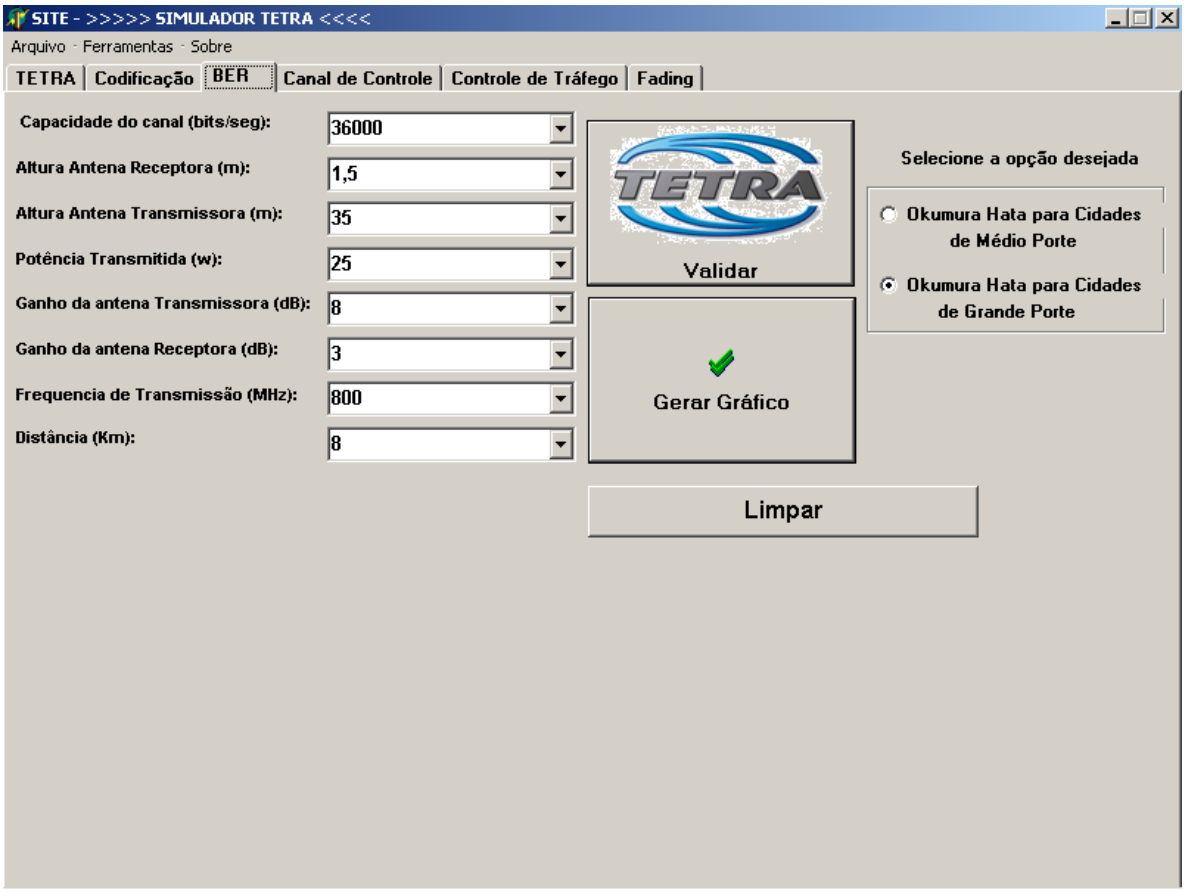


Figura 7 : Selecionando opção para cálculo da BER

Existem dois tipos de BER possíveis de serem consideradas, uma denominada pré-FEC BER e a outra pós-FEC BER. O pré-FEC BER é um teste baseado no número de erros detectados no primeiro estágio de processamento do FEC (Forward Error Correction). Como mede os erros encontrados no sinal recebido, e não do sinal corrigido, ele indica os danos causados por interferências e ruídos, e mostra os efeitos das deteriorações embutidos dentro do *haystack* digital, ou seja, de deteriorações que permanacem escondidas no sinal. Isso nos permite avaliar quando ocorrerá o ingresso na zona de precipício, ou seja, a perda do sinal. O pós-FEC BER é medido através da segunda fase do processamento do FEC, medindo os erros que sobreviveram à primeira fase do processamento, e que permaneceram até o sinal final.

De acordo com simulações feitas com base na referencia [4] para analisar a performance do sistema TETRA, a BER aceitável para diferentes condições de propagação, é dada pela Tabela 7 e mostrada na Figura 9 e Figura 11.

Modelo de canal	BER
Canal Perfeito	Sem transmissão de erros
TU50	1%
HT50	3%
TU100	5%
HT150	5%

Tabela 7: BER aceitavél para diferentes condições de propagação

Os resultados obtidos com o SITE são dados na Tabela 8.

BER	Distância
1%	16,5 Km
3%	18 Km
5%	19 km

Tabela 8: Resultado Simulado pelo SITE

Os passos para a simulação produzida no LABTECC foram feitos de acordo com os cálculos demonstrados abaixo:

1 Simulação da BER

Um conjunto de aproximações analíticas para a perda de propagação foi proposto por Hata, com base na formula de Okumura [5], utilizando a seguinte fórmula para a perda em áreas urbanas:

$$L = 69{,}55 + 26{,}16 \log f - 13{,}82 \log ht - A(hr) + (44{,}9 - 6{,}55 \log ht)\log d \quad (dB) \qquad [33]$$

Onde:

$$150\ MHz \leq f \leq 1000\ MHz$$

$$30m \leq Ht \leq 300\ m$$

$$1km \leq R \leq 20\ km \qquad [34]$$

O fator de correção A(hr) é computado da seguinte forma:

- Para cidades de médio porte

$$A(hr) = (1{,}1 \log f - 0{,}7)hr - (1{,}56 \log f - 0{,}8) \text{ (dB)} \quad [35]$$

em que $1m \leq hr \leq 10m$.

- Para grandes cidades

$$A(hr) = 8{,}29 \log2(1{,}54hr) - 1{,}1 \text{ (dB)} \quad [36]$$

($f \leq 200$ MHz)

$$A(hr) = 3{,}2 \log2(11{,}75hr) - 4{,}97 \text{ (dB)}$$

($f \geq 400$ MHz) [37]

Com o modelo de OKUMURA-HATA é possível obter a perda na potência do sinal transmitido e predizer então qual a potência recebida no aparelho receptor.

Assim temos:

$$Pr = Pt - L \text{ (dB)} \quad [38]$$

em que Pr: Potência do sinal no receptor

Pt: Potência do sinal no transmissor

L: Perdas

A energia do bit é dada por:

$$Eb = \frac{Pr(w)}{\text{Capacidade Canal}} \quad [39]$$

em que a Capacidade do Canal é de 36.000 bps.

A densidade de potência de ruído N_o (W/Hz) é dada pela constante de Boltzmann multiplicada pela temperatura do sistema, que em nosso caso adotamos 300K [6].

A probabilidade erro é encontrada usando as propriedades estatísticas da função erro complementar (1- função erro), é descrita abaixo:

$$Pe = erfc\left(\sqrt{\frac{Eb}{No}}\right) \quad [40]$$

Utilizando os dados da função probabilidade de erro, finalmente encontramos a taxa de erro de bits (BER), assumindo a constelação de sinal com codificação Gray (mudança de apenas um bit para os bits adjacentes):

$$BER = \frac{Pe}{\log_2 M} * 100 \ (\%) \qquad [41]$$

Em que M é igual a 2 bits/símbolo, devido à modulação $\pi/4$ DQPSK, utilizada no sistema TETRA.

Através da Figura 8, pode-se ver um exemplo de simulação gerada pelo software SITE. Neste exemplo, os parâmetros adotados (para o modelo de Okumura Hata para grandes cidades) estão mostrados na tela.

Esse software permite a variação dos dados de entrada, tendo como resultado o gráfico gerado e exemplificado pela Figura 9.

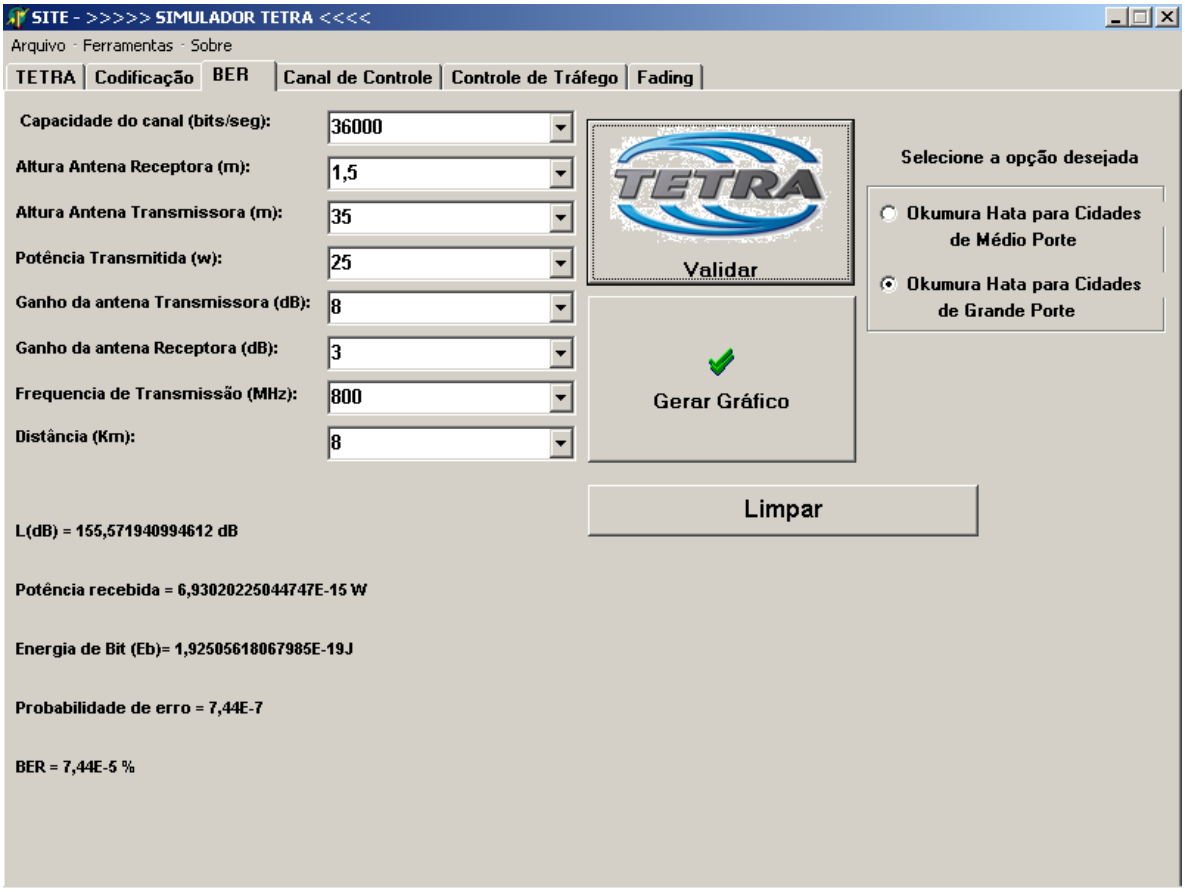


Figura 8 : Cálculo da BER para cidades de grande porte

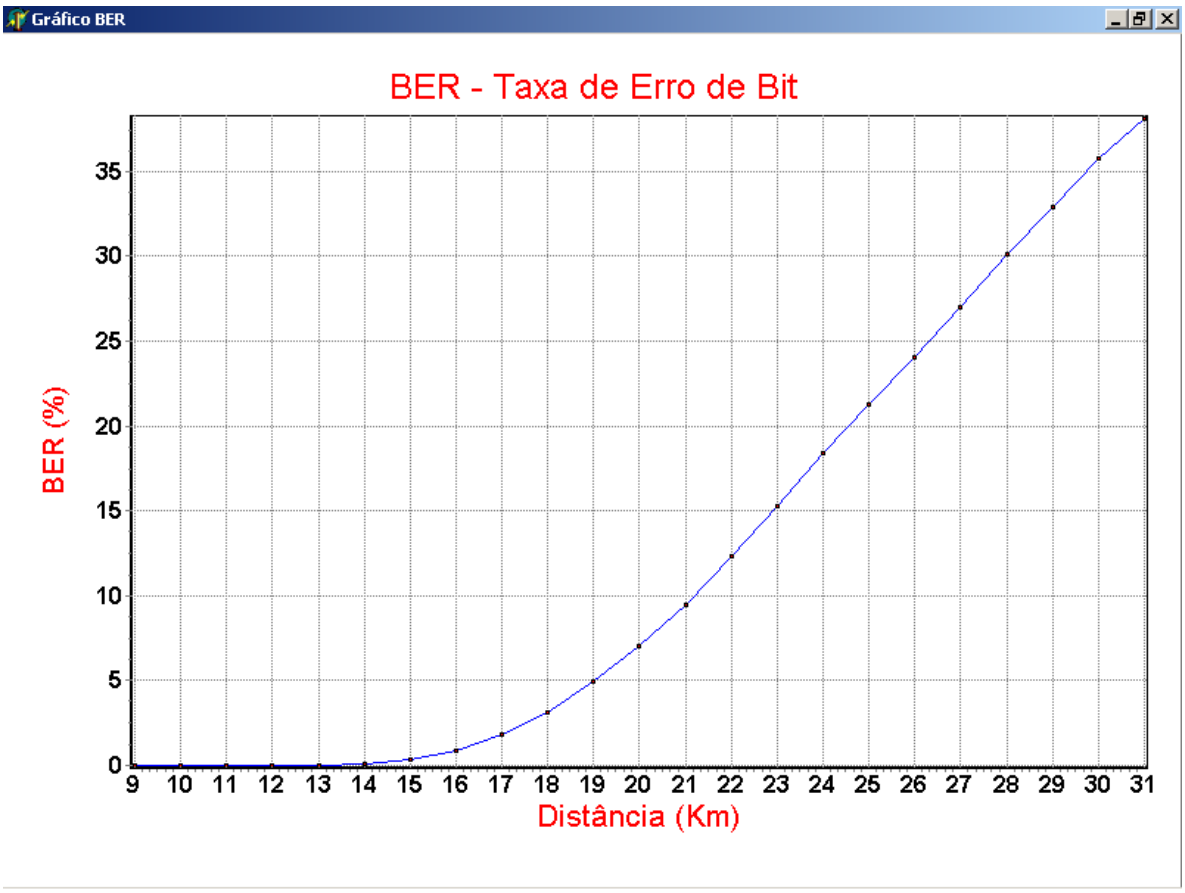


Figura 9 : Gráfico gerado pela BER

O software SITE oferece a possibilidade de se efetuar um zoom nos valores gerados como mostrado na **Figura 11** e descritos na **Tabela 8.** Desta forma, o usuário tem a opção de visualizar dados não mostrados no gráfico original, além de poder visualizar de forma mais precisa uma dada parte da curva.

Abaixo é mostrado um fluxograma dos procedimentos adotados pela BER e em seguida, através do Apêndice F como essa interface foi implementada.

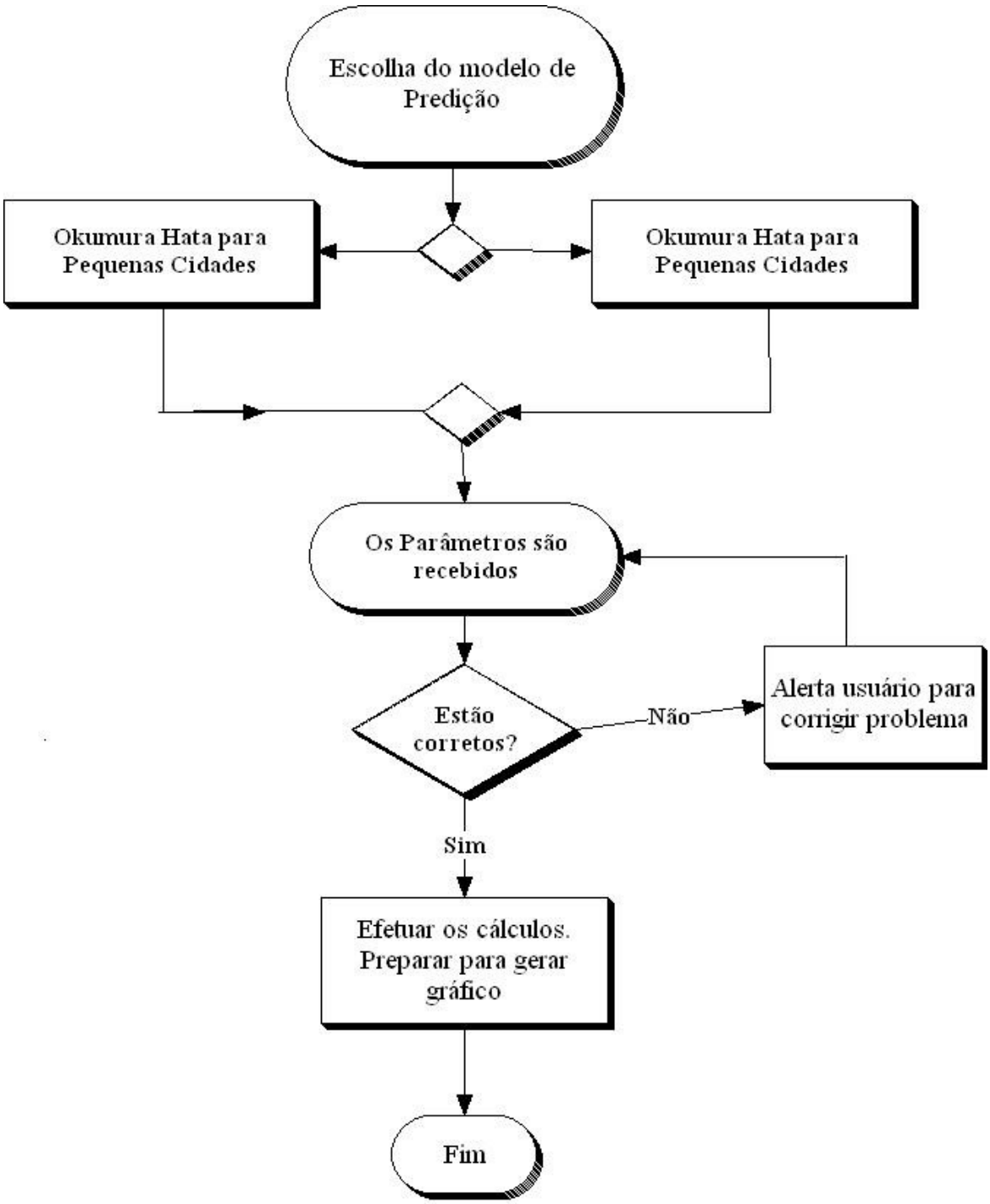


Figura 10: Fluxograma da BER

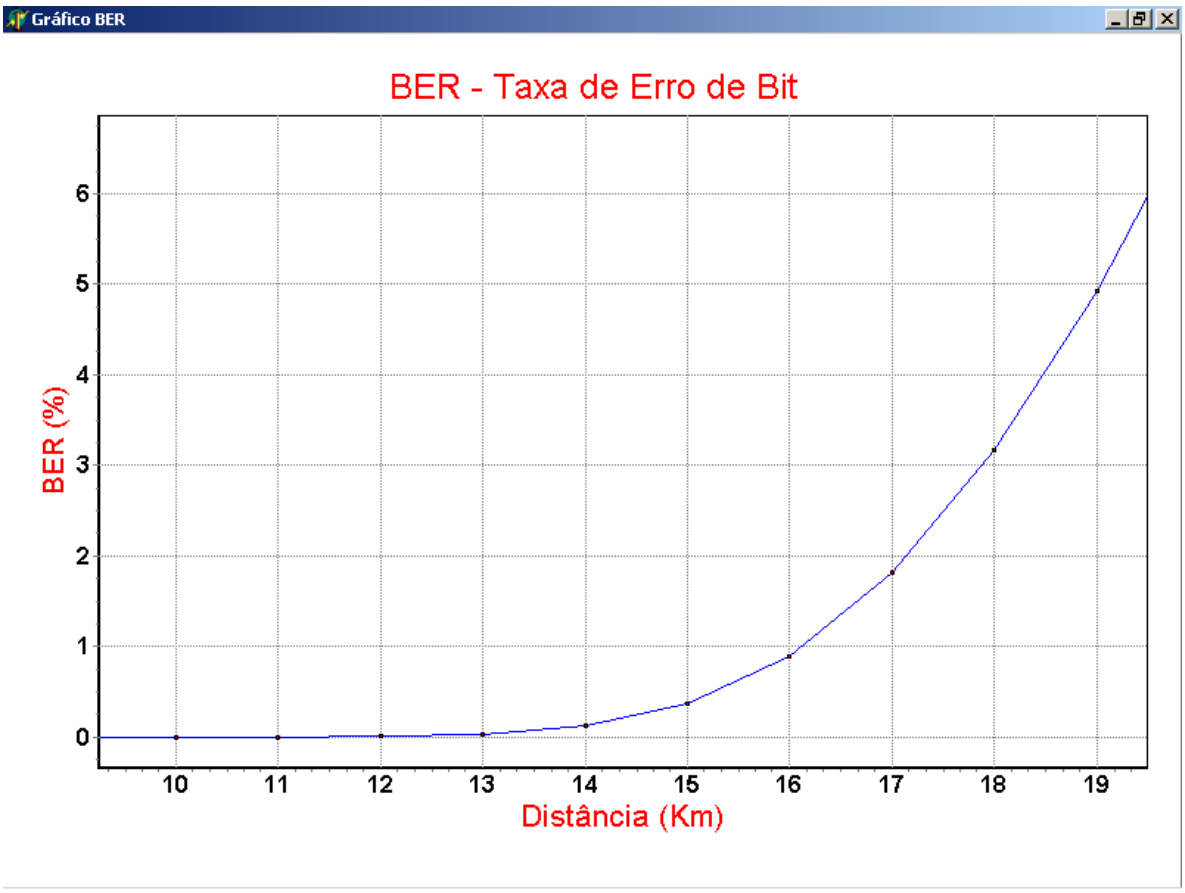


Figura 11: Zoom na curva da BER para 1% e 5 %

O código utilizado para implementação da BER está descrito no Apêndice F.

VI. Canal de controle

O diagrama da estrutura da TDMA é mostrada na Figura 12. As transmissões dos bits de *downlink* e *uplink* são feitas com diferença de dois *time-slots* para permitir que a operação *half-duplex* seja suportada por rádios que não possuam duplexador.

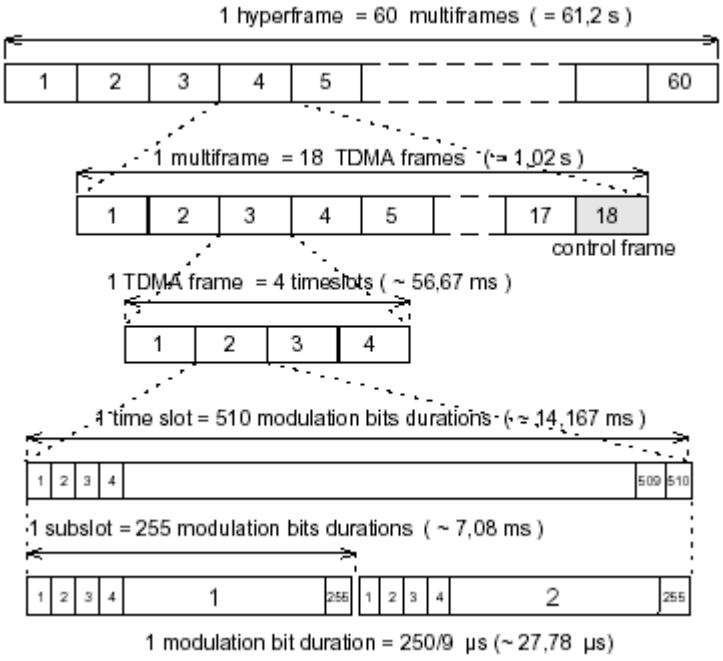


Figura 12: Representação diagramática da estrutura TDMA

Um par de portadoras por célula é designado para “carregar” o principal canal de controle (MCCH). Em condições normais, o *timeslot* 1 de cada frame (*uplink* e *downlink*) de cada portadora desse par é alocado para propósitos de controle, que é chamado de canal de controle físico (CP). Os outros três *time slots* (canais) de cada portadora são usados para tráfego, e são chamados de canais físicos de tráfego (TC). Os dezessete primeiros *time slots* de um canal físico de tráfego transmitem tráfego. O décimo oitavo *time slot* desse canal transmite informações de controle. Portanto, informações de controle podem ser transmitidas tanto no canal físico de controle quanto no de tráfego.

Os bits no padrão TETRA são transmitidos através de *bursts*. Há sete tipos de *bursts*, sendo suas estruturas mostradas na Figura 13 Os conteúdos de cada tipo de *burst* são definidos no padrão [7]

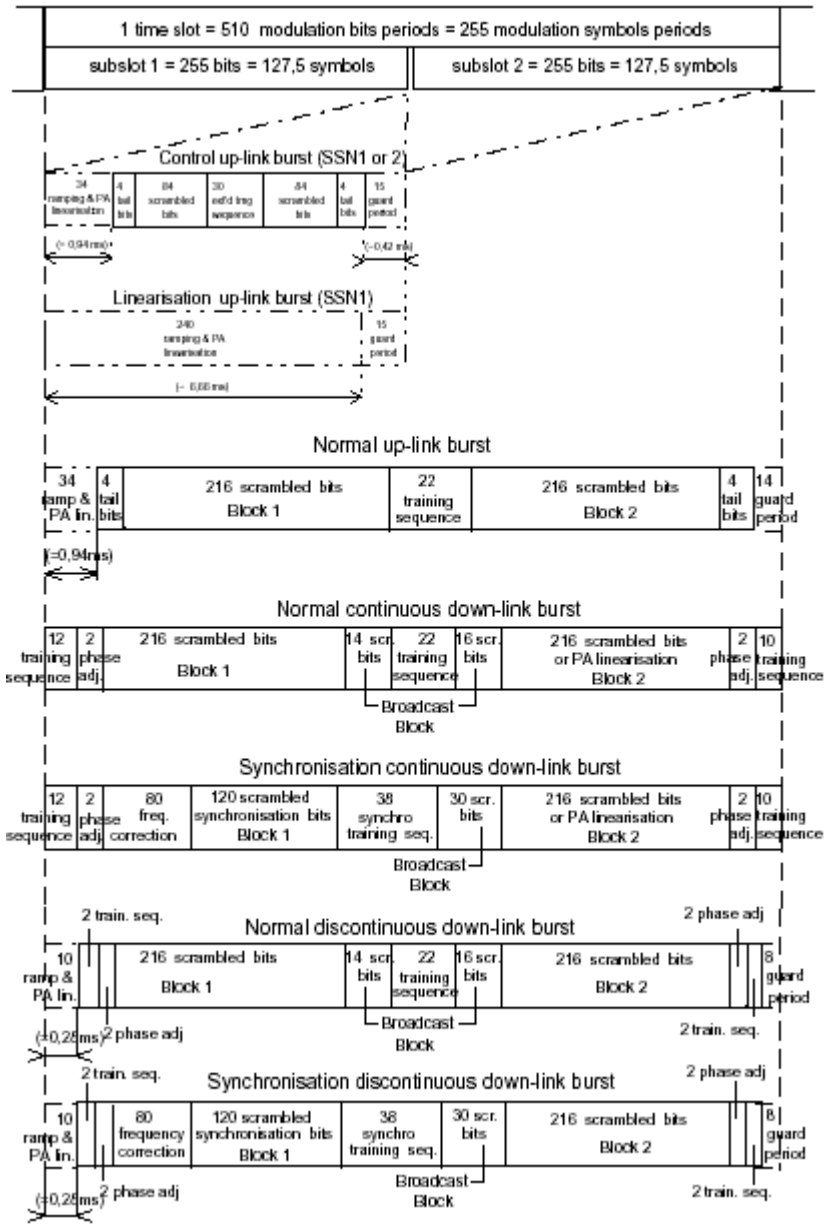


Figura 13: Tipos de bursts

O mapeamento dos canais lógicos em canais físicos é dado na Figura 14, para diferentes tipos de *bursts*.

Logical channel	Dire-ction	Burst type	SSN/ Block	Physical channel	*FN*	*TN*
BSCH	DL	SB	*BKN1* *BKN1*	CP, TP UP	18 1...18	4-(*MN*+1)mod4# 1...4
BNCH	DL	NDB NDB SB	*BKN2* *BKN2* *BKN2*	CP,TP CP UP	18 1...18 1...18	4-(*MN*+3)mod4# 1...4 1...4
AACH	DL	NDB, SB	*BBK*	CP, TP, UP	1...18	1...4#
BLCH	DL	NDB,SB	*BKN2*	CP, UP TP	1...18 18	1...4 1...4
CLCH	UL	LB	*SSN1* *SSN1*	CP, TP CP, UP	18 1...18	4-(*MN*+1)mod4# 1...4
SCH/F	DL	NDB	*BKN1+BKN2*	CP TP	1...18 18	1...4 1...4
	UL	NUB	*BKN1+BKN2*	CP TP	1...18 18	1...4 1...4
SCH/HD	DL	NDB, SB	*BKN1, BKN2*	CP, UP TP	1...18 18	1...4 1...4
SCH/HU	UL	CB	*SSN1, SSN2*	CP TP	1...18 18	1...4 1...4
TCH	DL	NDB	*BKN1, BKN2*	TP	1...17	1...4
	UL	NUB	*BKN1, BKN2*	TP	1...17	1...4
STCH	DL	NDB	*BKN1, BKN2*	TP	1...17	1...4
	UL	NUB	*BKN1, BKN2*	TP	1...17	1...4
NOTE: # indicates a mandatory mapping.						

Figura 14: Mapeamento dos canais lógicos em canais físicos

Uma ilustração de um circuito de chamada foi feita no SITE, e mostra a transmissão de mensagens de controle e tráfego. Cada seta indica a transmissão de *downlink* ou *uplink*, sendo explicadas no programa.

Abaixo é mostrado a interface para o canal de controle, através da Figura 15.

No programa é possível visualizar uma simulação do canal de controle de três formas:

- Simulação Automática: Onde o usuário, apenas com um clique, pode ver toda a simulação acontecer.

- Simulação Manual: Onde o usuário pode efetuar passo à passo a simulação, tendo a opção de avançar e retroceder.

Da Figura 16 à Figura 30 é mostrada essa simulação. Note que na parte inferior da tela, existe uma outra janela, onde são mostradas informações a respeito da operação realizada e dos protocolos de comunicação utilizados naquela operação.

Existe ainda uma última opção:

- Direto: Essa opção já se inicia na Figura 30, a diferença é que o usuário tem a possibilidade de variar a distância que o móvel se encontrará da antena transmissora.

Ao final da simulação fica disponível uma opção “Saiba Mais!!!”, onde o usuário pode obter mais informações sobre o protocolo *U-Setup*. Essa explicação está mostrada na Figura 17.

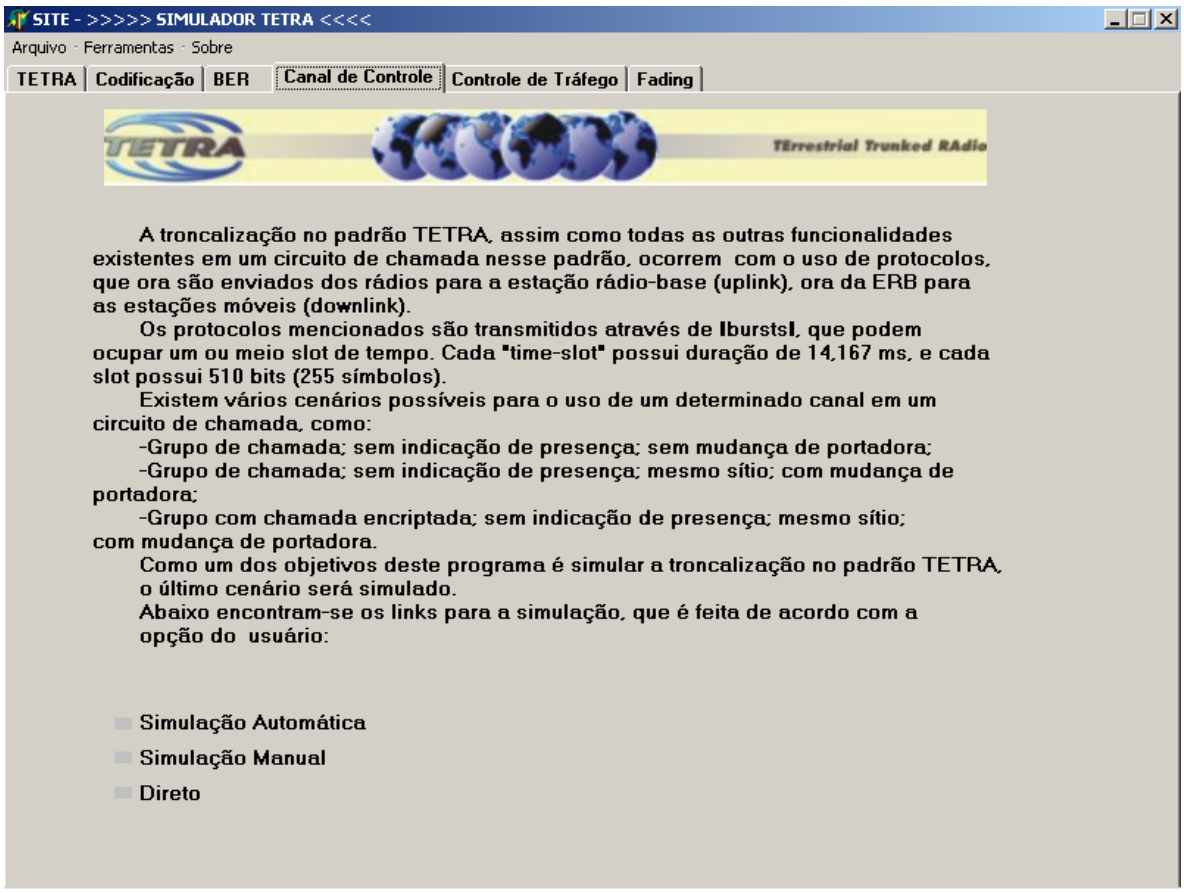


Figura 15 : Tela de apresentação para o canal de controle

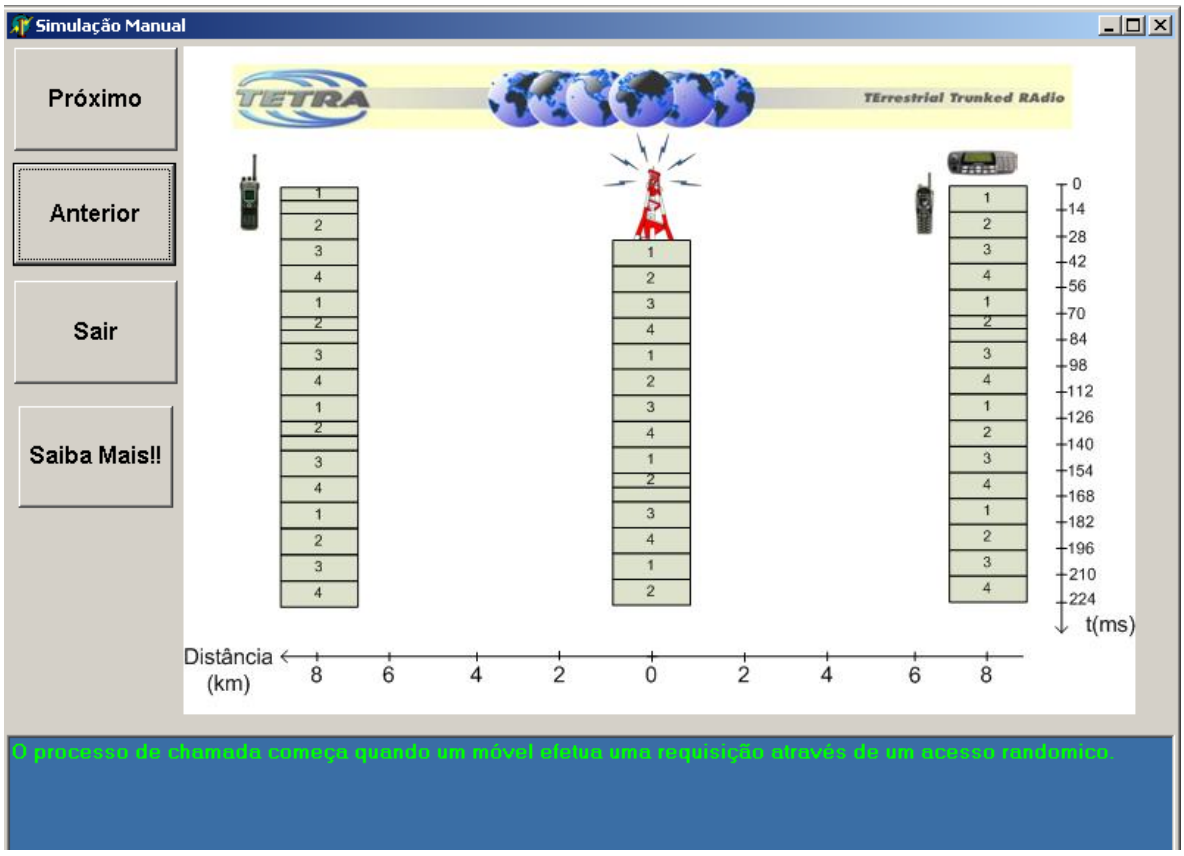


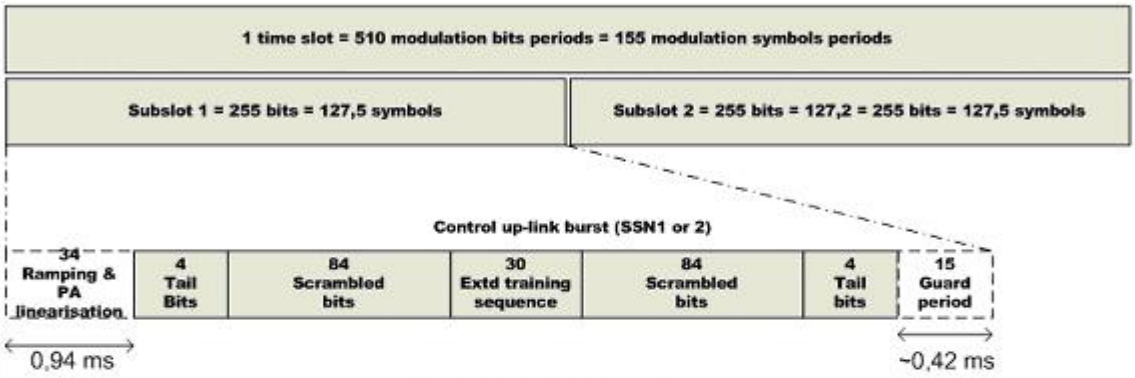
Figura 16: Simulação manual do canal de controle etapa 0

Esquema do protocolo U-Setup

O início de um circuito de chamada se dá pela transmissão de um burst de up-link (U-Setup), do rádio para a ERB. Após o recebimento desse protocolo, a estação base envia uma autenticação do pedido para o início da chamada através de um burst de down-link (D-call proceeding), indicando que a chamada está sendo processada.

O esquema do burst que "carrega" o protocolo U-setup é mostrado abaixo. Tal burst não é usado somente pelo protocolo U-setup, mas também para a transmissão de outras mensagens de controle dos rádios para a estação base.

Abaixo da figura encontra-se uma tabela explicativa sobre o conteúdo do burst, assim como os bits da "extended training sequence" e os "tail bits", que não variam.



Control uplink Burst (CB)

Bit Number (BN)	Field length (bits)	Field content
1 to 4	4	tail bits
5 to 88	84	scrambled control bits
89 to 118	30	extended training sequence
119 to 202	84	scrambled control bits
203 to 206	4	tail bits

Tail bits: (t1,t2,t3,t4) = (1, 1, 0, 0)

Extended training sequence: (y1,y2,...,y38) = (1,1, 0,0, 0,0, 0,1, 1,0, 0,1, 1,1, 0,0, 1,1, 1,0, 1,0, 0,1, 1,1, 0,0, 0,0, 0,1, 1,0, 0,1, 1,1)

Scrambled bits: Nesses bits encontram-se as informações necessárias para o início da chamada, tal como o ID do rádio a ser chamado ou do grupo de chamada.

Figura 17: Opção "Saiba Mais" sobre o protocolo *U-Setup*

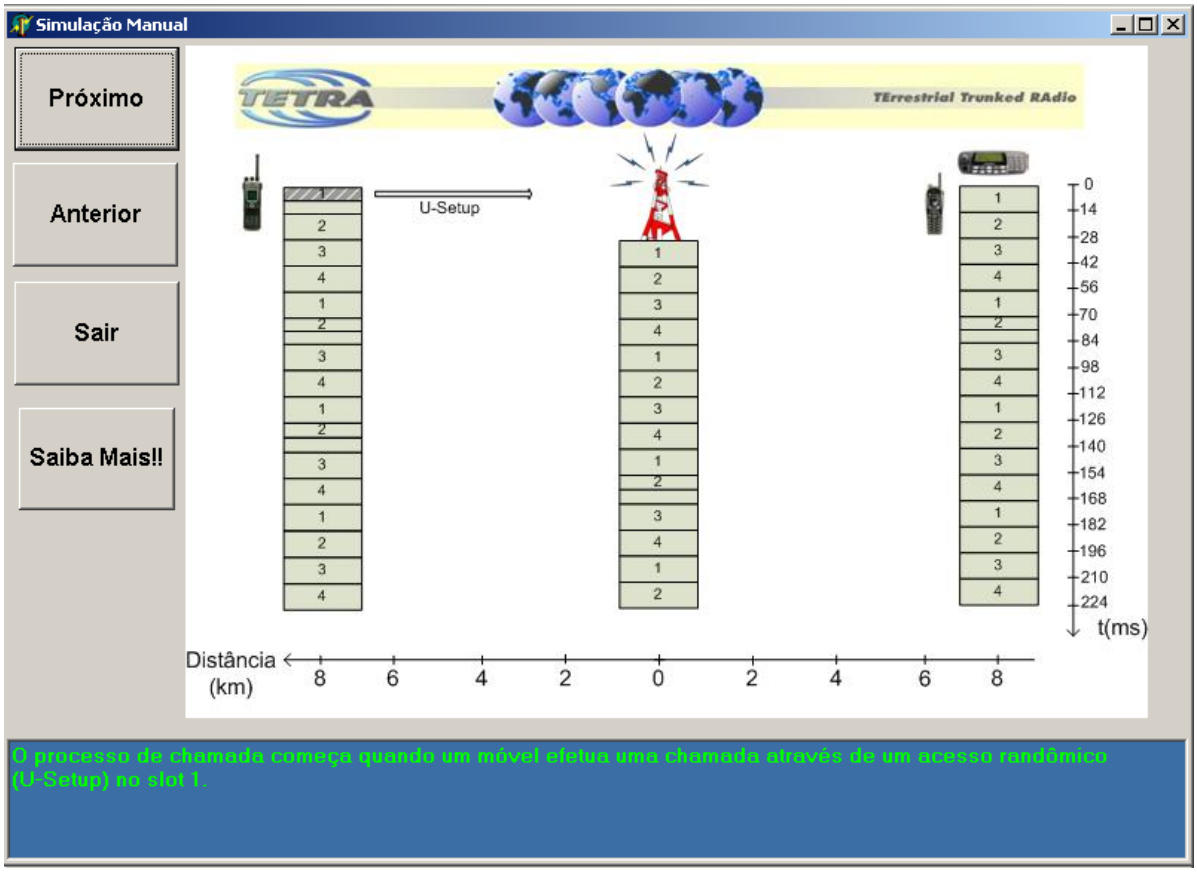


Figura 18: Simulação manual do canal de controle etapa 1 *(U-Setup*)

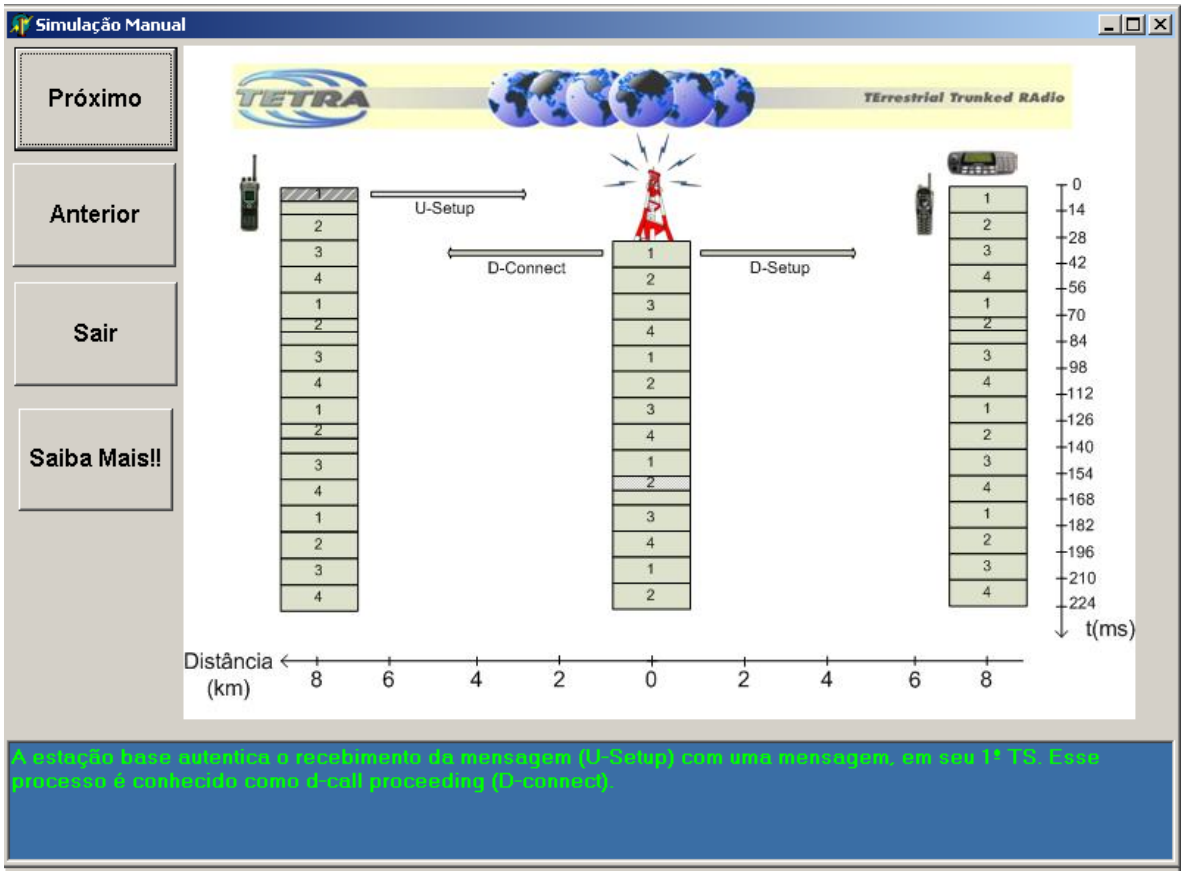


Figura 19: Simulação manual do canal de controle etapa 2 (*D-connect e D-Setup*)

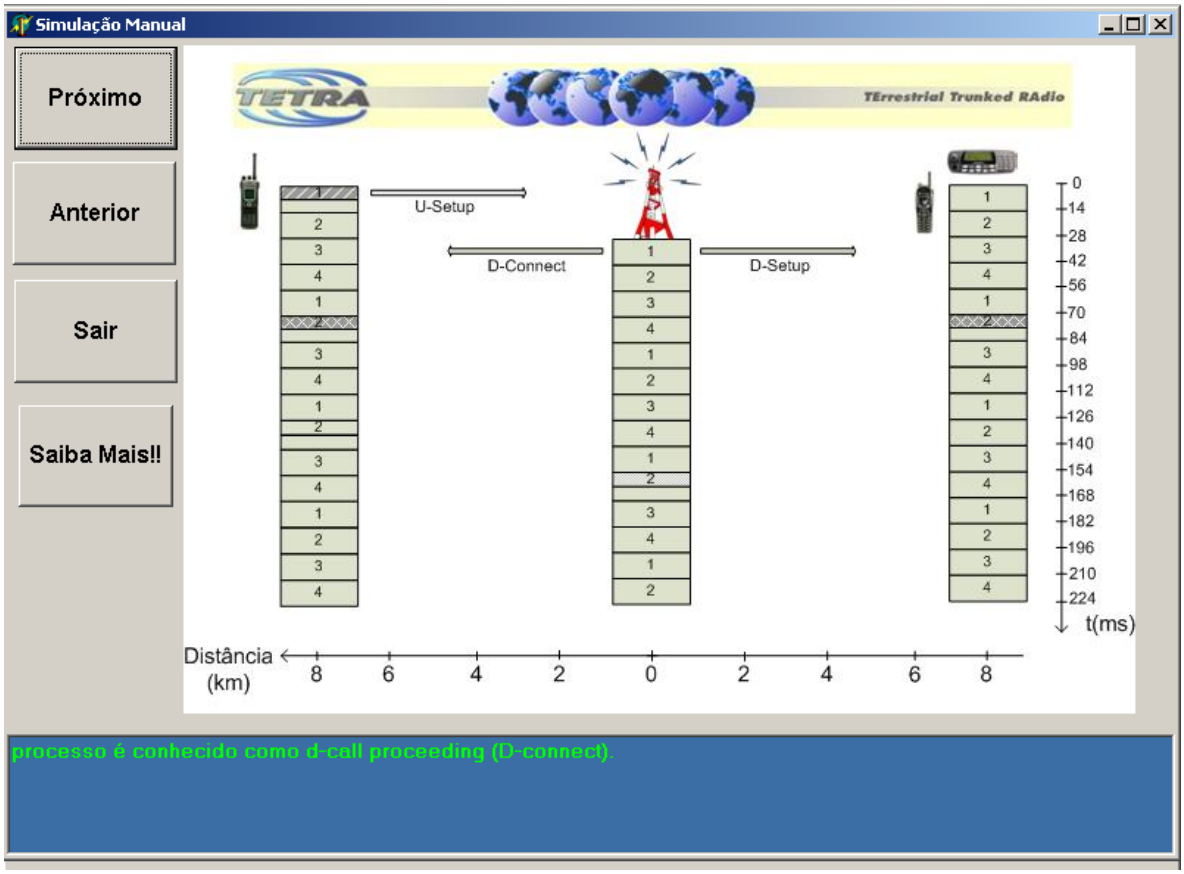


Figura 20: Mostra a requisição de meio *time slot* para a linearização (etapa 3)

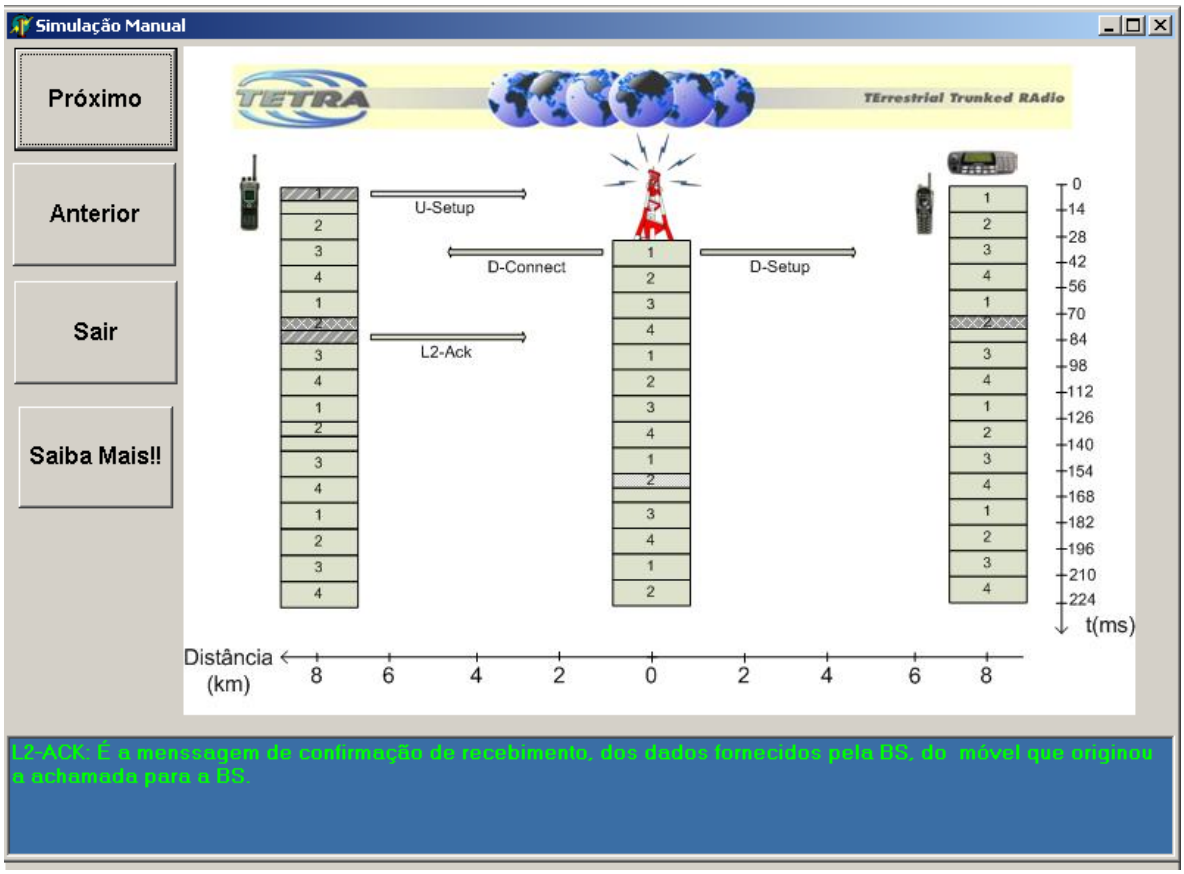


Figura 21: Simulação manual do canal de controle etapa 4

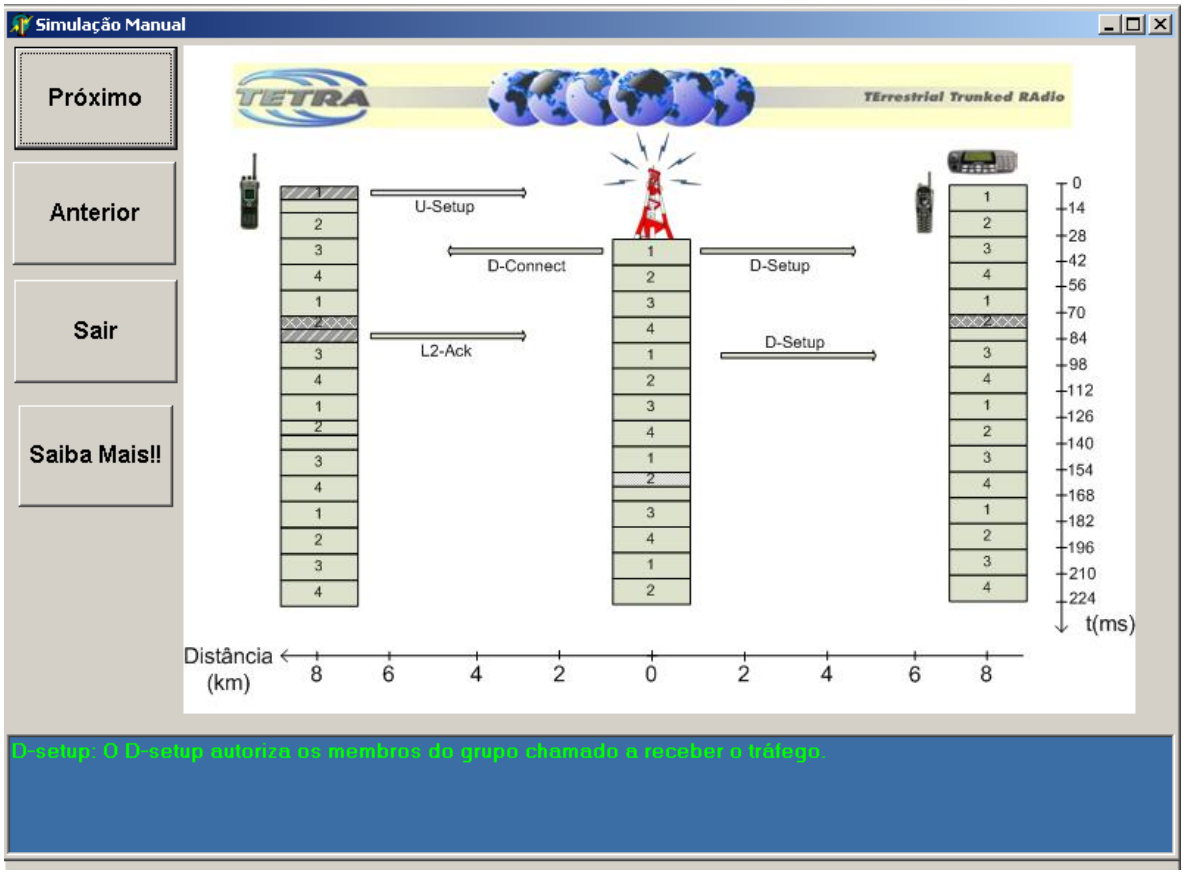


Figura 22: Simulação manual do canal de controle etapa 5

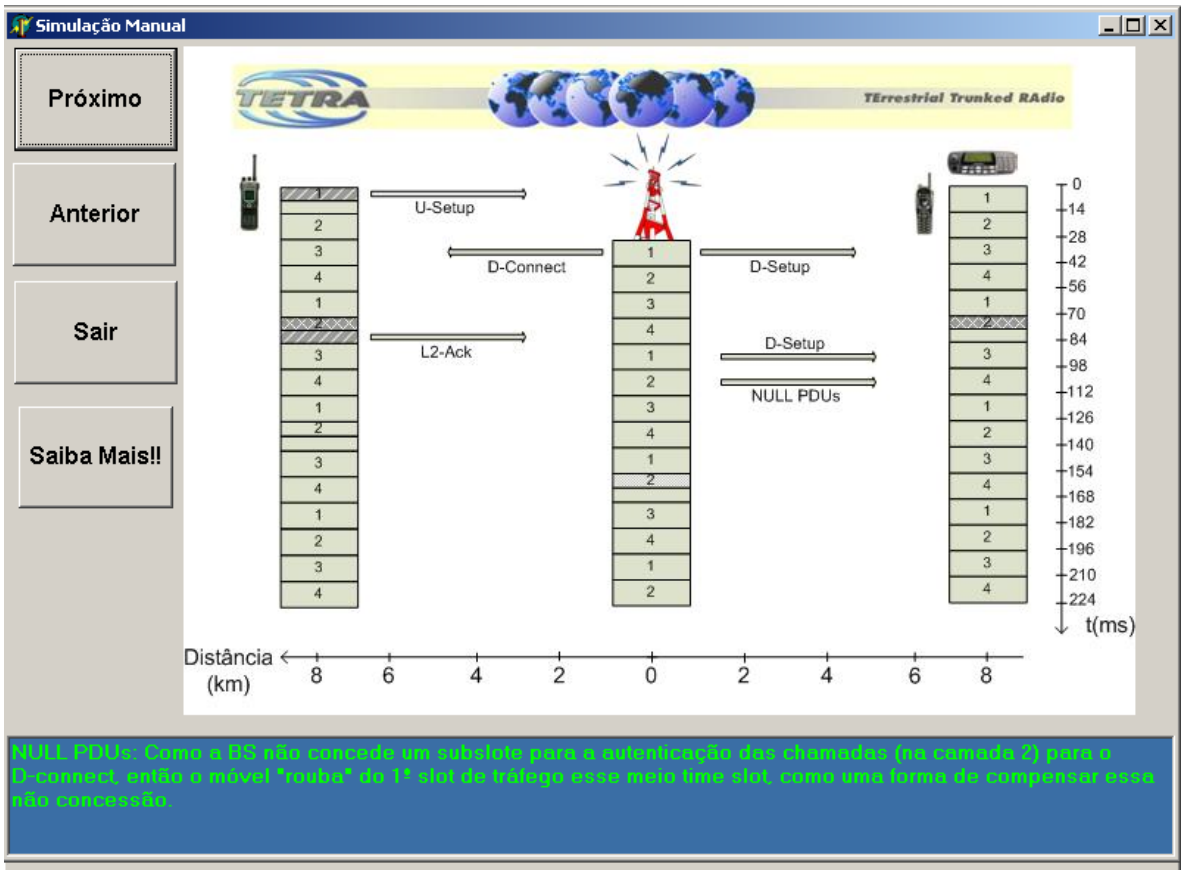


Figura 23: Simulação manual do canal de controle etapa 6

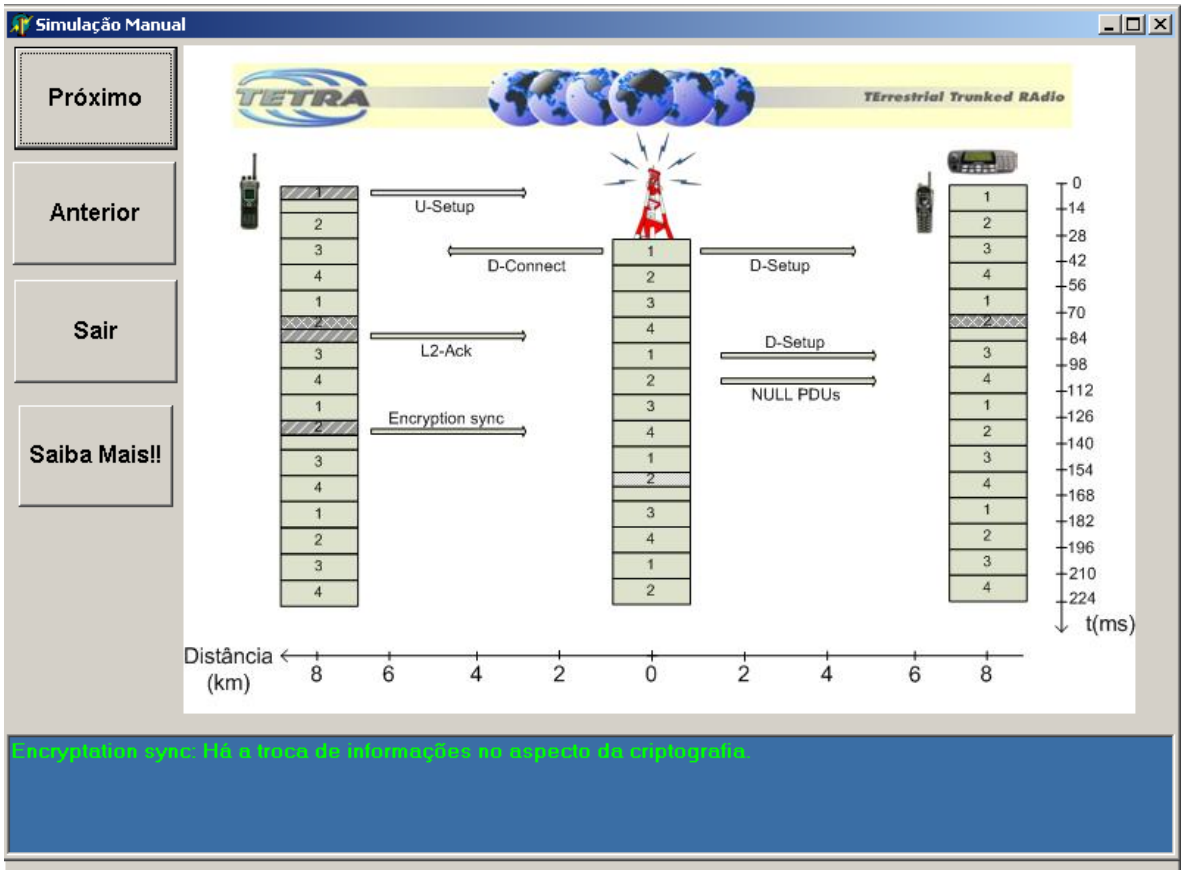


Figura 24: Simulação manual do canal de controle etapa 7

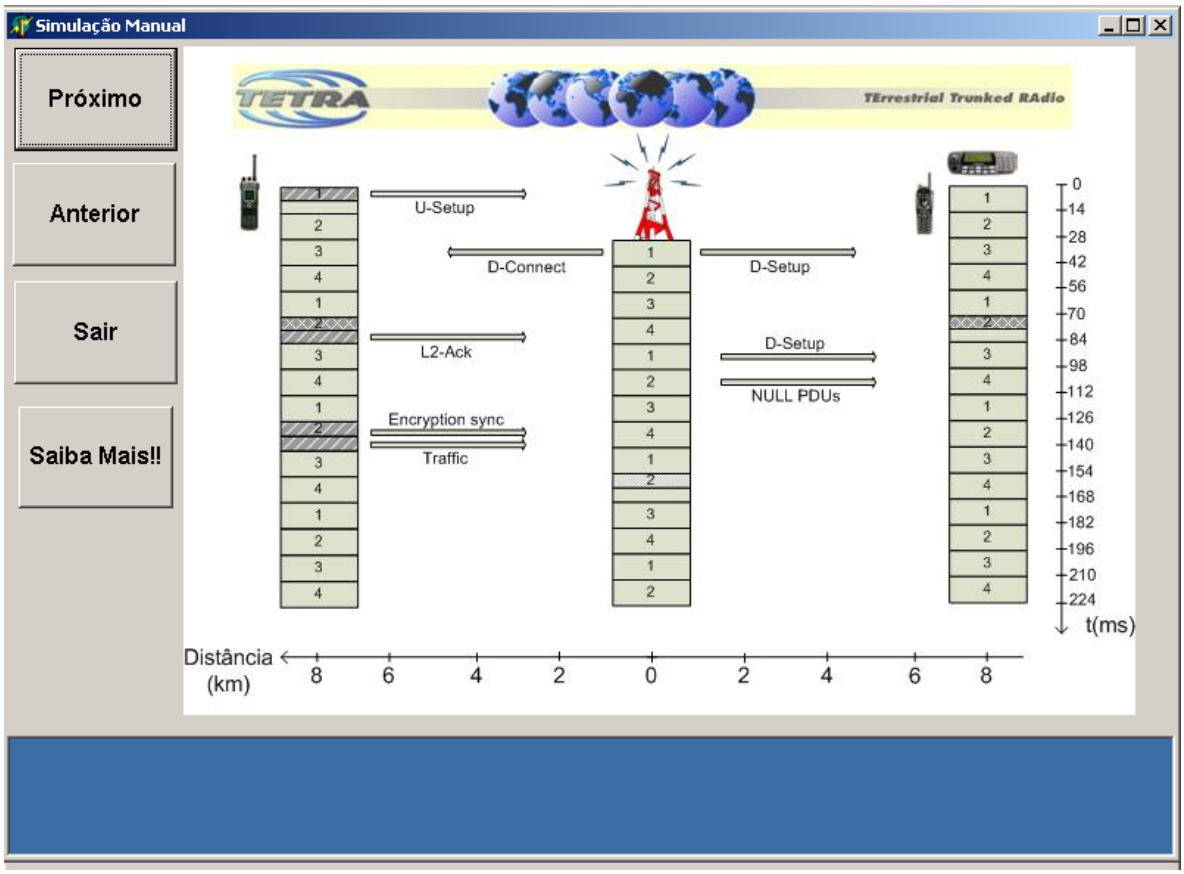


Figura 25: Simulação manual do canal de controle etapa 8

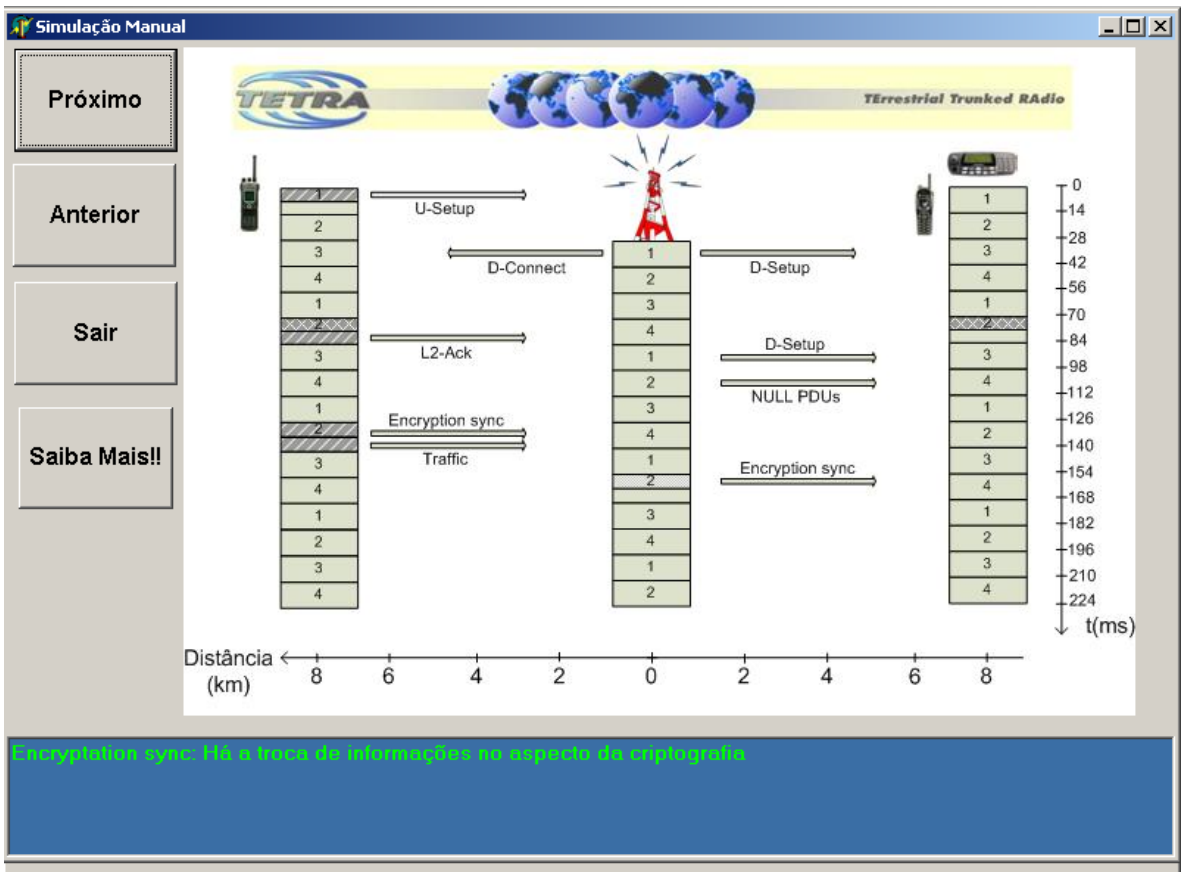


Figura 26 : Simulação manual do canal de controle etapa 9

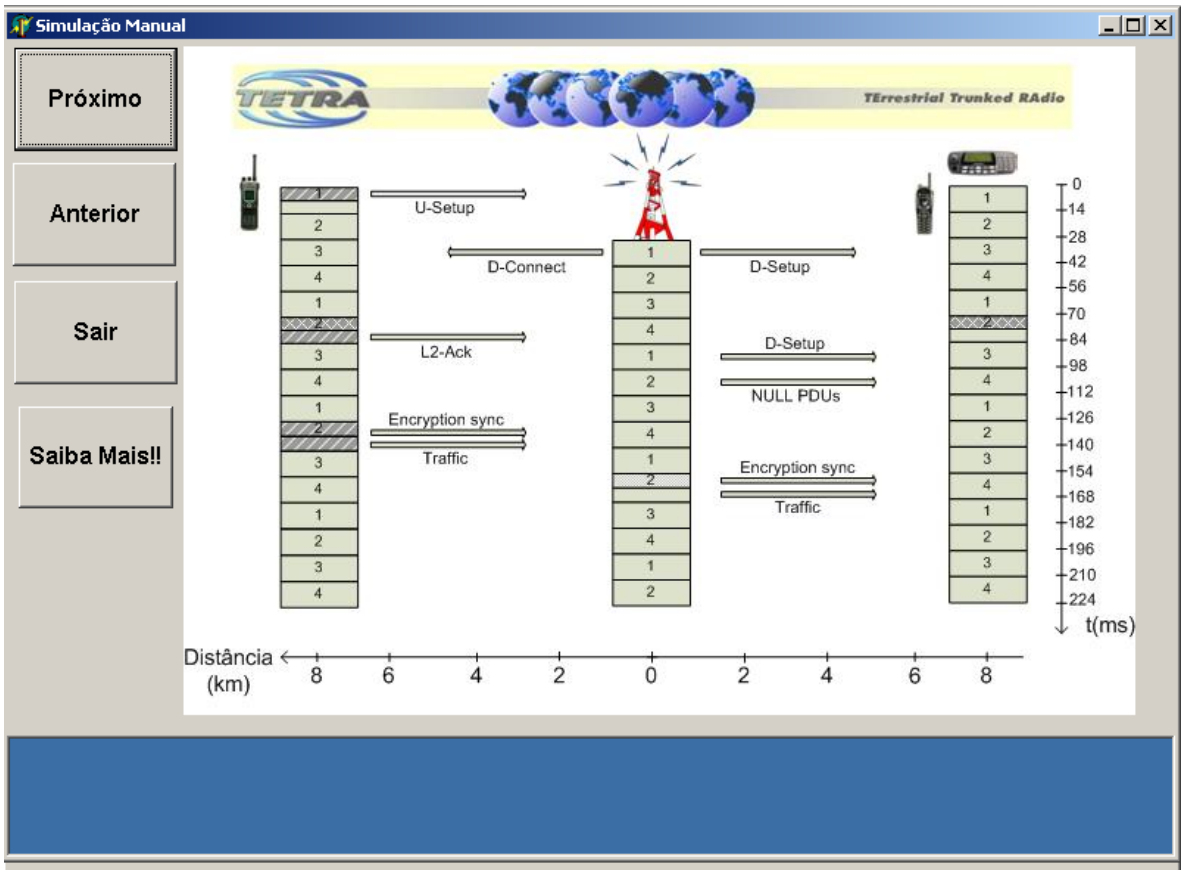


Figura 27: Simulação manual do canal de controle etapa 10

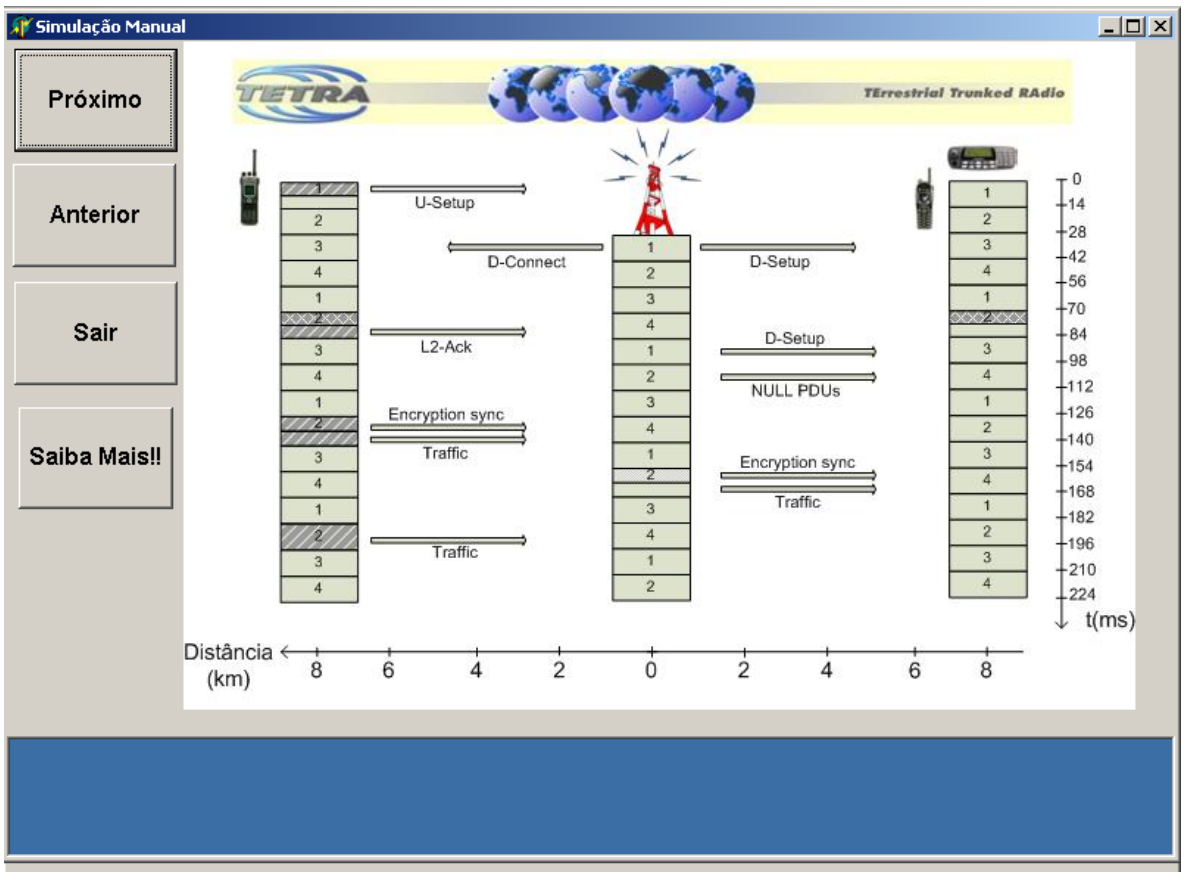


Figura 28: Simulação manual do canal de controle etapa 11

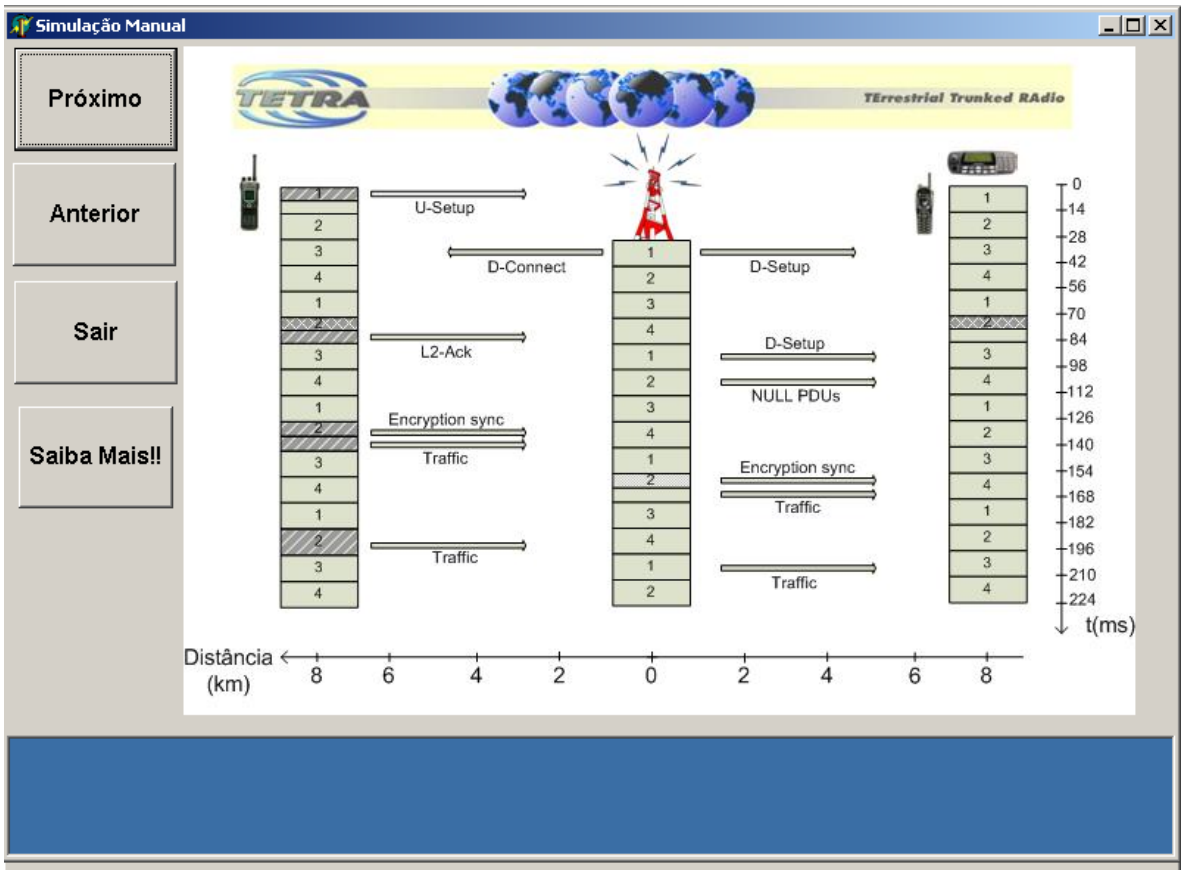


Figura 29: Simulação manual do canal de controle etapa 12

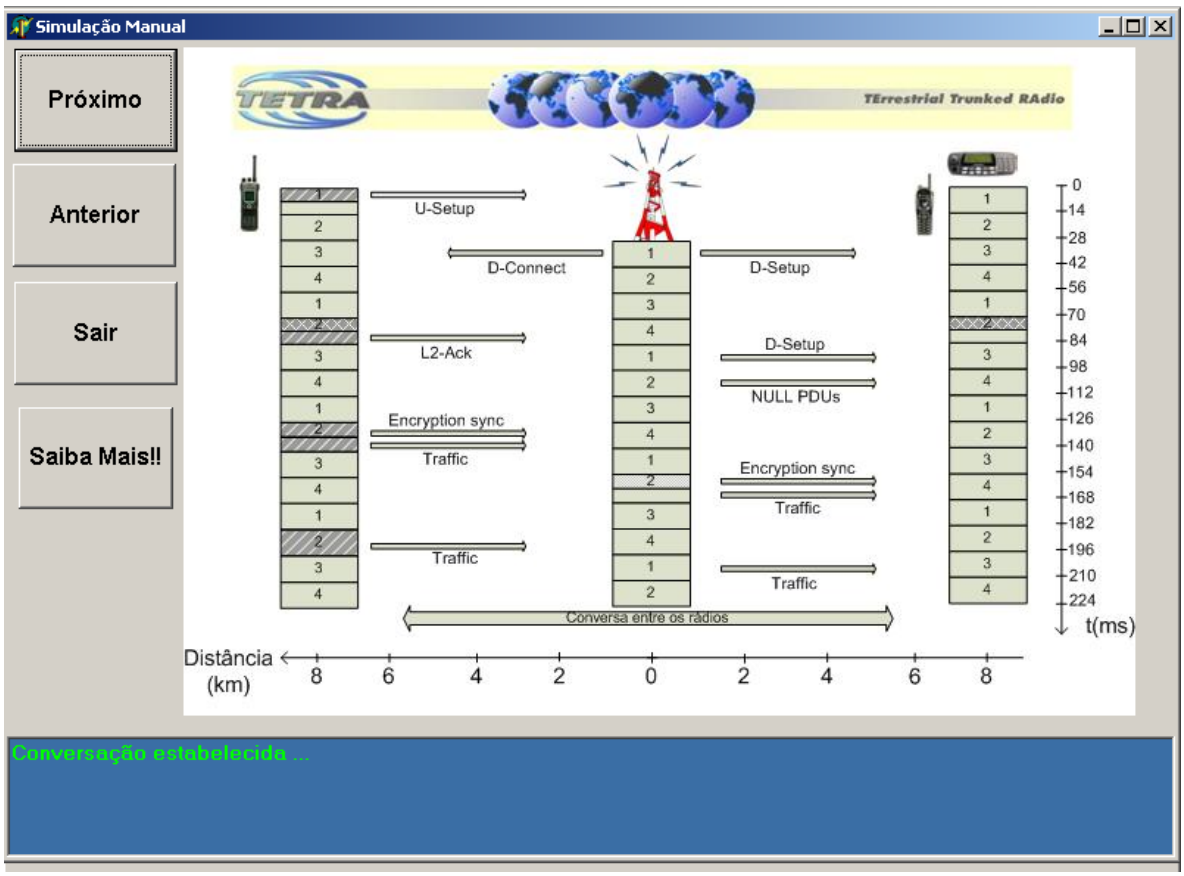


Figura 30 : Simulação manual do canal de controle etapa final. Conversação estabelecida

1 Código em Delphi para implementação do Canal de controle

O comando abaixo é utilizado para chamar o formulário que fará a simulação no "modo automático".

```
procedure Tfrmtetra.Label2Click(Sender: TObject);
begin
    frmSimAut.ShowModal;

end;
```

Para maiores detalhes sobre o código que realmente executa o procedimento do canal de controle.ver Apêndice H

VII. Controle de tráfego

Nessa interface do programa, foi desenvolvida a simulação do canal de controle. Nas figuras que se seguem será ilustrado como o *software* funciona.

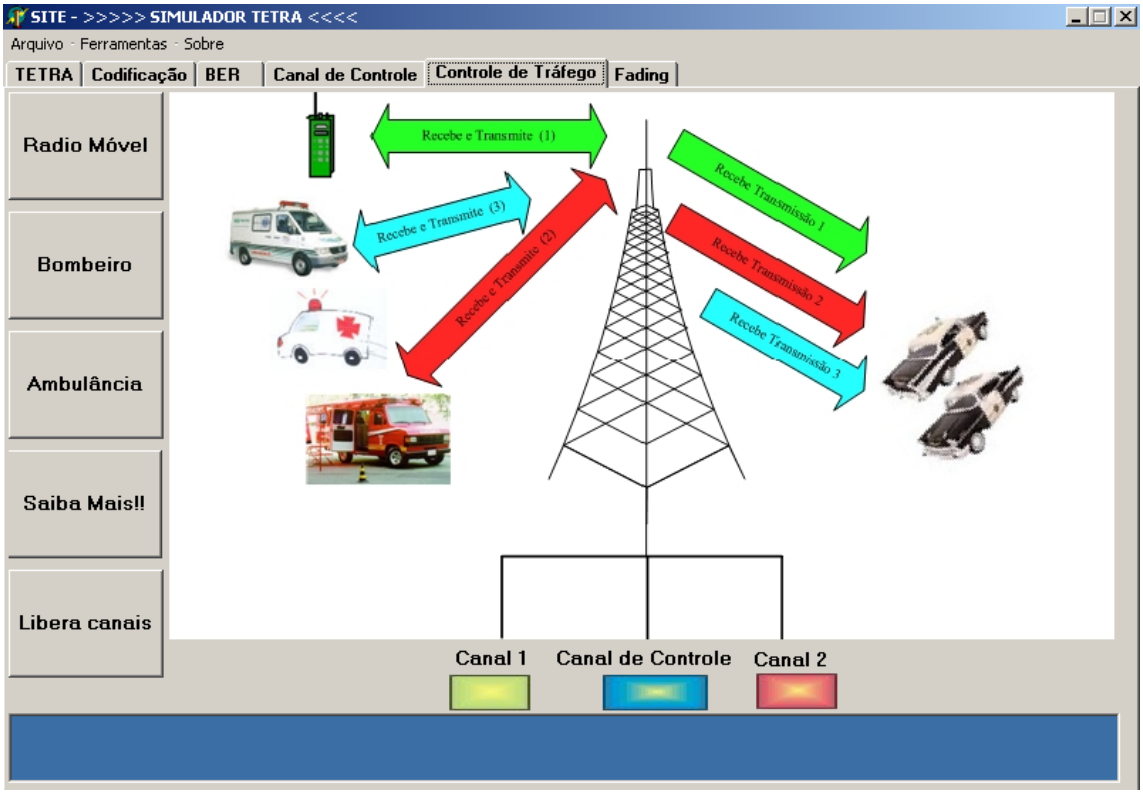


Figura 31: Tela inicial do controle de tráfego

Nessa interface o usuário pode simular a comunicação entre várias entidades de segurança pública.

Ao se pressionar qualquer uma das três opções (Rádio Móvel (policia), Bombeiro ou Ambulância) este inicia a comunicação com o canal de controle, verificando a disponibilidade de uma portadora vaga para assim poder estabelecer a comunicação.

"Ambulância se comunica com o Canal de Controle para solicitar um canal de comunicação livre". Informações do que está ocorrendo são mostradas em uma janela localizada na parte inferior do programa. O exemplo mostrado abaixo foi quando se pressionou o botão "Ambulância". A seguinte sequecia de frases foi mostrada:

"Um Time Slot livre é então escolhido."

"O Canal escolhido foi o 2 e está ocupando o slot número 3."

“Trocando infomações com a ambulância...”

“Trocando infomações com a ambulância... OK”

“Clique em saiba mais para obter mais detalhes sobre o funcionamento do canal de tráfego”.

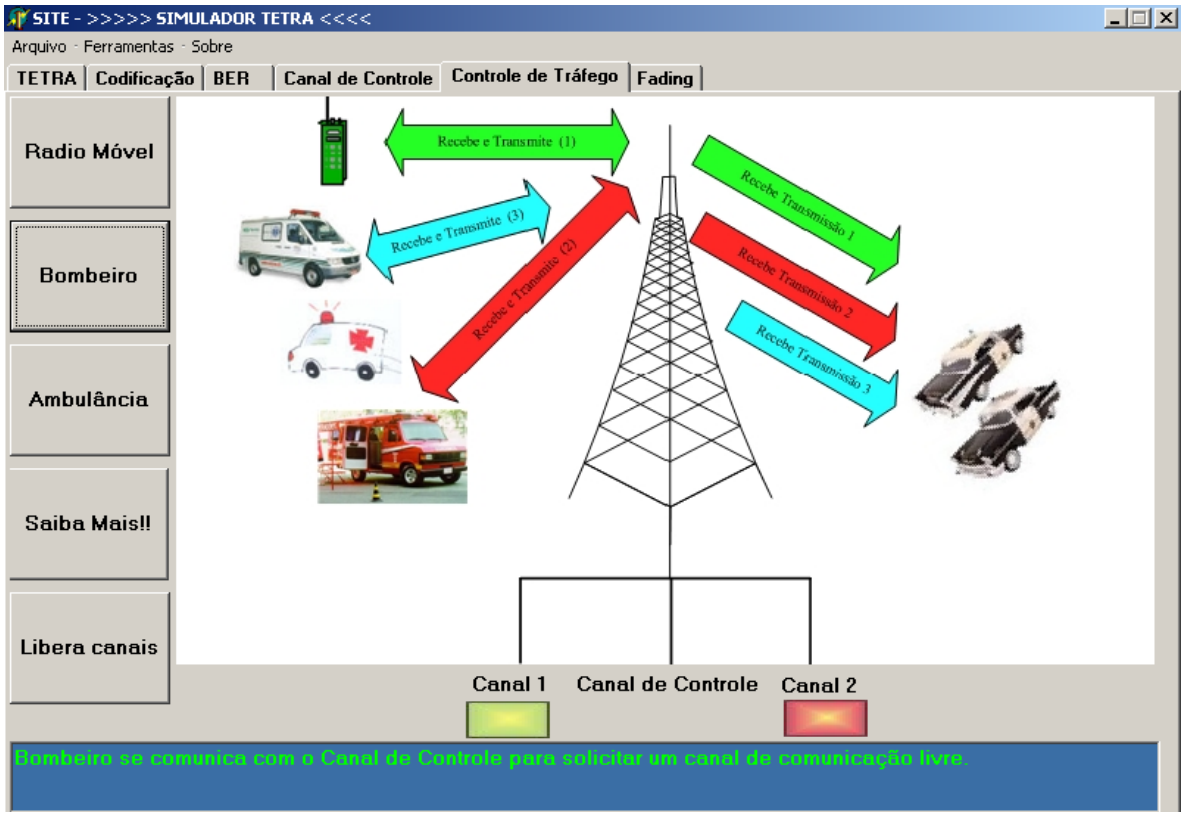


Figura 32: Bombeiro se conectando ao canal de controle

Essas são as mensagens que ocorrem quando o canal acessado não está ocupado. Porém, caso isso não seja verdade então as mensagens são as seguintes:

“Ambulância se comunica com o Canal de Controle para solicitar um canal de comunicação livre.”

“Este TS já está ocupado! Estamos redirecioando para um disponível. Você foi alocado no canal 2 TS número 4”

“O Canal escolhido foi o 2 e está ocupando o slot número 4”

“Trocando infomações com a ambulância ....”

“Trocando infomações com a ambulância .... OK”

“Clique em saiba mais para obter mais detalhes sobre o funcionamento do canal de tráfego.”

Neste caso, ao se tentar acessar um certo *Time Slot* (TS) o programa verifica se aquele TS desejado está ocupado, caso esteja ele tenta redirecionar para algum outro TS livre. Caso todos estejam ocupados, muda-se de canal e tenta-se novamente alocar em algum TS disponível.

Na Figura 33 é mostrado a comunicação com o canal 1 e *o time slot* 2.

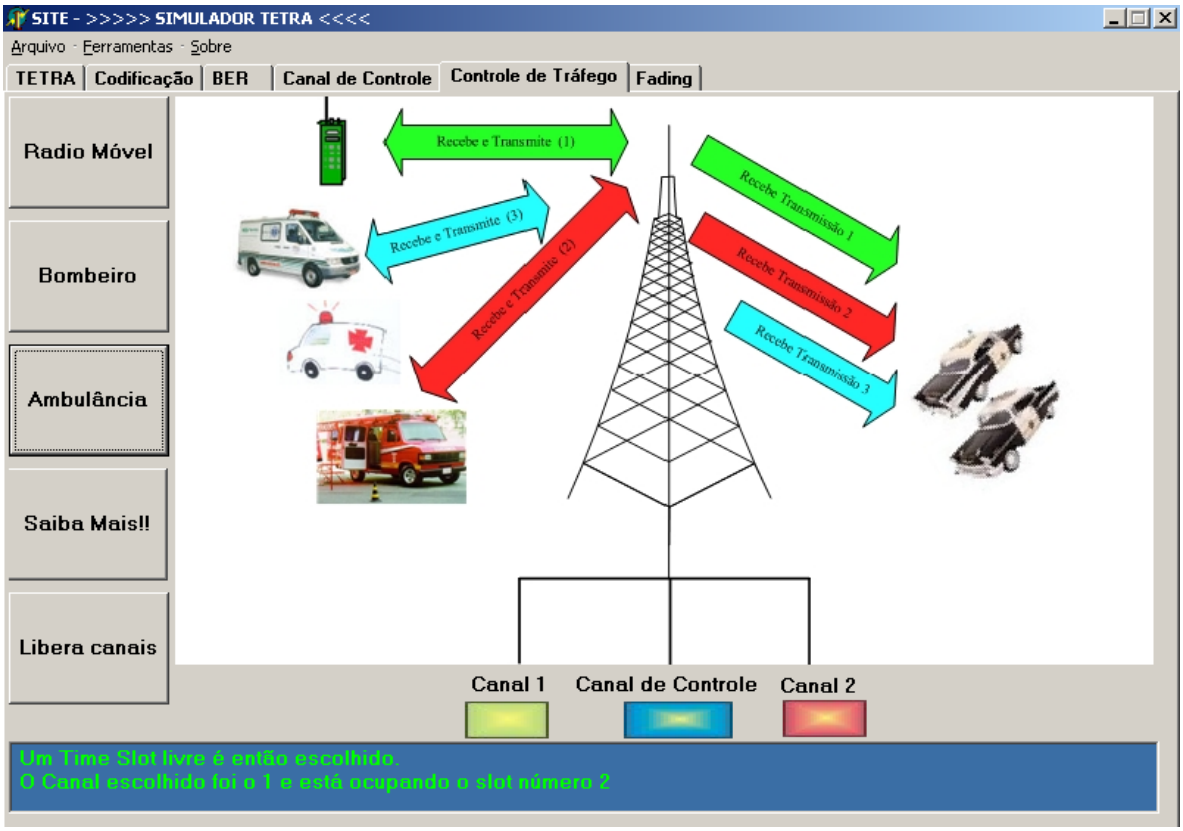


Figura 33: Canal 2 escolhido para estabelecer comunicação

Os quadrados que representam os canais e o canal de controle ficam piscando quando estão sendo solicitados. Na Figura 32 pode-se ver que o canal de controle é que está se atuando, logo em seguida, após a escolha do canal, quem passa a sinalizar é o quadrado do canal escolhido, como mostrado na Figura 33.

O código usado para implementar a ação do botão “Rádio Móvel” do canal de controle se encontra no Apêndice I.

VIII. Fading Rápido

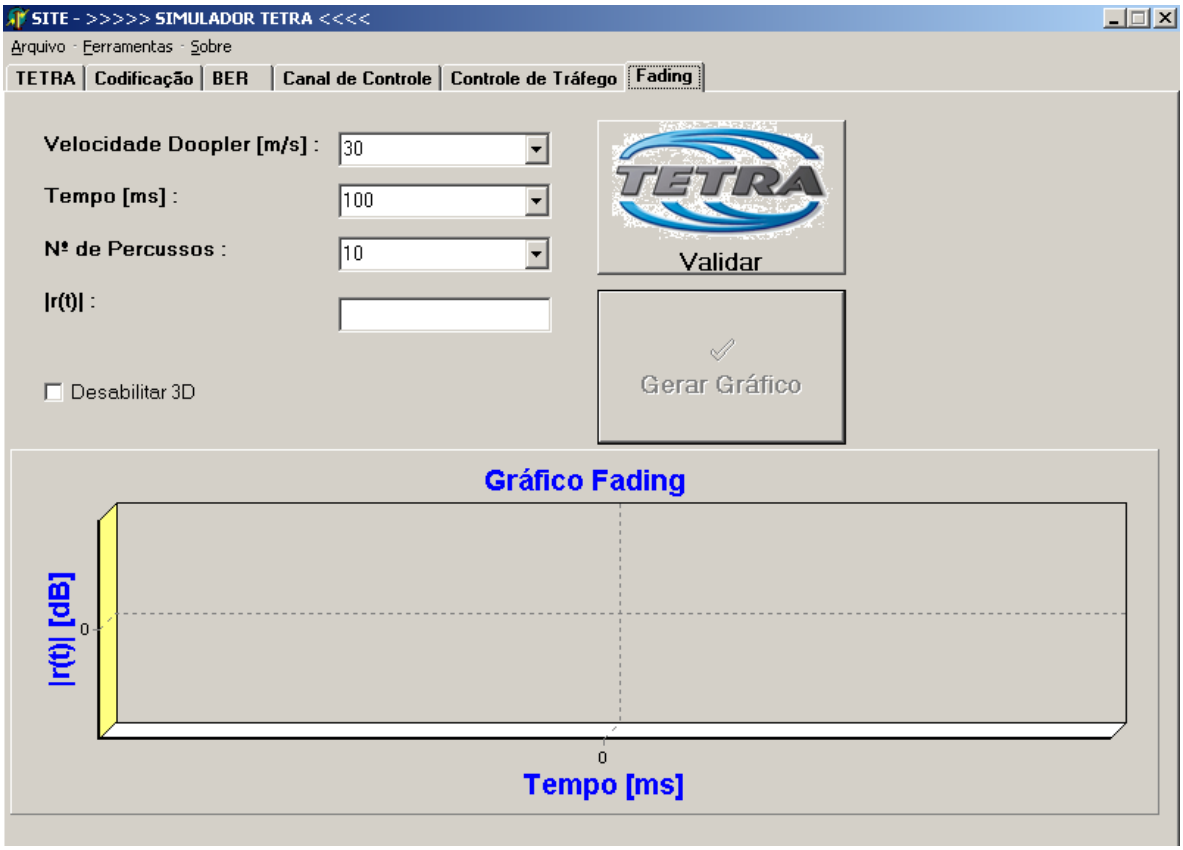


Figura 34: Interface inicial para o *Fading*

Quando um rádio está parado, o sinal recebido é formado pelo vetor soma dos componentes dispersos do sinal transmitido, que chegam à antena por diferentes caminhos e com diferentes amplitudes e fases. Essa soma resulta em um sinal com amplitude constante. Quando um rádio se move, os caminhos dos sinais dispersos mudam, o que é equivalente a introduzir amplitudes e variações de fase aleatórias nos sinais constituintes.

Considere **k** caminhos pelos quais componentes do sinal transmitido são recebidos na antena de um dado rádio. Seja o i-ésimo sinal que chega pelo i-ésimo caminho dado por $a_i e^{j(2\pi ft+\theta_i)}$. Tem-se dessa forma que o sinal resultante é:

$$s_k(t) = \sum_k a_i e^{j(2\pi ft+\theta_i)} = \sum_k a_i \cos(2\pi ft+\theta_i) + j\sum_k a_i sen(2\pi ft+\theta_i) \quad [42]$$

Sendo que f é a freqüência *doppler*, dada por:

$f = \frac{v_D}{2.\pi}$, onde v_D é a velocidade *Doppler*. [43]

O ângulo θ_i é independente e distribuído uniformemente. Já o valor de a_i é a amplitude do i-ésimo sinal. Para fins de simulação, considerou-se $a_i = 1$.

A frequencia Doppler foi obtida da seguinte forma:

Temos que $f = \frac{v}{\lambda}.\cos(\theta)$ **[44]**

Onde v é a velocidade do móvel, θ é o ângulo de desvio da onda e λ é o comprimento de onda, definido por :

$$\lambda = \frac{c}{f_P}. \quad \textbf{[45]}$$

Considere que c seja a velocidade da luz e $f_A = f_P + f$, **[46]**

Onde onde f_A é a freqüência aparente e f_P a freqüência da portadora.

Para a freqüência angular W_D, temos a máxima freqüência Doppler (W_m):

$W_D = W_m.\cos\theta$ **[47]**

Onde $W_m = \beta.v$ **e** **[48]**

$$\beta = \frac{2.\pi}{\lambda} \quad \textbf{[49]}$$

Fazendo as manipulações adequadas chegar-se-á a formula definida na equação [43]

Normalizando o gráfico de r(t) com relação ao valor máximo de r(t) em um dado período (R(t)), e tomando o logaritmo na base 10 multiplicado por 20 desse resultado, tem-se:

$$R(dB) = 20\log(R(t)) \quad \textbf{[50]}$$

$$r(t) = |s_k(t)| = \sqrt{(\sum_k a_i \cos(2\pi ft + \theta_i))^2 + (\sum_k a_i sen(2\pi ft + \theta_i))^2} \quad \textbf{[51]}$$

Como o móvel não está parado existe uma variação na sua fase (ϕ) seja devido aos multipercusos, seja simplesmente pela variação do seu moviemento. Como o movimento não é constante, adotamos essa mudança de fase de forma aleatória. O valor de ϕ é definido pelo seguinte intervalo:

$0 < \phi < 1$ radianos

Então substituindo a equação 51 pela equação 52, encontramos a equação utilizada para os cálculos no simulador:

$$r(t) = |\, s_k(t) \,| = \sqrt{\left(\sum_k a_i \cos(\phi_i - 2\pi f t.\cos(\theta_i))\right)^2 + \left(\sum_k a_i sen(\varphi_i - 2\pi f t.\cos(\theta_i))\right)^2} \quad [52]$$

Vide Figura 35 como ilustração do ângulo.

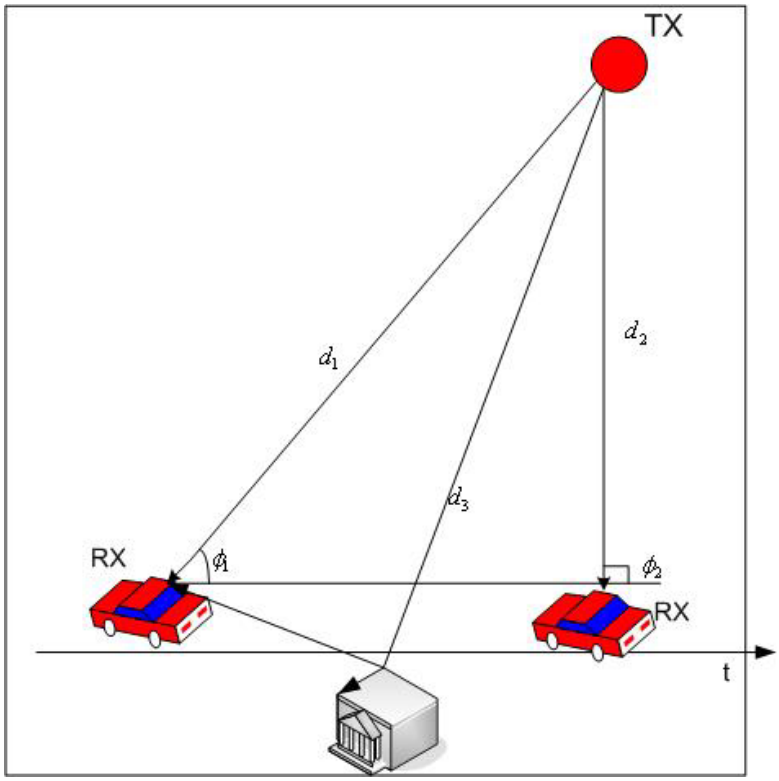


Figura 35:Ilustração Angulo de fase

Este resultado descreve o desvanecimento do sinal em um dado instante de tempo (t). O gráfico obtido em excel para k=10 e velocidade de 100 km/h para um dado rádio é o seguinte:

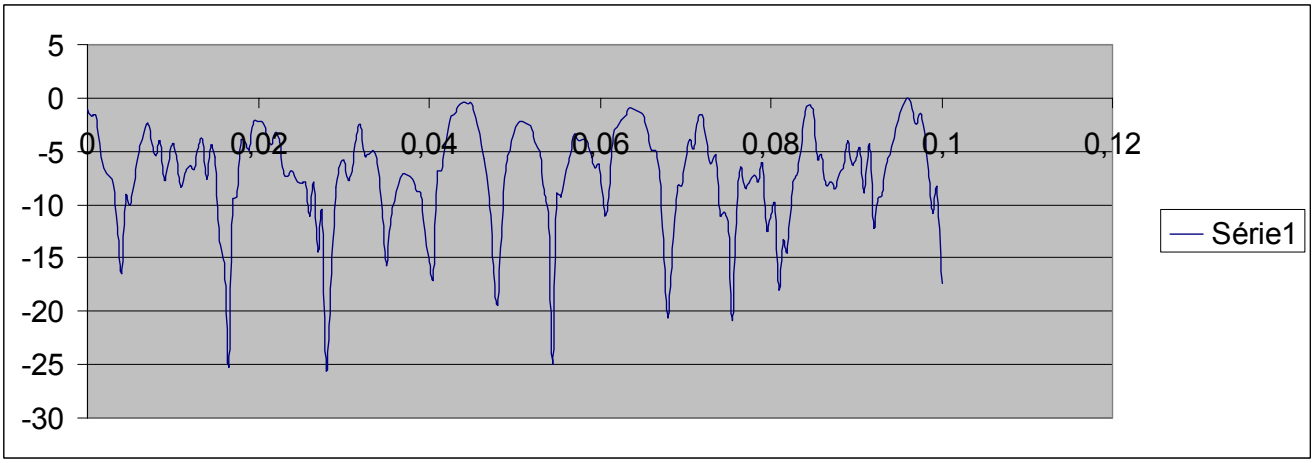


Figura 36: Fading para 10 caminhos gerada pelo Excel

Os resultados no SITE indicam que para um número pequeno de caminhos o desvanecimento do sinal (*fading*) é pequeno, como era esperado.

Fazendo a comparação entre o software SITE e o Excel , abaixo serão mostados os resultados obtidos.

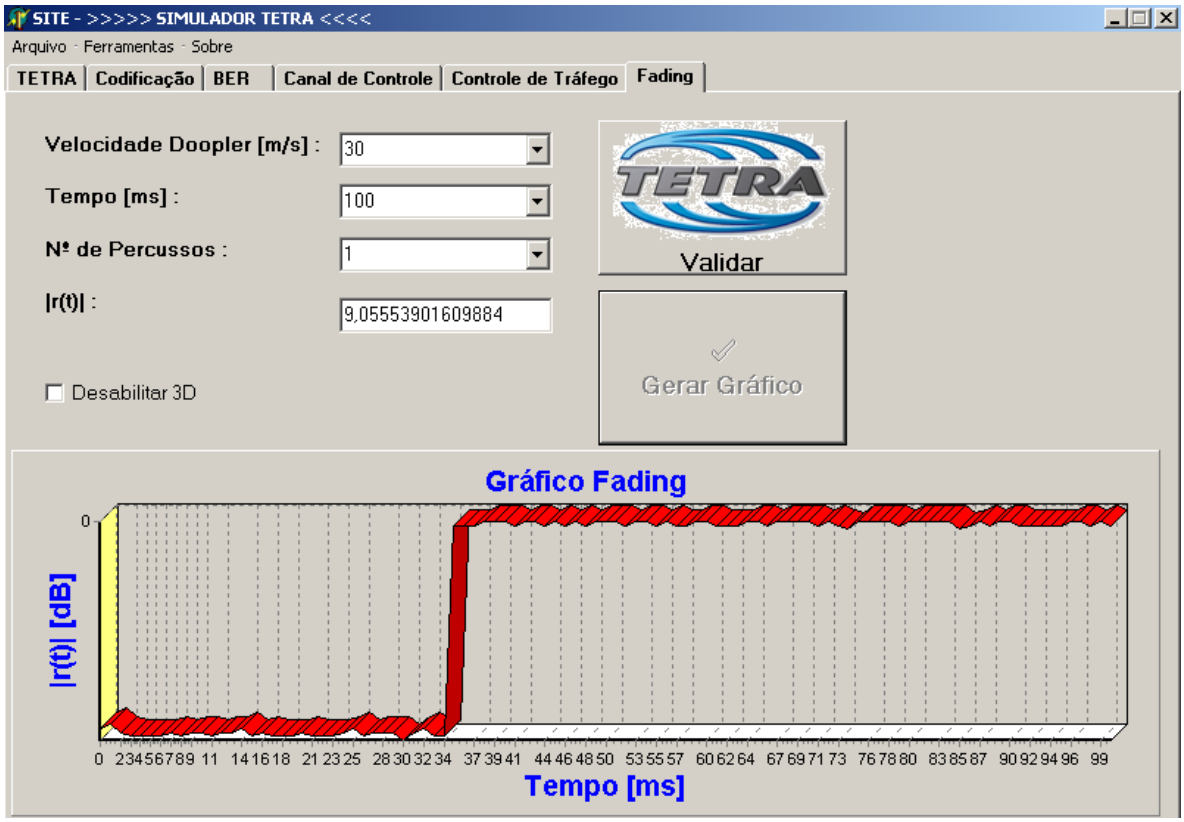


Figura 37: Fading para 1 caminho

Nesse caso o *fading* foi igual a zero. O que já era esperado.

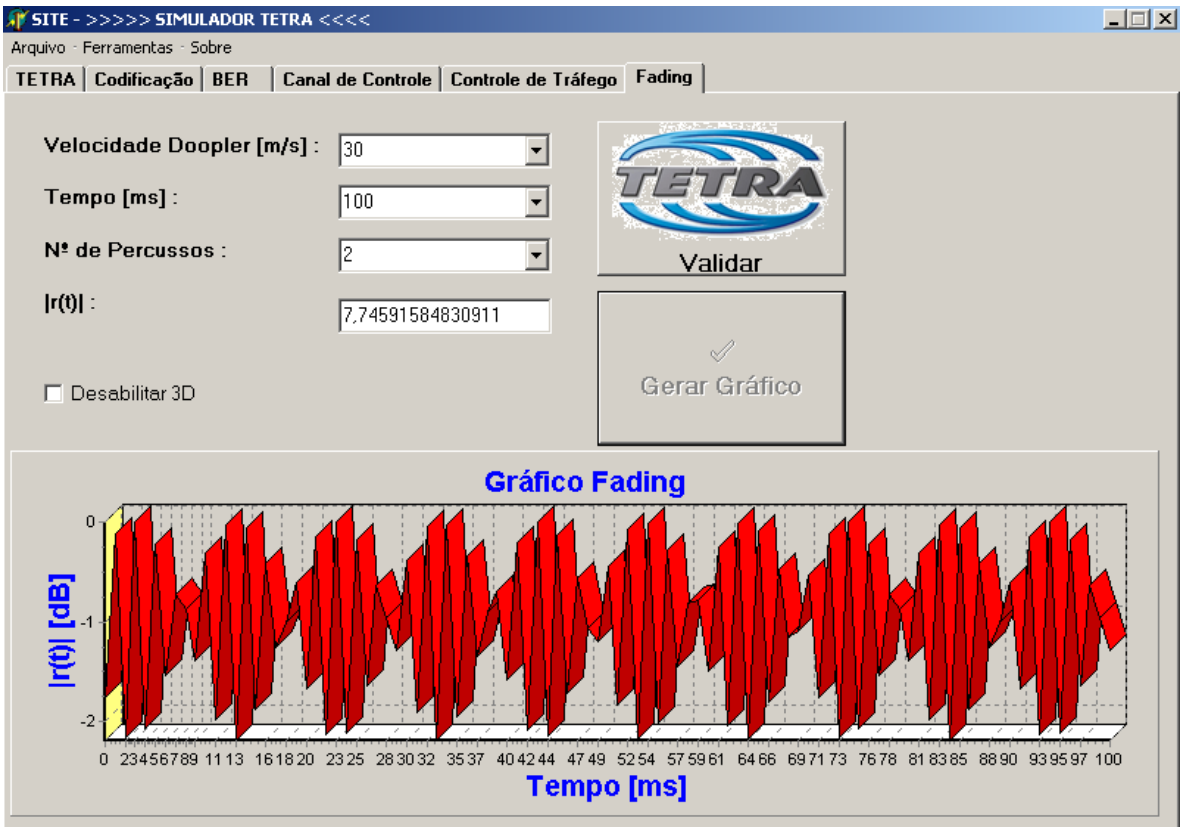


Figura 38: Fading para 2 caminhos

Usando uma facilidade oferecida pelo software, foi feito um zoom e verificado que para dois caminhos há uma perda de aproximadamente 2,45 dB.

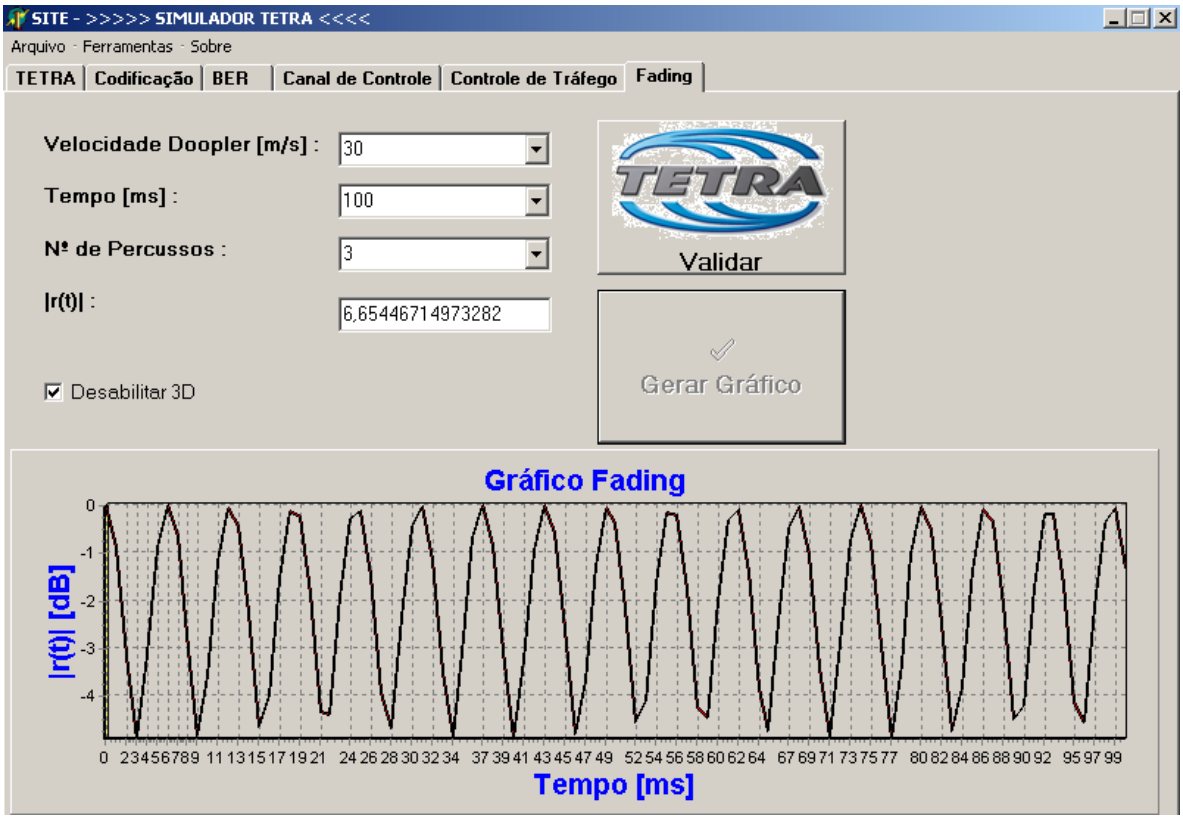


Figura 39: Fading para 3 caminhos

No caso acima, mostrado pela Figura 39, foi desabilitado a opção de 3D e foi utilizado 3 caminhos. O desvanecimento obtido foi de aproximadamente 4 dB.

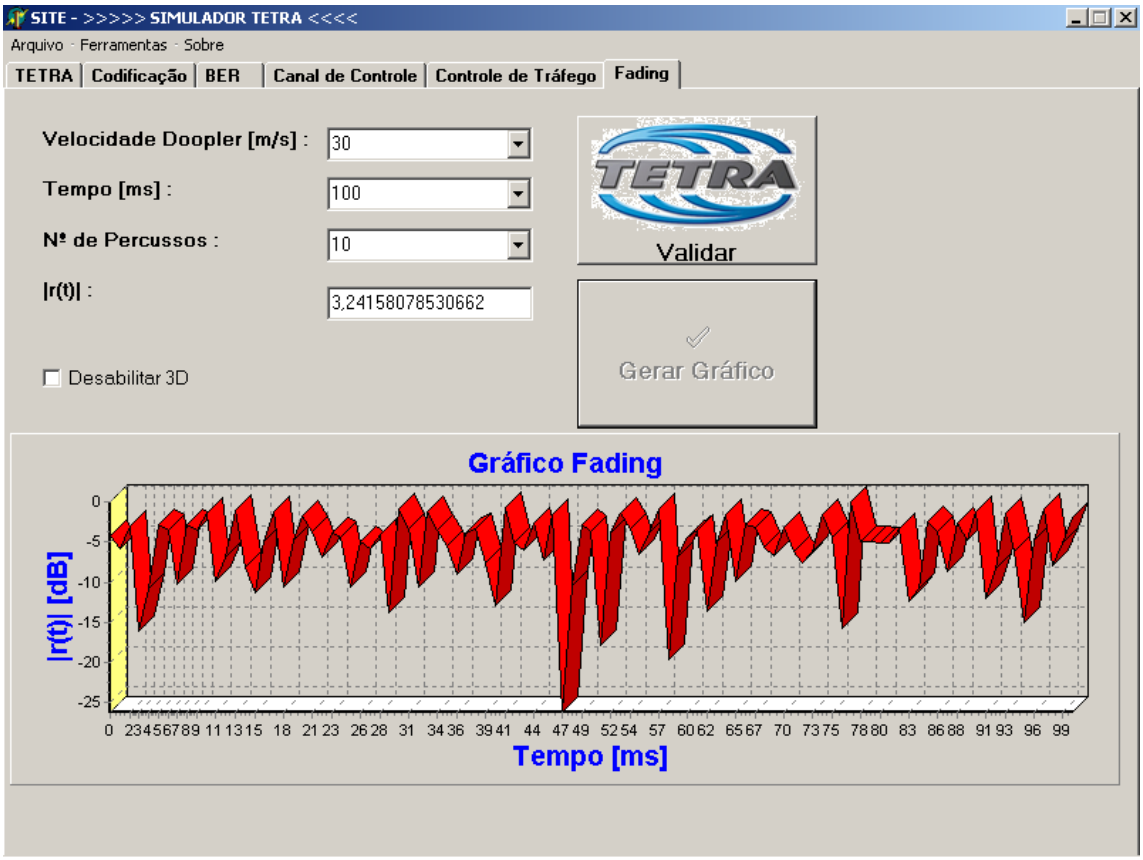


Figura 40: Fading para 10 caminhos

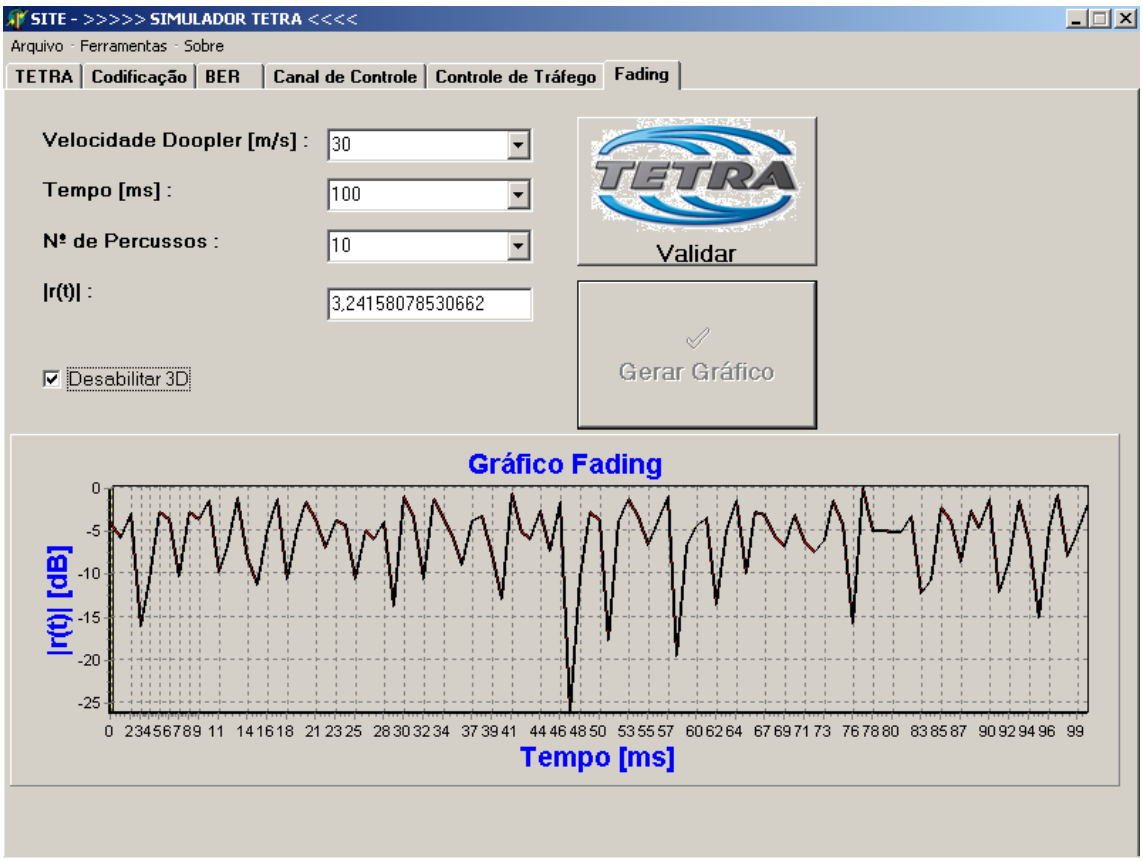


Figura 41: Fading para 10 caminhos com 3D desabilitado

Pode-se observar a semelhança entre a Figura 41, que foi gerada pelo software SITE e a Figura 36 gerada pelo Excel. O valor obtido para o *fading* foi de aproximadamente 26 dB.

Para maiores detalhes sobre o software o leitor deverá consultar o Apêndice J.

IX. Conclusão

Através dos estudos realizados para a implementação desta interface, foi possível observar que além de variáveis como BER, fading, potência de transmissão, taxa de transmissão e controle de trafego que foram algumas das variáveis abordadas nesse trabalho, não podemos esquecer que para casos práticos, devemos ainda levar em consideração outras variáveis tais como aproveitamento do legado, custo para implementação do sistema, valor dos rádios móveis, disponibilidade de recurso financeiro para implementação das ferramentas (serviços) disponibilizadas pelo padrão adotado são fatores muito importantes na hora de se avaliar qual padrão de segurança pública adotar.

Foi possível ver a aplicação do modelo de Okumura Hata para pequenas e grandes cidades verificando a diferença na propagação do sinal, bem como as vantagens oferecidas por sistemas críticos e troncalizados que são mais robustos e estáveis.

Além disso, constatamos, através de estudos prévios para elaboração deste trabalho, a vantagem de um sistema TDM, quando comparado a um FDM quando comparamos o tráfego, ou seja, sistemas TDM utilizam o espectro de forma mais otimizada.

A relevância do processo de desvainecimento do sinal foi um item que foi abordado com êxito neste estudo, mostrando, em caso especifico para o TETRA, a importância de se ter uma comunicação que além de existir tem que ser inteligível.

Como sugestão para trabalhos futuros a plataforma de simulação poderia conter uma ferramenta capaz de dimensionar um sistema TETRA dada a necessidade dos órgãos de segurança.

Apêndice

Apêndice A: Código em Delphi para concatenação dos frames A e B e reposicionamento dos bits e geração da matriz geradora do código CRC

```
    //sensibilidade 0
M[1]:=A[35];  M[2]:=B[35];  M[3]:=A[36];  M[4]:=B[36];  M[5]:=A[37];  M[6]:=B[37];
M[7]:=A[38];
M[8]:=B[38];  M[9]:=A[39];  M[10]:=B[39];  M[11]:=A[40];  M[12]:=B[40];  M[13]:=A[41];
M[14]:=B[41];  M[15]:=A[42];  M[16]:=B[42];  M[17]:=A[43];  M[18]:=B[43];  M[19]:=A[47];
M[20]:=B[47];  M[21]:=A[48];  M[22]:=B[48];  M[23]:=A[56];  M[24]:=B[56];  M[25]:=A[61];
M[26]:=B[61];  M[27]:=A[62];  M[28]:=B[62];  M[29]:=A[63];  M[30]:=B[63];  M[31]:=A[64];
M[32]:=B[64];  M[33]:=A[65];  M[34]:=B[65];  M[35]:=A[66];  M[36]:=B[66];  M[37]:=A[67];
M[38]:=B[67];  M[39]:=A[68];  M[40]:=B[68];  M[41]:=A[69];  M[42]:=B[69];  M[43]:=A[70];
M[44]:=B[70];  M[45]:=A[74];  M[46]:=B[74];  M[47]:=A[75];  M[48]:=B[75];  M[49]:=A[83];
M[50]:=B[83];  M[51]:=A[88];  M[52]:=B[88];  M[53]:=A[89];  M[54]:=B[89];  M[55]:=A[90];
M[56]:=B[90];  M[57]:=A[91];  M[58]:=B[91];  M[59]:=A[92];  M[60]:=B[92];  M[61]:=A[93];
M[62]:=B[93];  M[63]:=A[94];  M[64]:=B[94];  M[65]:=A[95];  M[66]:=B[95];  M[67]:=A[96];
M[68]:=B[96];  M[69]:=A[97];  M[70]:=B[97];  M[71]:=A[101];  M[72]:=B[101];  M[73]:=A[102];
M[74]:=B[102];  M[75]:=A[110];  M[76]:=B[110];  M[77]:=A[115];  M[78]:=B[115];
M[79]:=A[116];
M[80]:=B[116];  M[81]:=A[117];  M[82]:=B[117];  M[83]:=A[118];  M[84]:=B[118];
M[85]:=A[119];
M[86]:=B[119];  M[87]:=A[120];  M[88]:=B[120];  M[89]:=A[121];  M[90]:=B[121];
M[91]:=A[122];
M[92]:=B[122];  M[93]:=A[123];  M[94]:=B[123];  M[95]:=A[124];  M[96]:=B[124];
M[97]:=A[128];
M[98]:=B[128];  M[99]:=A[129];  M[100]:=B[129];  M[101]:=A[137];  M[102]:=B[137];

    //sensibilidade 1
M[103]:=A[58];  M[104]:=B[58];  M[105]:=A[85];  M[106]:=B[85];  M[107]:=A[112];
M[108]:=B[112];  M[109]:=A[54];
M[110]:=B[54];  M[111]:=A[81];  M[112]:=B[81];  M[113]:=A[108];  M[114]:=B[108];
M[115]:=A[135];  M[116]:=B[135];
```

M[117]:=A[50]; M[118]:=B[50]; M[119]:=A[77]; M[120]:=B[77]; M[121]:=A[104]; M[122]:=B[104]; M[123]:=A[131];
M[124]:=B[131]; M[125]:=A[45]; M[126]:=B[45]; M[127]:=A[72]; M[128]:=B[72]; M[129]:=A[99]; M[130]:=B[99];
M[131]:=A[126]; M[132]:=B[126]; M[133]:=A[55]; M[134]:=B[55]; M[135]:=A[82]; M[136]:=B[82]; M[137]:=A[109];
M[138]:=B[109]; M[139]:=A[136]; M[140]:=B[136]; M[141]:=A[5]; M[142]:=B[5]; M[143]:=A[13]; M[144]:=B[13];
M[145]:=A[34]; M[146]:=B[34]; M[147]:=A[8]; M[148]:=B[8]; M[149]:=A[16]; M[150]:=B[16]; M[151]:=A[17];
M[152]:=B[17]; M[153]:=A[22]; M[154]:=B[22]; M[155]:=A[23]; M[156]:=B[23]; M[157]:=A[24]; M[158]:=B[24];
M[159]:=A[25]; M[160]:=B[25]; M[161]:=A[26]; M[162]:=B[26]; M[163]:=A[6]; M[164]:=B[6]; M[165]:=A[14];
M[166]:=B[14]; M[167]:=A[7]; M[168]:=B[7]; M[169]:=A[15]; M[170]:=B[15]; M[171]:=A[60]; M[172]:=B[60];
M[173]:=A[87]; M[174]:=B[87]; M[175]:=A[114]; M[176]:=B[114]; M[177]:=A[46]; M[178]:=B[46]; M[179]:=A[73];
M[180]:=B[73]; M[181]:=A[100]; M[182]:=B[100]; M[183]:=A[127]; M[184]:=B[127]; M[185]:=A[44]; M[186]:=B[44];
M[187]:=A[71]; M[188]:=B[71]; M[189]:=A[98]; M[190]:=B[98]; M[191]:=A[125]; M[192]:=B[125]; M[193]:=A[33];
M[194]:=B[33]; M[195]:=A[49]; M[196]:=B[49]; M[197]:=A[76]; M[198]:=B[76]; M[199]:=A[103]; M[200]:=B[103];
M[201]:=A[130]; M[202]:=B[130]; M[203]:=A[59]; M[204]:=B[59]; M[205]:=A[86]; M[206]:=B[86]; M[207]:=A[113];
M[208]:=B[113]; M[209]:=A[57]; M[210]:=B[57]; M[211]:=A[84]; M[212]:=B[84]; M[213]:=A[111]; M[214]:=B[111];

//sensibilidade 2

M[215]:=A[18]; M[216]:=B[18]; M[217]:=A[19]; M[218]:=B[19]; M[219]:=A[20]; M[220]:=B[20]; M[221]:=A[21];
M[222]:=B[21]; M[223]:=A[31]; M[224]:=B[31]; M[225]:=A[32]; M[226]:=B[32]; M[227]:=A[53]; M[228]:=B[53];

M[229]:=A[80]; M[230]:=B[80]; M[231]:=A[107]; M[232]:=B[107]; M[233]:=A[134]; M[234]:=B[134]; M[235]:=A[1];

M[236]:=B[1]; M[237]:=A[2]; M[238]:=B[2]; M[239]:=A[3]; M[240]:=B[3]; M[241]:=A[4]; M[242]:=B[4];

M[243]:=A[9]; M[244]:=B[9]; M[245]:=A[10]; M[246]:=B[10]; M[247]:=A[11]; M[248]:=B[11]; M[249]:=A[12];

M[250]:=B[12]; M[251]:=A[27]; M[252]:=B[27]; M[253]:=A[28]; M[254]:=B[28]; M[255]:=A[29]; M[256]:=B[29];

M[257]:=A[30]; M[258]:=B[30]; M[259]:=A[52]; M[260]:=B[52]; M[261]:=A[79]; M[262]:=B[79]; M[263]:=A[106];

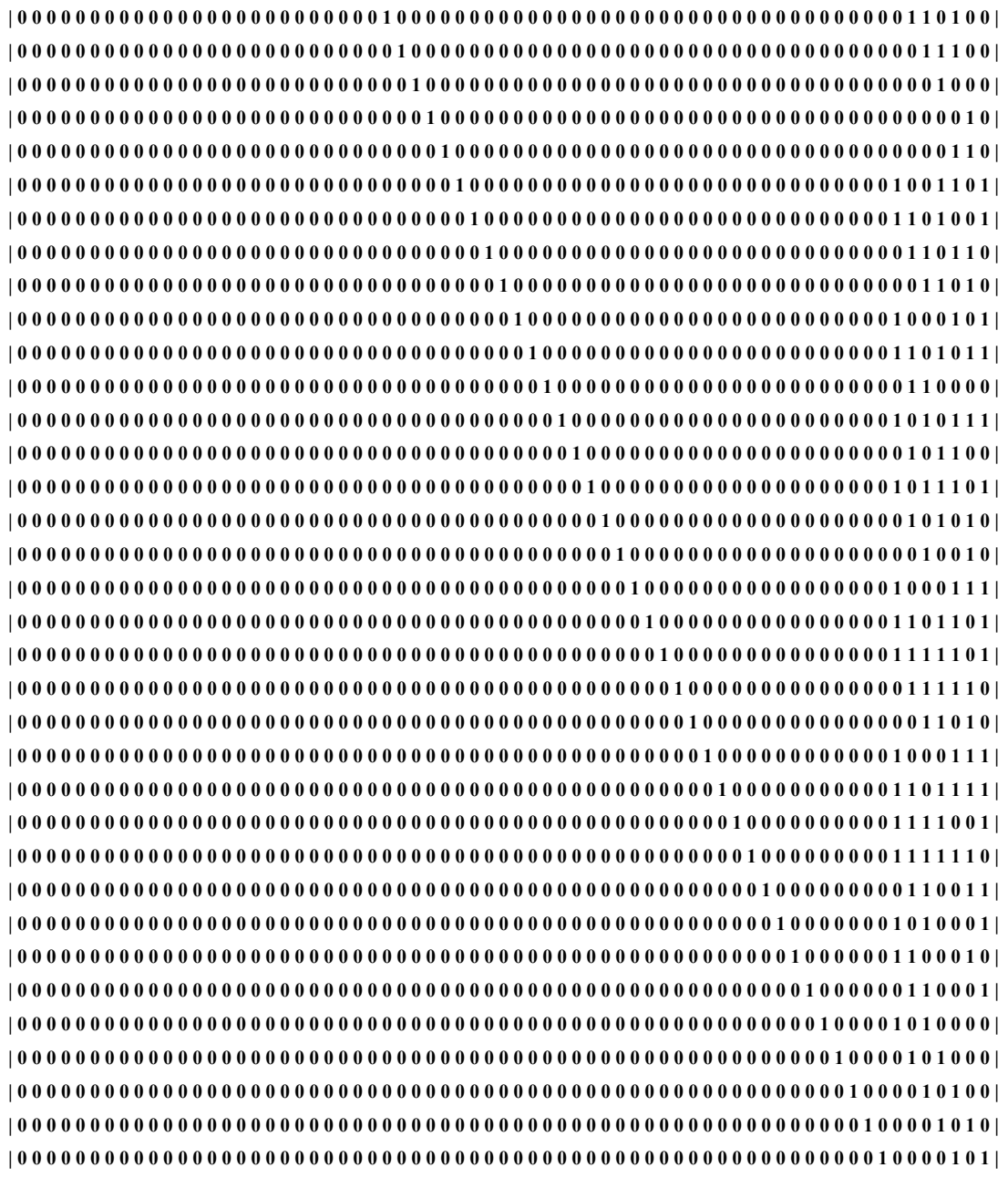
M[264]:=B[106]; M[265]:=A[133]; M[266]:=B[133]; M[267]:=A[51]; M[268]:=B[51]; M[269]:=A[78]; M[270]:=B[78];

M[271]:=A[105]; M[272]:=B[105]; M[273]:=A[132]; M[274]:=B[132];

Apêndice B: Matriz Geradora do código CRC

Matriz Geradora do Código CRC

```
|1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0|
|0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0|
|0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0|
|0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0|
|0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1|
|0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0|
|0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0|
|0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1|
|0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1|
|0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0|
|0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0|
|0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0|
|0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 0|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1|
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1|
```



Apêndice C: Implementação em delphi para codificação

```
begin
//------------------------------BITS de classe 0 ----------------------------------------------------
    edtclass0.Color:=clWhite; // muda a cor do edit de cinza para branco
    For i:=1 TO 102 DO
    edtclass0.Text:=edtclass0.Text + inttostr(rcpc[i]);  // preenche o edit com os bits de concatenação de 1
a 102.
//------------------------------Fim Classe 0 ----------------------------------------------------
//------------------------------BITS de classe 1 ----------------------------------------------------
    edtclass1.Color:=clWhite;
```

```
    For i:=1 TO 112 DO
    edtclass1.Text:=edtclass1.Text + inttostr(msg1[i]);  // preenche o edit com os bits de concatenação de 103 a  214.
//------------------------- Fim Classe 1-----------------------------------------------------------
//-----------------------------BITS de classe 2 ------------------------------------------------------
    edtclass2.Color:=clWhite;
    For i:=1 to 60 do
    edtclass2.text:=edtclass2.Text + inttostr(msg2[i]);  // preenche o edit com os bits de concatenação de 215 a 286. 60  gerados + 8 bits de paridade + 4 tail bits.
//------------------------------FIM CLASSE 2-----------------------------------------------------------
//------------------------------CRC --------------------------------------------------------------------
    edtcrc.Color:=clWhite;
    for i:= 1 to 60 do
    edtcrc.Text:= edtcrc.Text + inttostr(msg2[i]); // Preenche o edit com os 60 bits provinientes dos bits de sensibilidade 2.
    for i:= 1 to 8 do
    edtcrc.Text:= edtcrc.Text + inttostr(m[i]);//  Preenche o edit com os 60 bits provenientes dos bits de sensibilidade 2 + 8 bits de paridade (m[1],m[2],...,m[8]).
//----------------------------- FIM CRC ----------------------------------------------------------
//GERANDO OS 4 TAIL BITS (Que são = 0)
      for i:=1 to 4 do
      begin
        sgRCPC.Cells[i+281,0]:=inttostr(i+282)+ ':  ' + inttostr(0);
        rcpc[i+282]:=0; //matriz para ser utilizada no rcpc (usar nas sensibilidades)
      end;
      for i:= 1 to 286 do
      begin
         edtRCPC2.Text:=edtRCPC2.Text + inttostr(rcpc[i]); // Preenche o edit RCPC 2.
      end;
//---------------------COMPLETANDO OS ULTIMOS 4 BITS PARA A MATRIZ msg2 -----------------
     For i:=1 to 4 do
     msg2[i+68]:= rcpc[i+282];
//--------------------------------Definindo valores para as variáveis--------------------------------
//declaração dos coeficientes das matrizes geradoras
      g10:=1;
```

```
        g11:=1;
        g12:=1;
        g13:=1;
        g14:=1;
        g20:=1;
        g21:=1;
        g23:=1;
        g24:=1;
        g22:=0;
//do RCPC mother code
        g30:=0;
        g32:=0;
        g34:=1;
        g31:=0;
        g33:=0;

//definição dos puncturing coefficients (rate 2/3)
        P1[1]:=1; P1[2]:=2; P1[3]:=4;

//definição dos puncturing coefficients (rate 8/18)
        P2[1]:=1; P2[2]:=2; P2[3]:=3; P2[4]:=4;
        P2[5]:=5; P2[6]:=7; P2[7]:=8; P2[8]:=10; P2[9]:=11;
//----------------------------- RCPC 1 ------------------------------------------------------------------------
{O Código RCPC é dividido  em duas etapas:
   *encoding by a 16-state mother code;
   *puncturing of mother code.
A matriz resultante desse código (M2) possui 432 bits}

// Código RCPC aplicado aos bits de sensibilidade 1
        edtrcpc1.Color:=clWhite;
        for k:=1 to 112 do
        begin
           if ((k-1)<0) or ((k-2)<0) or ((k-3)<0)or ((k-4)<0)    then
           begin
             msg1[k-1]:=0;
```

```
          msg1[k-2]:=0;
          msg1[k-3]:=0;
          msg1[k-4]:=0;
        end
        else
        begin
          V1[3*k-2]:=(msg1[k]*g10) XOR (msg1[k-1]*g11) XOR (msg1[k-2]*g12) XOR (msg1[k-
3]*g13) XOR (msg1[k-4]*g14);
          V1[3*k-1]:=(msg1[k]*g20) XOR (msg1[k-1]*g21) XOR (msg1[k-2]*g22) XOR (msg1[k-
3]*g23) XOR (msg1[k-4]*g24);
          V1[3*k] := (msg1[k]*g30) XOR (msg1[k-1]*g31) XOR (msg1[k-2]*g32) XOR (msg1[k-
3]*g33) XOR (msg1[k-4]*g34);
        end;
      end;
//-----------------------------------------------Puncturing of mother code (rate 2/3)----------------
      for j:=1 to 168 do
      begin
        h1[j]:=6*((j-1)DIV 3)+P1[j-3*((j-1)DIV 3)];
        C1[j]:=V1[h1[j]];  // c1 matriz de saida do rcpc1
        edtrcpc1.Text:=edtrcpc1.Text + inttostr(C1[j]);
      end;
//------------------------------------ FIM  RCPC 1 ----------------------------------------------------
//---------------------------------- RCPC 2 ----------------------------------------------------------
// Código RCPC aplicado aos bits de sensibilidade 2
      edtrcpc2.Color:=clWhite;
      for k:=1 to 72 do
      begin
        if ((k-1)<0)or ((k-2)<0) or ((k-3)<0) or ((k-4)<0) then
        begin
          msg2[k-1]:=0;
          msg2[k-2]:=0;
          msg2[k-3]:=0;
          msg2[k-4]:=0;
        end
        else
```

```
          begin
             V2[3*k-2]:= (msg2[k]*g10) XOR (msg2[k-1]*g11) XOR (msg2[k-2]*g12) XOR (msg2[k-
3]*g13) XOR (msg2[k-4]*g14);
             V2[3*k-1]:= (msg2[k]*g20) XOR (msg2[k-1]*g21) XOR (msg2[k-2]*g22) XOR (msg2[k-
3]*g23) XOR (msg2[k-4]*g24);
             V2[3*k]  := (msg2[k]*g30) XOR (msg2[k-1]*g31) XOR (msg2[k-2]*g32) XOR (msg2[k-
3]*g33) XOR (msg2[k-4]*g34);
          end;
       end;
//***Puncturing of mother code (rate 8/18)***
       for j:=1 to 162 DO
       Begin
          h2[j]:=12*((j-1)DIV 9)+P2[j-3*((j-1)DIV 9)];
          C2[j]:=V2[h2[j]];
       end;
//---------------------------------------FIM RCPC2 ----------------------------------------------------
//-------------Concatenação dos bits de sensibilidade 0, RCPC1 e RCPC 2 ---------------------
      for j:=1 to 102 do
      begin
          M2[j]:=msg0[j];
      end;
      i:=0;
      for j:=103 to 270 do
      begin
        i:=i+1;
        M2[j]:=msg1[i];
      end;
      i:=0;
      for j:=271 to 432 do
      begin
          i:=i+1;
          M2[j]:=msg2[i];
      end;
     edtconc2.Color:=clWhite;
     for i:=1 to 432 do
```

```
    Begin
      sgM2.Cells[i-1,0]:=inttostr(M2[i]);
      edtconc2.Text:= edtconc2.Text + inttostr(M2[i]); // Preenche o edit concatenação 2.
    end;
//-------------------------------FIM MATRIZ CONCATENADA - matriz de saída M2 ---------
```

Apêndice D: Código em Delphi para o interleaving

```
//-----------------------------------COMEÇO DO INTERLEAVING ------------------------------
     j:=0;
     for i:=1 to 17 do    //---> Formação da matriz na saída do interleaving do 25º termo até o 42º termo
     begin
        M3[i*24+j]:=M2[j*18+i];
     end;
     for j:=1 To 24 Do    //---> Outros termos da matriz M3
     begin
        for i:=0 to 17 do
        M3[i*24+j]:=M2[j*18+i];
     end;
//---------MANDANDO A MATRIZ M3 PARA UM COMPONENTE VISUAL: M3 -------
     edtinter.Color:=clWhite;
     for i:=1 to 432 do
     Begin
       sgM3.Cells[i-1,0]:=inttostr(M3[i]);
       edtinter.text:=edtinter.text + inttostr(M3[i]); // Preenche o edit do Interleaving.
     end;
//---------------------------------- FIM DO INTERLEAVING  -----------------------------------
```

Apêndice E: Código em Delphi para implementação do Scrambling

```
//------------------------------ COMEÇO DO SCRAMBLING -----------------------------------
// Inicialização para a definição do k-ésimo bit da sequência de scrambling
// ---------------------------------Definindo a matriz e ---------------------------------------------
      for i:=1 to 30 do
      e[i]:=0;
//----> Por apção a adotamos toda zero.
//-----------------------------------fim da matriz e ------------------------------------------------
```

```
      for k:=-31 to 0 do
         begin
         if k>=-29  then
           begin
           P[k]:=e[1-k];
           end
         else
          begin
              P[k]:=1;
          end;
         end;
//----------------------- Definição do k-ésimo bit da sequência de scrambling ------------------
      for K:=1 TO 432 DO
        P[k] := (P[k-1]) XOR (P[k-2]) XOR (P[k-4]) XOR (P[k-5]) XOR (P[k-7]) XOR (P[k-8]) XOR
(P[k-10]) XOR (P[k-11]) XOR (P[k-12]) XOR (P[k-16]) XOR (P[k-22]) XOR (P[k-23]) XOR (P[k-26])
XOR (P[k-32]);
//------------------------Definição da matriz de saída do scrambling ---------------------------
      edtscramb.Color:=clWhite;
      for k:=1 To 432 Do
      Begin
        M4[k]:=(M3[k]) XOR (P[k]);
        sgM4.Cells[k-1,0]:=inttostr(M4[K]);
        edtscramb.Text:= edtscramb.Text + inttostr(M4[k]);// Preenche o edit scrambling.
      End;
//--------------------*FIM DO SCRAMBLING - Matriz de saída:M4 ----------------------------
```

Apêndice F: Código em Delphi para implementação da BER

```
procedure Tfrmtetra.btnvalidarClick(Sender: TObject);
var
//------variaveis que receberão os valores dos comboxes ----------------------------------------
Rb,Hr,Ht,Pt,Gt,Gr,F,R,L,Pr,Eb,RSR,PE,BER: extended;
//-------------------------------------------------------------------------------------------------
//L(dB) --> perdas
//Rb(dB)--> capacidade do canal
```

```
//Hr(m)--> altura da antena receptora
//Ht(m)--> altura da antena transmissora
//Pt(dB)--> Potencia transmitida
//Pr(dB)--> Potencia de recepção
//Gt(dB)--> Ganho de transmissão
//Gr(dB)--> Ganho de Recepção
//F (MHz)-> frequencia
//R (km)--> Distancia
//Eb(dB)--> Energia de Bit
//PE -->  Probabilidade de erro.
//RSR --> Relação sinal ruido (Eb/No)
//BER --> Taxa de erro de bit
//backupR --> serve para guardar o valor do Raio inicial e dps que plotar voltar ao normal.
//i --> Contador para numerar o sgDadosBER
//-------------------------------------------------------------------------------------------------
//--------------Previne que a rotina seja executada sem valores-----------------------------------
begin
   if (cbgr.text='') or (cbHt.Text='')or(cbRb.Text='') or (cbR.text='')or (cbHr.text='') or (cbpt.text='') or
(cbgt.text='') or (cbF.text='') then
   showmessage('Preencha os campos vazios!')
//-------------------------------------------------------------------------------------------------
else
//------------------Previne que a rotina seja executada com  valores impossiveis.----------------
begin
   if (cbgr.text='0') or (cbHt.Text='0')or(cbRb.Text='0') or (cbR.text='0')or (cbHr.text='0') or
(cbpt.text='0') or (cbgt.text='0') or (cbF.text='0') then
   showmessage('Preencha com valores válidos! A partir de 1.')
//-------------------------------------------------------------------------------------------------
else
begin
//----------------------Armazena os valores dos comboxes em variaveis -------------------------
rb:=StrToFloat(cbRb.text);
Hr:=StrToFloat(cbHr.text);
Ht:=StrToFloat(cbHt.text);
Pt:=StrToFloat(cbpt.text);
```

```
Gt:=StrToFloat(cbgt.text);
Gr:=StrToFloat(cbgr.text);
F:=StrToFloat(cbF.text);
R:=StrToFloat(cbR.text);
//-------------------------------------------------------------------------------------------------------
if RPGP.Checked = true then
begin
// >>>>>>>>>> OKUMURA HATA PARA GRANDES CIDADES <<<<<<<<<<<<<<
//--------------------Cálculo para definição das Perdas (L(dB)) ----------------------------------
    L:=69.55          +          (26.16*(log10(strtofloat(cbF.text))))-(13.82*(log10(strtofloat(cbht.text))))-
((3.2*(log10(11.75*strtofloat(cbhr.text)))*(log10(11.75*strtofloat(cbhr.text))))-4.97)+(44.9                -
(6.55*log10(strtofloat(cbht.text))))*log10(strtofloat(cbR.text));
    lbperdas.Caption:='L(dB) = '+ FloatToStr(L)+ ' dB';
//-------------------------------------------------------------------------------------------------------
//----------------------------Convertendo Potencia transmitida de w para dB ---------------------
   Pt:=10*Log10(Pt);
//    showmessage('pt é : ' + floattostr(Pt));
//-------------------------------------------------------------------------------------------------------
//-----------------Cálculo para definição da Potencia Recebida No aparelho receptor ----------
Pr:=Pt-L; // Em dBw
Pr:=Power(10,(Pr/10));// Passa de dBw para Watt.
lbPr.Caption:= 'Potência recebida = ' + FloatToStr(Pr) + ' W';
//-------------------------------------------------------------------------------------------------------
//-----------------------Cálculo para definição da Energia de Bit (Eb) ----------------------------
Eb:=Pr/Rb;
lbEb.Caption:='Energia de Bit (Eb)= ' + FloatToStr(Eb) + 'J ';
//-------------------------------------------------------------------------------------------------------
//---------------------------------Relação Sinal Ruído (Eb/No) ----------------------------------
RSR:=Eb/(1.38*power(10,-23)*300*);
// Foi desconsiderado o ruído de ação humana.
{Onde 1,38 * 10 elevado -23 é a constante de Boltzmann, 300 é a temperatura do sistema em Kelvin.
Temos que a Probabilidade de erro é encontrada quando utilizamos a probabilidade de erro
complementar. Para isso  tiramos a raiz quadrada de RSR e olhamos na tabela da erfc. Assim:}
//-------------------------------------------------------------------------------------------------------
//---------------------------------Probabilidade de Erro ----------------------------------------------
```

```
PE:= sqrt(RSR);
//MANDA BUSCAR NA TABELA DA ERFC.
//-------------------------FAZENDO VARREDURA NA TABELA DA ERFC -----------------
//Tabela abaixo (VER Apêndice G)
//---------------------------------------FIM TBL ERFC ----------------------------------------------
lbPE.Caption:='Probabilidade de erro = '+ FloatToStr(PE);
//-------------------------------------------------------------------------------------------------
//--------------------------------- Cálculo da BER ------------------------------------------------
{Temos que a BER é dada pela razao PE/LOG2(M) Onde M é o número de bits/simbolo. Como estamos
trabalhando com a modulação
PI/4 QPSK entao nosso M = 2.}
BER2:= PE/log2(2);
BER:= (PE/log2(2))*100;

lbBER.Caption:='BER = ' + FloatToStr(BER) + ' %';
//-------------------------------------------------------------------------------------------------
//---------------------------------------Fim valores para plotar-----------------------------------
//<<<<<<<<<<<<<FIM OKUMURA HATA PARA GRANDES CIDADES <<<<<<<<<<
end;
//--------------------Faz os labels aparecerem ---------------------------------------------------
        lbperdas.Visible:=true;
        lbPr.Visible:=true;
        lbEb.Visible:=true;
        lbPE.Visible:=true;
        lbBER.Visible:=true;
//-------------------------------------------------------------------------------------------------
end;
end;
end;

procedure Tfrmtetra.btnlimparBERClick(Sender: TObject);
begin
      lbperdas.Visible:=false;
      lbPr.Visible:=false;
      lbEb.Visible:=false;
```

```
        lbPE.Visible:=false;
        lbBER.Visible:=false;
        cbRb.SetFocus;

procedure Tfrmtetra.btngeragraficoClick(Sender: TObject);
var
i: integer;
backupR:integer;// serve para guardar o valor do Raio inicial e dps que plotar voltar ao normal.

begin
//--------------------------------Nomeia as duas colunas do SGDadosBeR----------------------
sgdadosBER.Cells[0,0]:='Distância [km]';
sgdadosBER.Cells[1,0]:='BER ';
//-------------------------------------------------------------------------------------------------------
//---------------------------------GERA VALORES PARA PLOTAR O GRÁFICO ---------
//-------------------Definino valores para a coluna Distância ----------------------------------
      for i:=0 to 99 do
      begin
         sgdadosBER.Cells[0,i+1]:=inttostr(i+1);
      end;
//-------------------------------------------------------------------------------------------------------
//-----------------Definindo valores para a coluna BER ----------------------------------------
      backupR:=strtoint(cbR.text);   //armazena valor inicial
      for i:=0 to 99 do
      begin
         cbR.Text:= inttostr(i+1);
//          delay(500);
         frmtetra.btnvalidarClick(btnvalidar);
         sgdadosBER.Cells[1,i+1]:=FloatToStr(BER2);
         //Para Plotar valores no gráfico
         x[i]:=BER2;
      end;
//-------------------------------------------------------------------------------------------------------
        cbR.Text:=inttostr(backupR);
        frmtetra.btnvalidarClick(btnvalidar);
```

```
    frmgrafico.showmodal; // Mostra o grafico
end;

procedure Tfrmtetra.RPGPClick(Sender: TObject);
begin
//------------- Valores Padrões para Okumura Hata (Grandes Cidades) ----------------------
    cbRb.Text:=inttostr(36000);
    cbHr.Text:=floattostr(1.5);
    cbHt.Text:=inttostr(35);
    cbpt.Text:=inttostr(25);
    cbgt.Text:=inttostr(8);
    cbgr.Text:=IntToStr(3);
    cbF.Text:=IntToStr(800);
    cbR.Text:=inttostr(8);
```

Apêndice G: Tabela ERFC

A seguir sera mostrado a tabela da ERFC que foi utilizada para cálculos da BER.

```
if (PE <= 0.01) and (PE >=0) then
PE := 0.989;
if (PE > 0.01) and (PE <= 0.02) then
PE:=0.977;
if (PE > 0.02) and (PE <= 0.03) then
PE:=0.966;
if (PE > 0.03) and (PE <= 0.04) then
PE:=0.955;
if (PE > 0.04) and (PE <= 0.05) then
PE:=0.944;
if (PE > 0.05) and (PE <= 0.06) then
PE:=0.932;
if (PE > 0.06) and (PE <= 0.07) then
PE:=0.921;
if (PE > 0.07) and (PE <= 0.08) then
PE:=0.910;
if (PE > 0.08) and (PE <= 0.09) then
PE:=0.899;
if (PE > 0.09) and (PE <= 0.10) then
PE:=0.888;
if (PE > 0.10) and (PE <= 0.11) then
PE:=0.876;
if (PE > 0.11) and (PE <= 0.12) then
PE:=0.856;
if (PE > 0.12) and (PE <= 0.13) then
PE:=0.854;
if (PE > 0.13) and (PE <= 0.14) then
PE:=0.843;
if (PE > 0.14) and (PE <= 0.15) then
PE:=0.832;
if (PE > 0.15) and (PE <= 0.16) then
PE:=0.821;
if (PE > 0.16) and (PE <= 0.17) then
PE:=0.810;
```

```
if (PE > 0.17) and (PE <= 0.18) then
PE:=0.799;
if (PE > 0.18) and (PE <= 0.19) then
PE:=0.788;
if (PE > 0.19) and (PE <= 0.20) then
PE:=0.777;
if (PE > 0.20) and (PE <= 0.21) then
PE:=0.766;
if (PE > 0.21) and (PE <= 0.22) then
PE:=0.756;
if (PE > 0.22) and (PE <= 0.23) then
PE:=0.745;
if (PE > 0.23) and (PE <= 0.24) then
PE:=0.734;
if (PE > 0.24) and (PE <= 0.25) then
PE:=0.724;
if (PE > 0.25) and (PE <= 0.26) then
PE:=0.713;
if (PE > 0.26) and (PE <= 0.27) then
PE:=0.703;
if (PE > 0.27) and (PE <= 0.28) then
PE:=0.692;
if (PE > 0.28) and (PE <= 0.29) then
PE:=0.682;
if (PE > 0.29) and (PE <= 0.30) then
PE:=0.671;
if (PE > 0.30) and (PE <= 0.31) then
PE:=0.661;
if (PE > 0.31) and (PE <= 0.32) then
PE:=0.651;
if (PE > 0.32) and (PE <= 0.33) then
PE:=0.641;
if (PE > 0.33) and (PE <= 0.34) then
PE:=0.631;
if (PE > 0.34) and (PE <= 0.35) then
PE:=0.621;
if (PE > 0.35) and (PE <= 0.36) then
PE:=0.611;
if (PE > 0.36) and (PE <= 0.37) then
PE:=0.601;
if (PE > 0.37) and (PE <= 0.38) then
PE:=0.591;
if (PE > 0.38) and (PE <= 0.39) then
PE:=0.581;
if (PE > 0.39) and (PE <= 0.40) then
PE:=0.572;
if (PE > 0.40) and (PE <= 0.41) then
PE:=0.562;
if (PE > 0.41) and (PE <= 0.42) then
PE:=0.553;
if (PE > 0.41) and (PE <= 0.42) then
PE:=0.553;
if (PE > 0.42) and (PE <= 0.43) then
PE:=0.553;
if (PE > 0.43) and (PE <= 0.44) then
PE:=0.534;
if (PE > 0.44) and (PE <= 0.45) then
PE:=0.525;
if (PE > 0.45) and (PE <= 0.46) then
PE:=0.515;
if (PE > 0.46) and (PE <= 0.47) then
PE:=0.506;
if (PE > 0.47) and (PE <= 0.48) then
PE:=0.497;
if (PE > 0.48) and (PE <= 0.49) then
PE:=0.488;
if (PE > 0.49) and (PE <= 0.50) then
PE:=0.479;
if (PE > 0.50) and (PE <= 0.51) then
PE:=0.471;
```

```
if (PE > 0.51) and (PE <= 0.52) then
PE:=0.462;
if (PE > 0.52) and (PE <= 0.53) then
PE:=0.454;
if (PE > 0.53) and (PE <= 0.54) then
PE:=0.445;
if (PE > 0.54) and (PE <= 0.55) then
PE:=0.437;
if (PE > 0.55) and (PE <= 0.56) then
PE:=0.428 ;
if (PE > 0.56) and (PE <= 0.57) then
PE:=0.420 ;
if (PE > 0.57) and (PE <= 0.58) then
PE:=0.412 ;
if (PE > 0.58) and (PE <= 0.59) then
PE:=0.404;
if (PE > 0.59) and (PE <= 0.60)then
PE:=0.396;
if (PE > 0.60) and (PE <= 0.61) then
PE:=0.388;
if (PE > 0.61) and (PE <= 0.62) then
PE:=0.381;
if (PE > 0.62) and (PE <= 0.63) then
PE:=0.373;
if (PE > 0.63) and (PE <= 0.64)then
PE:=0.365 ;
if (PE > 0.64) and (PE <= 0.65) then
PE:=0.358 ;
if (PE > 0.65) and (PE <= 0.66) then
PE:=0.351;
if (PE > 0.66) and (PE <= 0.67) then
PE:=0.343;
if (PE > 0.67) and (PE <= 0.68) then
PE:=0.336;
if (PE > 0.68) and (PE <= 0.69) then
PE:=0.329;
if (PE > 0.69) and (PE <= 0.70) then
PE:=0.322;
if (PE > 0.70) and (PE <= 0.71) then
PE:=0.315;
if (PE > 0.71) and (PE <= 0.72) then
PE:=0.302;
if (PE > 0.72) and (PE <= 0.73) then
PE:=0.301;
if (PE > 0.73) and (PE <= 0.74) then
PE:=0.295;
if (PE > 0.74) and (PE <= 0.75) then
PE:=0.289;
if (PE > 0.75) and (PE <= 0.76) then
PE:=0.282;
if (PE > 0.76) and (PE <= 0.77) then
PE:=0.276;
if (PE > 0.77) and (PE <= 0.78) then
PE:=0.270;
if (PE > 0.78) and (PE <= 0.79) then
PE:=0.264;
if (PE > 0.79) and (PE <= 0.80) then
PE:=0.258;
if (PE > 0.80) and (PE <= 0.81) then
PE:=0.252;
if (PE > 0.81) and (PE <= 0.82) then
PE:=0.246;
if (PE > 0.82) and (PE <= 0.83) then
PE:=0.240;
if (PE > 0.83) and (PE <= 0.84) then
PE:=0.235;
if (PE > 0.84) and (PE <= 0.85) then
PE:=0.229;
if (PE > 0.85) and (PE <= 0.86) then
PE:=0.224;
```

```
if (PE > 0.86) and (PE <= 0.87) then
PE:=0.219;
if (PE > 0.87) and (PE <= 0.88) then
PE:=0.213;
if (PE > 0.88) and (PE <= 0.89) then
PE:=0.208;
if (PE > 0.89) and (PE <= 0.90) then
PE:=0.203;
if (PE > 0.90) and (PE <= 0.91) then
PE:=0.198;
if (PE > 0.91) and (PE <= 0.92) then
PE:=0.193;
if (PE > 0.92) and (PE <= 0.93) then
PE:=0.188;
if (PE > 0.93) and (PE <= 0.94) then
PE:=0.184;
if (PE > 0.94) and (PE <= 0.95) then
PE:=0.179;
if (PE > 0.95) and (PE <= 0.96) then
PE:=0.175;
if (PE > 0.96) and (PE <= 0.97) then
PE:=0.170;
if (PE > 0.97) and (PE <= 0.98) then
PE:=0.166;
if (PE > 0.98) and (PE <= 0.99) then
PE:=0.161;
if (PE > 0.99) and (PE <= 1) then
PE:=0.157;
if (PE > 1) and (PE <= 1.01) then
PE:=0.153;
if (PE > 1.01) and (PE <= 1.02) then
PE:=0.149;
if (PE > 1.02) and (PE <= 1.03) then
PE:=0.145;
if (PE > 1.03) and (PE <= 1.04) then
PE:=0.141;
if (PE > 1.04) and (PE <= 1.05) then
PE:=0.138;
if (PE > 1.05) and (PE <= 1.06) then
PE:=0.134;
if (PE > 1.06) and (PE <= 1.07) then
PE:=0.130;
if (PE > 1.07) and (PE <= 1.08) then
PE:=0.127;
if (PE > 1.08) and (PE <= 1.09) then
PE:=0.123;
if (PE > 1.09) and (PE <= 1.10) then
PE:=0.120;
if (PE > 1.10) and (PE <= 1.11) then
PE:=0.116;
if (PE > 1.11) and (PE <= 1.12) then
PE:=0.113;
if (PE > 1.12) and (PE <= 1.13) then
PE:=0.110;
if (PE > 1.13) and (PE <= 1.14) then
PE:=0.107;
if (PE > 1.14) and (PE <= 1.15) then
PE:=0.104;
if (PE > 1.15) and (PE <= 1.16) then
PE:=0.101;
if (PE > 1.16) and (PE <= 1.17) then
PE:=0.0980;
if (PE > 1.17) and (PE <= 1.18) then
PE:=0.0950;
if (PE > 1.18) and (PE <= 1.19) then
PE:=0.0924;
if (PE > 1.19) and (PE <= 1.20) then
PE:=0.0897;
if (PE > 1.20) and (PE <= 1.21) then
PE:=0.0870;
```

```
if (PE > 1.21) and (PE <= 1.22) then
PE:=0.0845;
if (PE > 1.22) and (PE <= 1.23) then
PE:=0.0819;
if (PE > 1.23) and (PE <= 1.24) then
PE:=0.0795;
if (PE > 1.24) and (PE <= 1.25) then
PE:=0.0771;
if (PE > 1.25) and (PE <= 1.26) then
PE:=0.0748;
if (PE > 1.26) and (PE <= 1.27) then
PE:=0.0725;
if (PE > 1.27) and (PE <= 1.28) then
PE:=0.0703;
if (PE > 1.28) and (PE <= 1.29) then
PE:=0.0681;
if (PE > 1.29) and (PE <= 1.30) then
PE:=0.0660;
if (PE > 1.30) and (PE <= 1.31) then
PE:=0.0639;
if (PE > 1.31) and (PE <= 1.32) then
PE:=0.0619;
if (PE > 1.32) and (PE <= 1.33) then
PE:=0.0600;
if (PE > 1.33) and (PE <= 1.34) then
PE:=0.0581;
if (PE > 1.34) and (PE <= 1.35) then
PE:=0.0562;
if (PE > 1.35) and (PE <= 1.36) then
PE:=0.0544;
if (PE > 1.36) and (PE <= 1.37) then
PE:=0.0527;
if (PE > 1.37) and (PE <= 1.38) then
PE:=0.0510;
if (PE > 1.38) and (PE <= 1.39) then
PE:=0.0492;
if (PE > 1.39) and (PE <= 1.40) then
PE:=0.0477;
if (PE > 1.40) and (PE <= 1.41) then
PE:=0.0461;
if (PE > 1.41) and (PE <= 1.42) then
PE:=0.0446;
if (PE > 1.42) and (PE <= 1.43) then
PE:=0.0431;
if (PE > 1.43) and (PE <= 1.44) then
PE:=0.0417;
if (PE > 1.44) and (PE <= 1.45) then
PE:=0.0403;
if (PE > 1.45) and (PE <= 1.46) then
PE:=0.0389;
if (PE > 1.46) and (PE <= 1.47) then
PE:=0.0376;
if (PE > 1.47) and (PE <= 1.48) then
PE:=0.0363;
if (PE > 1.48) and (PE <= 1.49) then
PE:=0.0351;
if (PE > 1.49) and (PE <= 1.50) then
PE:=0.0339;
if (PE > 1.50) and (PE <= 1.51) then
PE:=0.0327;
if (PE > 1.51) and (PE <= 1.52) then
PE:=0.0316;
if (PE > 1.52) and (PE <= 1.53) then
PE:=0.0305;
if (PE > 1.53) and (PE <= 1.54) then
PE:=0.0294;
if (PE > 1.54) and (PE <= 1.55) then
PE:=0.0284;
if (PE > 1.55) and (PE <= 1.56) then
PE:=0.0274;
```

```
if (PE > 1.56) and (PE <= 1.57) then
PE:=0.0264;
if (PE > 1.57) and (PE <= 1.58) then
PE:=0.0255;
if (PE > 1.58) and (PE <= 1.59) then
PE:=0.0245;
if (PE > 1.59) and (PE <= 1.60) then
PE:=0.0237;
if (PE > 1.60) and (PE <= 1.61) then
PE:=0.0228;
if (PE > 1.61) and (PE <= 1.62) then
PE:=0.0220;
if (PE > 1.62) and (PE <= 1.63) then
PE:=0.0212;
if (PE > 1.63) and (PE <= 1.64) then
PE:=0.0204;
if (PE > 1.64) and (PE <= 1.65) then
PE:=0.0196;
if (PE > 1.65) and (PE <= 1.66) then
PE:=0.0189;
if (PE > 1.66) and (PE <= 1.67) then
PE:=0.0182;
if (PE > 1.67) and (PE <= 1.68) then
PE:=0.0175;
if (PE > 1.68) and (PE <= 1.69) then
PE:=0.0168;
if (PE > 1.69) and (PE <= 1.70) then
PE:=0.0162;
if (PE > 1.70) and (PE <= 1.71) then
PE:=0.0156;
if (PE > 1.71) and (PE <= 1.72) then
PE:=0.0150;
if (PE > 1.72) and (PE <= 1.73) then
PE:=0.0144;
if (PE > 1.73) and (PE <= 1.74) then
PE:=0.0139;
if (PE > 1.74) and (PE <= 1.75) then
PE:=0.0133;
if (PE > 1.75) and (PE <= 1.76) then
PE:=0.0128;
if (PE > 1.76) and (PE <= 1.77) then
PE:=0.0123;
if (PE > 1.77) and (PE <= 1.78) then
PE:=0.0118;
if (PE > 1.78) and (PE <= 1.79) then
PE:=0.0114;
if (PE > 1.79) and (PE <= 1.80) then
PE:=0.0109;
if (PE > 1.80) and (PE <= 1.81) then
PE:=0.0105;
if (PE > 1.81) and (PE <= 1.82) then
PE:=0.0101;
if (PE > 1.82) and (PE <= 1.83) then
PE:=9.65*power(10,-3);
if (PE > 1.83) and (PE <= 1.85) then
PE:=8.89*power(10,-3);
if (PE > 1.85) and (PE <= 1.86) then
PE:=8.56*power(10,-3);
if (PE > 1.86) and (PE <= 1.87) then
PE:=8.18*power(10,-3);
if (PE > 1.87) and (PE <= 1.88) then
PE:=7.84*power(10,-3);
if (PE > 1.88) and (PE <= 1.89) then
PE:=7.52*power(10,-3);
if (PE > 1.89) and (PE <= 1.90) then
PE:=7.21*power(10,-3);
if (PE > 1.90) and (PE <= 1.91) then
PE:=6.91*power(10,-3);
if (PE > 1.91) and (PE <= 1.92) then
PE:=6.62*power(10,-3);
```

```
if (PE > 1.92) and (PE <= 1.93) then
PE:=6.34*power(10,-3);
if (PE > 1.93) and (PE <= 1.94) then
PE:=6.08*power(10,-3);
if (PE > 1.94) and (PE <= 1.95) then
PE:=5.82*power(10,-3);
if (PE > 1.95) and (PE <= 1.96) then
PE:=5.57*power(10,-3);
if (PE > 1.96) and (PE <= 1.97) then
PE:=5.34*power(10,-3);
if (PE > 1.97) and (PE <= 1.98) then
PE:=5.11*power(10,-3);
if (PE > 1.98) and (PE <= 1.99) then
PE:=4.89*power(10,-3);
if (PE > 1.99) and (PE <= 2) then
PE:=4.88*power(10,-3);
if (PE > 2) and (PE <= 2.01) then
PE:=4.48*power(10,-3);
if (PE > 2.01) and (PE <= 2.02) then
PE:=4.28*power(10,-3);
if (PE > 2.02) and (PE <= 2.03) then
PE:=4.09*power(10,-3);
if (PE > 2.03) and (PE <= 2.04) then
PE:=3.91*power(10,-3);
if (PE > 2.04) and (PE <= 2.05) then
PE:=3.74*power(10,-3);
if (PE > 2.05) and (PE <= 2.06) then
PE:=3.58*power(10,-3);
if (PE > 2.06) and (PE <= 2.07) then
PE:=3.42*power(10,-3);
if (PE > 2.07) and (PE <= 2.08) then
PE:=3.27*power(10,-3);
if (PE > 2.08) and (PE <= 2.09) then
PE:=3.12*power(10,-3);
if (PE > 2.09) and (PE <= 2.10) then
PE:=2.98*power(10,-3);
if (PE > 2.10) and (PE <= 2.11) then
PE:=2.85*power(10,-3);
if (PE > 2.11) and (PE <= 2.12) then
PE:=2.72*power(10,-3);
if (PE > 2.12) and (PE <= 2.13) then
PE:=2.59*power(10,-3);
if (PE > 2.13) and (PE <= 2.14) then
PE:=2.47*power(10,-3);
if (PE > 2.14) and (PE <= 2.15) then
PE:=2.36*power(10,-3);
if (PE > 2.15) and (PE <= 2.16) then
PE:=2.25*power(10,-3);
if (PE > 2.16) and (PE <= 2.17) then
PE:=2.15*power(10,-3);
if (PE > 2.17) and (PE <= 2.18) then
PE:=2.05*power(10,-3);
if (PE > 2.18) and (PE <= 2.19) then
PE:=1.95*power(10,-3);
if (PE > 2.19) and (PE <= 2.20) then
PE:=1.86*power(10,-3);
if (PE > 2.20) and (PE <= 2.21) then
PE:=1.78*power(10,-3);
if (PE > 2.21) and (PE <= 2.22) then
PE:=1.69*power(10,-3);
if (PE > 2.22) and (PE <= 2.23) then
PE:=1.61*power(10,-3);
if (PE > 2.23) and (PE <= 2.24) then
PE:=1.54*power(10,-3);
if (PE > 2.24) and (PE <= 2.25) then
PE:=1.46*power(10,-3);
if (PE > 2.25) and (PE <= 2.26) then
PE:=1.39*power(10,-3);
if (PE > 2.26) and (PE <= 2.27) then
PE:=1.33*power(10,-3);
```

```
if (PE > 2.27) and (PE <= 2.28) then
PE:=1.26*power(10,-3);
if (PE > 2.28) and (PE <= 2.29) then
PE:=1.20*power(10,-3);
if (PE > 2.29) and (PE <= 2.30) then
PE:=1.14*power(10,-3);
if (PE > 2.30) and (PE <= 2.31) then
PE:=1.09*power(10,-3);
if (PE > 2.31) and (PE <= 2.32) then
PE:=1.03*power(10,-3);
if (PE > 2.32) and (PE <= 2.33) then
PE:=9.84*power(10,-4);
if (PE > 2.33) and (PE <= 2.34) then
PE:=9.36*power(10,-4);
if (PE > 2.34) and (PE <= 2.35) then
PE:=8.89*power(10,-4);
if (PE > 2.35) and (PE <= 2.36) then
PE:=8.45*power(10,-4);
if (PE > 2.36) and (PE <= 2.37) then
PE:=8.03*power(10,-4);
if (PE > 2.37) and (PE <= 2.38) then
PE:=7.63*power(10,-4);
if (PE > 2.38) and (PE <= 2.39) then
PE:=7.25*power(10,-4);
if (PE > 2.39) and (PE <= 2.40) then
PE:=6.89*power(10,-4);
if (PE > 2.40) and (PE <= 2.41) then
PE:=6.54*power(10,-4);
if (PE > 2.41) and (PE <= 2.42) then
PE:=6.21*power(10,-4);
if (PE > 2.42) and (PE <= 2.43) then
PE:=5.89*power(10,-4);
if (PE > 2.43) and (PE <= 2.44) then
PE:=5.59*power(10,-4);
if (PE > 2.44) and (PE <= 2.45) then
PE:=5.31*power(10,-4);
if (PE > 2.45) and (PE <= 2.46) then
PE:=5.03*power(10,-4);
if (PE > 2.46) and (PE <= 2.47) then
PE:=4.78*power(10,-4);
if (PE > 2.47) and (PE <= 2.48) then
PE:=4.53*power(10,-4);
if (PE > 2.48) and (PE <= 2.49) then
PE:=4.29*power(10,-4);
if (PE > 2.49) and (PE <= 2.50) then
PE:=4.07*power(10,-4);
if (PE > 2.50) and (PE <= 2.51) then
PE:=3.86*power(10,-4);
if (PE > 2.51) and (PE <= 2.52) then
PE:=3.66*power(10,-4);
if (PE > 2.52) and (PE <= 2.53) then
PE:=3.46*power(10,-4);
if (PE > 2.53) and (PE <= 2.54) then
PE:=3.28*power(10,-4);
if (PE > 2.54) and (PE <= 2.55) then
PE:=3.11*power(10,-4);
if (PE > 2.55) and (PE <= 2.56) then
PE:=2.94*power(10,-4);
if (PE > 2.56) and (PE <= 2.57) then
PE:=2.79*power(10,-4);
if (PE > 2.57) and (PE <= 2.58) then
PE:=2.64*power(10,-4);
if (PE > 2.58) and (PE <= 2.59) then
PE:=2.50*power(10,-4);
if (PE > 2.59) and (PE <= 2.60) then
PE:=2.36*power(10,-4);
if (PE > 2.60) and (PE <= 2.61) then
PE:=2.23*power(10,-4);
if (PE > 2.61) and (PE <= 2.62) then
PE:=2.11*power(10,-4);
```

```
if (PE > 2.62) and (PE <= 2.63) then
PE:=2.00*power(10,-4);
if (PE > 2.63) and (PE <= 2.64) then
PE:=1.89*power(10,-4);
if (PE > 2.64) and (PE <= 2.65) then
PE:=1.79*power(10,-4);
if (PE > 2.65) and (PE <= 2.66) then
PE:=1.69*power(10,-4);
if (PE > 2.66) and (PE <= 2.67) then
PE:=1.59*power(10,-4);
if (PE > 2.67) and (PE <= 2.68) then
PE:=1.51*power(10,-4);
if (PE > 2.68) and (PE <= 2.69) then
PE:=1.42*power(10,-4);
if (PE > 2.69) and (PE <= 2.70) then
PE:=1.34*power(10,-4);
if (PE > 2.70) and (PE <= 2.71) then
PE:=1.27*power(10,-4);
if (PE > 2.71) and (PE <= 2.72) then
PE:=1.20*power(10,-4);
if (PE > 2.72) and (PE <= 2.73) then
PE:=1.13*power(10,-4);
if (PE > 2.73) and (PE <= 2.74) then
PE:=1.07*power(10,-4);
if (PE > 2.74) and (PE <= 2.75) then
PE:=1.01*power(10,-4);
if (PE > 2.75) and (PE <= 2.76) then
PE:=9.50*power(10,-5);
if (PE > 2.76) and (PE <= 2.77) then
PE:=8.96*power(10,-5);
if (PE > 2.77) and (PE <= 2.78) then
PE:=8.44*power(10,-5);
if (PE > 2.78) and (PE <= 2.79) then
PE:=7.96*power(10,-5);
if (PE > 2.79) and (PE <= 2.80) then
PE:=7.50*power(10,-5);
if (PE > 2.80) and (PE <= 2.81) then
PE:=7.07*power(10,-5);
if (PE > 2.81) and (PE <= 2.82) then
PE:=6.66*power(10,-5);
if (PE > 2.82) and (PE <= 2.83) then
PE:=6.28*power(10,-5);
if (PE > 2.83) and (PE <= 2.84) then
PE:=5.91*power(10,-5);
if (PE > 2.84) and (PE <= 2.85) then
PE:=5.57*power(10,-5);
if (PE > 2.85) and (PE <= 2.86) then
PE:=5.24*power(10,-5);
if (PE > 2.86) and (PE <= 2.87) then
PE:=4.94*power(10,-5);
if (PE > 2.87) and (PE <= 2.88) then
PE:=4.64*power(10,-5);
if (PE > 2.88) and (PE <= 2.89) then
PE:=4.37*power(10,-5);
if (PE > 2.89) and (PE <= 2.90) then
PE:=4.11*power(10,-5);
if (PE > 2.90) and (PE <= 2.91) then
PE:=3.87*power(10,-5);
if (PE > 2.91) and (PE <= 2.92) then
PE:=3.64*power(10,-5);
if (PE > 2.92) and (PE <= 2.93) then
PE:=3.42*power(10,-5);
if (PE > 2.93) and (PE <= 2.94) then
PE:=3.22*power(10,-5);
if (PE > 2.94) and (PE <= 2.95) then
PE:=3.02*power(10,-5);
if (PE > 2.95) and (PE <= 2.96) then
PE:=2.84*power(10,-5);
if (PE > 2.96) and (PE <= 2.97) then
PE:=2.67*power(10,-5);
```

```
if (PE > 2.97) and (PE <= 2.98) then
PE:=2.51*power(10,-5);
if (PE > 2.98) and (PE <= 2.99) then
PE:=2.35*power(10,-5);
if (PE > 2.99) and (PE <= 3.00) then
PE:=2.21*power(10,-5);
if (PE > 3.00) and (PE <= 3.01) then
PE:=2.08*power(10,-5);
if (PE > 3.01) and (PE <= 3.02) then
PE:=1.95*power(10,-5);
if (PE > 3.02) and (PE <= 3.03) then
PE:=1.83*power(10,-5);
if (PE > 3.03) and (PE <= 3.04) then
PE:=1.72*power(10,-5);
if (PE > 3.04) and (PE <= 3.05) then
PE:=1.61*power(10,-5);
if (PE > 3.05) and (PE <= 3.06) then
PE:=1.52*power(10,-5);
if (PE > 3.06) and (PE <= 3.07) then
PE:=1.42*power(10,-5);
if (PE > 3.07) and (PE <= 3.08) then
PE:=1.33*power(10,-5);
if (PE > 3.07) and (PE <= 3.08) then
PE:=1.33*power(10,-5);
if (PE > 3.08) and (PE <= 3.09) then
PE:=1.24*power(10,-5);
if (PE > 3.09) and (PE <= 3.10) then
PE:=1.17*power(10,-5);
if (PE > 3.10) and (PE <= 3.11) then
PE:=1.09*power(10,-5);
if (PE > 3.11) and (PE <= 3.12) then
PE:=1.02*power(10,-5);
if (PE > 3.12) and (PE <= 3.13) then
PE:=9.59*power(10,-6);
if (PE > 3.13) and (PE <= 3.14) then
PE:=8.98*power(10,-6);
if (PE > 3.14) and (PE <= 3.15) then
PE:=8.41*power(10,-6);
if (PE > 3.15) and (PE <= 3.16) then
PE:=7.87*power(10,-6);
if (PE > 3.16) and (PE <= 3.17) then
PE:=7.36*power(10,-6);
if (PE > 3.17) and (PE <= 3.18) then
PE:=6.89*power(10,-6);
if (PE > 3.18) and (PE <= 3.19) then
PE:=6.45*power(10,-6);
if (PE > 3.19) and (PE <= 3.20) then
PE:=6.03*power(10,-6);
if (PE > 3.20) and (PE <= 3.21) then
PE:=5.64*power(10,-6);
if (PE > 3.21) and (PE <= 3.22) then
PE:=5.27*power(10,-6);
if (PE > 3.22) and (PE <= 3.23) then
PE:=4.93*power(10,-6);
if (PE > 3.23) and (PE <= 3.24) then
PE:=4.61*power(10,-6);
if (PE > 3.24) and (PE <= 3.25) then
PE:=4.31*power(10,-6);
if (PE > 3.25) and (PE <= 3.26) then
PE:=4.02*power(10,-6);
if (PE > 3.26) and (PE <= 3.27) then
PE:=3.76*power(10,-6);
if (PE > 3.27) and (PE <= 3.28) then
PE:=3.51*power(10,-6);
if (PE > 3.28) and (PE <= 3.29) then
PE:=3.28*power(10,-6);
if (PE > 3.29) and (PE <= 3.30) then
PE:=3.06*power(10,-6);
if (PE > 3.30) and (PE <= 3.31) then
PE:=2.86*power(10,-6);
```

```
if (PE > 3.31) and (PE <= 3.32) then
PE:=2.67*power(10,-6);
if (PE > 3.32) and (PE <= 3.33) then
PE:=2.49*power(10,-6);
if (PE > 3.33) and (PE <= 3.34) then
PE:=2.32*power(10,-6);
if (PE > 3.34) and (PE <= 3.35) then
PE:=2.17*power(10,-6);
if (PE > 3.35) and (PE <= 3.36) then
PE:=2.02*power(10,-6);
if (PE > 3.36) and (PE <= 3.37) then
PE:=1.88*power(10,-6);
if (PE > 3.37) and (PE <= 3.38) then
PE:=1.75*power(10,-6);
if (PE > 3.38) and (PE <= 3.39) then
PE:=1.64*power(10,-6);
if (PE > 3.39) and (PE <= 3.40) then
PE:=1.52*power(10,-6);
if (PE > 3.40) and (PE <= 3.41) then
PE:=1.42*power(10,-6);
if (PE > 3.41) and (PE <= 3.42) then
PE:=1.32*power(10,-6);
if (PE > 3.42) and (PE <= 3.43) then
PE:=1.23*power(10,-6);
if (PE > 3.43) and (PE <= 3.44) then
PE:=1.15*power(10,-6);
if (PE > 3.44) and (PE <= 3.45) then
PE:=1.07*power(10,-6);
if (PE > 3.45) and (PE <= 3.46) then
PE:=9.94*power(10,-7);
if (PE > 3.46) and (PE <= 3.47) then
PE:=9.25*power(10,-7);
if (PE > 3.47) and (PE <= 3.48) then
PE:=8.60*power(10,-7);
f (PE > 3.48) and (PE <= 3.49) then
PE:=8.00*power(10,-7);
if (PE > 3.49) and (PE <= 3.50) then
PE:=7.44*power(10,-7);

if PE > 3.5 then
PE:=0.000000744;
```

Apêndice H: Código em Delphi para implementação do Canal de Controle

```
procedure TfrmSimAut.btniniciarSimAutClick(Sender: TObject);

begin
        btniniciarSimAut.Enabled:=false;        //Desativa o botão para simular.
        i1.Visible:=true;    // mostra a imagem numero 1 do canal de controle
        delay(1000);        // espera 1 segundos e mostra a figura numero 2 (comando abaixo)

        i2.Visible:=true;
        i1.Visible:=false;    //oculta figura 1
        delay(1000);
```

```
i3.Visible:=true;
i2.Visible:=false;
delay(1000);

i4.Visible:=true;
i3.Visible:=false;
delay(1000);

i5.Visible:=true;
i4.Visible:=false;
delay(1000);

i6.Visible:=true;
i5.Visible:=false;
delay(3000);

i7.Visible:=true;
i6.Visible:=false;
delay(1000);

i8.Visible:=true;
i7.Visible:=false;
delay(1000);

i9.Visible:=true;
i8.Visible:=false;
delay(1000);

i10.Visible:=true;
i9.Visible:=false;
delay(1000);

i11.Visible:=true;
i10.Visible:=false;
```

```
        delay(1000);

        i12.Visible:=true;
        i11.Visible:=false;
        delay(1000);

        i13.Visible:=true;
        i12.Visible:=false;
        delay(1000);

        i14.Visible:=true;
        i13.Visible:=false;
        delay(1000);
        btniniciarSimAut.Caption:='Reiniciar';     //Botão iniciar é renomeado para Reiniciar.
        btniniciarSimAut.Enabled:=true;
end;

procedure TfrmSimAut.FormClose(Sender: TObject; var Action: TCloseAction);
begin
                btniniciarSimAut.Caption:='Iniciar';   // Botão Reiniciar volta a se chamar Iniciar.
                btniniciarSimAut.Enabled:=true;        //Volta a ativar o botão
                i1.visible:=false;
                i2.Visible:=false;
                i3.Visible:=false;
                i4.Visible:=false;
                i5.Visible:=false;
                i6.Visible:=false;
                i7.Visible:=false;
                i8.Visible:=false;
                i9.Visible:=false;
                i10.Visible:=false;
                i11.Visible:=false;
                i12.Visible:=false;
                i13.Visible:=false;
                i14.Visible:=false;
```

```
end;
```

Apêndice I: Código em Delphi para implementação do Controle de tráfego

O código abaixo foi utilizado para implementer a ação do botão “Rádio Móvel”.

```
procedure Tfrmtetra.btnRMClick(Sender: TObject);
var
i: integer;
//ch,ts,chp,tsp,chb,tsb,cha,tsa: integer;
begin
clique:= clique + 1;
    i:=0;
    for i:=0 to 20 do
    memocontroletrafego.Lines[i]:=''; // permite que as proximas 20 linhas do memo sejam utilizadas.
    memocontroletrafego.Lines[0]:='Rádio Móvel se comunica com o Canal de Controle para solicitar um canal de comunicação livre.';    //Faz aparecer no memo a mensagem ente aspas.
    imageCC.Visible:= not imageCC.Visible;    //faz a imagem do canal de controle ficar piscando
    delay(300);                    // faz uma pausa de meio segundo.
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCC.Visible:= not imageCC.Visible;
```

```
        delay(300);
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCC.Visible:= true;

        memocontroletrafego.Lines[1]:='Um Time Slot livre é então escolhido.';
        delay(900);
        Randomize;
        ch:=StrToInt(tsch[random(2)]);
        chp:=ch;
        ts:= StrToInt(tsch[random(4)]);
        tsp:=ts;
//---> manda para o sgcheck para verficar os canais utilizados!-------------------------
         //para os canais
         sgcheck.Cells[0,clique]:=inttostr(chp);  // para os canais
         sgcheck.Cells[1,clique]:=inttostr(tsp);  // para os Time slots
         for i:=1 to clique do
         begin
            for w:=1 to (clique - 1) do   // é ate -1 para que ele nao se compare consigo mesmo.
            begin
              if (sgcheck.Cells[0,clique] = sgcheck.Cells[0,w]) then       //para verificar se algum canal
              begin
                 if sgcheck.Cells[1,clique]= sgcheck.Cells[1,w] then //verfica se o ts se choca
                 begin
                    if tsp<>4 then
                    begin //apagar linha 2
```

```
                tsp:=tsp+1;    //se o ts ocupado  não for o 4º ele redimensionará para o proximo TS
vago
                end
                else
                begin
                 tsp:=tsp-1;      //se o ts ocupado for o 4º ele redimensionará para o TS anterior vago
                end;
                memocontroletrafego.Lines[1]:='Este TS já está ocupado! Estamos redirecioando
para um disponível. Você foi alocado no canal ' + inttostr(chp) +' TS número ' + inttostr(tsp);
                sgcheck.Cells[0,clique]:=inttostr(chp);  // para os canais
                sgcheck.Cells[1,clique]:=inttostr(tsp);  // para os Time slots
              end;
              //colocar um loop para caso caia em um caso qeu já tenha. volta lá pra cima para
escolher novo canal e novo ts.
            end;
          end;
       end;
       sgcheck.Cells[0,clique]:=inttostr(chp);  // para os canais
       sgcheck.Cells[1,clique]:=inttostr(tsp);  // para os Time slots
//-----------------------------------------FIM-----------------------------------------------------
      memocontroletrafego.Lines[2]:=' O Canal escolhido foi o ' + inttostr(chp) +' e está ocupando o
slot número ' + inttostr(tsp);
      delay(2000);
      memocontroletrafego.Lines[3]:='Trocando infomações Com o Rádio Móvel ....';
      if ch=1 then
      begin
         imageCH1.visible:= not imageCH1.visible;
         delay(300);
         imageCH1.visible:= not imageCH1.visible;
         imageCC.Visible:= not imageCC.Visible;
         delay(300);
         imageCH1.visible:= not imageCH1.visible;
         imageCC.Visible:= not imageCC.Visible;
         delay(300);
         imageCH1.visible:= not imageCH1.visible;
```

```
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH1.visible:= not imageCH1.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH1.visible:= not imageCH1.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH1.visible:= not imageCH1.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH1.visible:= not imageCH1.visible;
    imageCC.Visible:= not imageCC.Visible;
    imageCC.Visible:=true;
end
else
begin
    imageCH2.visible:= not imageCH2.visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
    imageCC.Visible:= not imageCC.Visible;
    delay(300);
    imageCH2.visible:= not imageCH2.visible;
```

```
        imageCC.Visible:= not imageCC.Visible;
        delay(300);
        imageCH2.visible:= not imageCH2.visible;
        imageCC.Visible:= not imageCC.Visible;
        imageCH2.Visible:= true;
        imageCC.Visible:= true;
    end;
    memocontroletrafego.Lines[4]:='Trocando infomações Com o Rádio Móvel ....OK';
    delay(1000);
    memocontroletrafego.Lines[5]:='Clique em saiba mais para obter mais detalhes sobre o
funcionamento do canal de tráfego.';
end;
```

Apêndice J: Código delphi para implementação do Fading

```
procedure Tfrmtetra.btnexecutar2Click(Sender: TObject);
var

  i,j: integer; // de 1 a 10--> será a contadora de fi1... theta1..
begin
//limpa sg

    for i:=0 to 30 do
    begin
       for j:=0 to (strtoint(cbtemp.Text) * 2) do
       sg.Cells[i,j]:='';
    end;
//-------------------------------------------------------------------------------
//----------------------------------HABILITA BTN GERAR GRAFICO--------------------------
      btngeragraficofading.Enabled:=true;
//-------------------------------------------------------------------------------------
//---------------------------------permite t somente até 500-----------------------
      if strtoint(cbtemp.Text) > 500 then
      begin
      showmessage('O valor máximo permitido é 500.');
      cbtemp.Text:=inttostr(500);
      end;
//-------------------------------------------------------------------------
//---------------------------------Definindo nome da coluna ---------------------------------------
    sg.Cells[0,0]:='i';
    sg.Cells[1,0]:='theta (i)';
    sg.Cells[2,0]:='fi';
    sg.Cells[3,0]:='cos(theta)';
    sg.Cells[4,0]:='vd.cos(theta)';
```

```
      sg.Cells[5,0]:='np';
      sg.Cells[6,0]:='Parte Real';   //cos(fi(i)-(t.vd.cos(theta)))
      sg.Cells[7,0]:='Parte Imag.';  //sen(fi(i)-(t.vd.cos(theta)))
      sg.Cells[8,0]:='r(t)';
      sg.Cells[9,0]:='|r(t)|';
      sg.Cells[10,0]:='normalizado';
      sg.Cells[11,0]:='em dB';
//-------------------------------------------------------------------------------------------------------
//--------------------------------------VELOCIDADE DOOPLER ---------------------------------
      vd:= StrToFloat(cbvd.Text);
      //vd:=fd*2*Pi;
//-------------------------------------------------------------------------------------------------------
//--------------------------------------Tempo ----------------------------------------------
      temp:= StrToInt(cbtemp.Text);
      for i:=1 to temp do
      begin
        temp2[i]:=i; // tempo que será plotado
      end;
      if temp < 500 then   // o que faltou para 500 será zero.
      begin
         for j:=(500-(500-temp)+1) to 500 do
         temp2[j]:= 0;
      end;
//-------------------------------------------------------------------------------------------------------
//--------------------------------------DEFININDO THETA ---------------------------------------
// 2*PI  divido pelo nº de multipercursos  * 1 até 10 (np)
     if StrToInt(cbnp.Text) > 10 then
     begin
       showmessage('Atualmente valor máximo permitido é 10.');
       cbnp.Text:='10';
     end;
     np:=StrToInt(cbnp.Text);
//---------------------Limpando as variáveis-------------------------------------------------------
     for i:=1 to 10 do
     begin
       theta[i]:=0;
       coss[i]:=0;
       fi[i]:=0;
     end;

     for i:=1 to 500 do
     r[t]:=0;
//-------------------------------------------------------------------------------------------------
     for i:=1 to np do
     begin
        theta[i]:=((2*Pi)/np)*i;// defini theta em radianos
        coss[i]:=vd*cos(theta[i]);
        sg.Cells[5,i]:=FloatToStr(i);
        sg.Cells[1,i]:=FloatToStr(theta[i]);// manda theta para sg
        sg.Cells[3,i]:=FloatToStr(cos(theta[i])); // manda cos(theta) para sg
        sg.Cells[4,i]:=FloatToStr(vd*cos(theta[i])); // manda vd* cos(theta) para sg
     end;
```

```
//-------------------------------------------------------------------------------------------------------
       for i:=1 to temp do
       sg.Cells[0,i]:=IntToStr(i);// numera coluna até o fim
//-------------------------Definir fi ------------------------------------------------------------
//---------------------------Função Aleatória-----------------------------------------------------
        for i:=1 to np do
        begin
           Aleat:=RandG(0.1,0.9);  // gera um numero entre 0 e 1 aproximadamente
           while (Aleat < 0) or (Aleat > 1) do
           Aleat:=RandG(0.1,0.9);
           fi[i]:=2*PI*aleat; // gera o angulo fi aleatorio
           sg.Cells[2,i]:=FloatToStr(fi[i]); // armazena esse angulo no sg.
        end;
//-------------------------------------------------------------------------------------------------------
       rmax:=0; //Valor inicial adotado para o valor maximo de r(t)
//------------------Simulação de 0 a temp. Segundos E divisão de parte real e parte imaginaria ---------
     for t:=0 to strtoint(cbtemp.text) do
     begin
       // Parte Real
       Part1[t]:=cos(fi[1]-t*vd*cos(theta[1]))+cos(fi[2]-t*vd*cos(theta[2]))+cos(fi[3]-
t*vd*cos(theta[3]))+cos(fi[4]-t*vd*cos(theta[4]))+cos(fi[5]-t*vd*cos(theta[5]))+cos(fi[6]-
t*vd*cos(theta[6]))+cos(fi[7]-t*vd*cos(theta[7]))+cos(fi[8]-t*vd*cos(theta[8]))+cos(fi[9]-
t*vd*cos(theta[9]))+cos(fi[10]-t*vd*cos(theta[10]));
        sg.Cells[6,t+1]:=FloatToStr(Part1[t]);

        //Parte Imaginaria
        Part2[t]:=sin(fi[1]-t*vd*cos(theta[1]))+sin(fi[2]-t*vd*cos(theta[2]))+sin(fi[3]-
t*vd*cos(theta[3]))+sin(fi[4]-t*vd*cos(theta[4]))+sin(fi[5]-t*vd*cos(theta[5]))+sin(fi[6]-
t*vd*cos(theta[6]))+sin(fi[7]-t*vd*cos(theta[7]))+sin(fi[8]-t*vd*cos(theta[8]))+sin(fi[9]-
t*vd*cos(theta[9]))+sin(fi[10]-t*vd*cos(theta[10]));
        sg.Cells[7,t+1]:=FloatToStr(Part2[t]);
        r[t]:=sqrt(sqr(Part1[t])+sqr(Part2[t]));//aplica modulo
        if r[t] > rmax then
        begin
          rmax:=r[t];
          sg.Cells[8,t+1]:=FloatToStr(r[t]);
          sg.Cells[9,t+1]:=FloatToStr(abs(r[t]));
          cbmodrt.Text:=FloatToStr(abs(r[t]));
        end
        else
        begin
          sg.Cells[8,t+1]:=FloatToStr(r[t]);
          sg.Cells[9,t+1]:=FloatToStr(abs(r[t]));
          cbmodrt.Text:=FloatToStr(abs(r[t]));
        end;
end;
//-------------------------------------------------------------------------------------------------------
//-----------------------------------NORMALIZANDO-----------------------------------------
     for t:=0 to strtoint(cbtemp.text) do
     begin
        r[t]:=r[t]/rmax;
        if r[t]< -30 then
```

```
        begin
        while r[t]<-30 do
        r[t]:=r[t]/rmax;
        showmessage(inttostr(t));
        end;
        sg.Cells[10,t+1]:=FloatToStr(r[t]);
        r[t]:=20*log10(r[t]);//transformando em dB
        sg.Cells[11,t+1]:=FloatToStr(r[t]);
      end;
//-------------------------------------------------------------------------------------------------------
end;

procedure Tfrmtetra.btngeragraficofadingClick(Sender: TObject);
//---------------------------------------gera grafico-----------------------------------------
begin
With Series1 do
  Begin
    Clear;
    for i:=0 to strtoint(cbtemp.text) do
    begin
         Add(r[i],inttostr(temp2[i]),clred);
    end;
  end;
//----------------------------------DESABILITA BTN GERAR GRAFICO--------------------
       btngeragraficofading.Enabled:=false;
//-----------------------------------------------------------------------------------------
end;

procedure Tfrmtetra.cb3dClick(Sender: TObject);
begin
// desabilita 3d
       if cb3d.Checked = true then
       Chfading.Chart3DPercent:=1
       else
       Chfading.Chart3DPercent:=15;
end;
```

X. Referências

[1] F.J.Macwilliams, N.J.A.Sloane, holland-North,The Theory of Error Correcting Codes",mathematical library

[2] ETS 300 395-02, TETRA - Speech codec for full-rate traffic channel.February 1998.

[3] ETS 300 392-02,Radio Equipments and Systems; TETRA; Speech codec for full-rate traffic channel; part 2: TETRA codec, clause 5.

[4] Terrestrial Trunked Radio (TETRA); Speech codec for full-rate traffic channel; Part 2: TETRA codec – annex E]

[5] Dunlop J, Girma D and Irvine J, Digitall Mobile Communications and the TETRA System, John Wiley & Sons, 1999 pp.130-131

[6] YACOUB,Michel – Foundations of Mobile Radio Engineering,1993.

[7] ETS 300 392-02,Radio Equipments and Systems; TETRA; Voice plus Data; Part 2: Air Interface.

[8] Como fazer monografia [http://www.delfim.info/documentologia/monog.htm]