

Leonardo Muttoni

Desenvolvimento de um Receptor ISDB-T 1-Seg através de Rádio Definido por Software

Uberlândia-MG

Dezembro de 2017

Leonardo Muttoni

Desenvolvimento de um Receptor ISDB-T 1-Seg através de Rádio Definido por Software

Texto da dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos necessários à obtenção do título de Mestre em Ciências.

Universidade Federal de Uberlândia – UFU

Faculdade de Engenharia Elétrica

Programa de Pós-Graduação em Engenharia Elétrica - COPEL

Orientador: Antônio Cláudio Paschoarelli da Veiga

Uberlândia-MG

Dezembro de 2017

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

M993d Muttoni, Leonardo, 1984-
2017 Desenvolvimento de um receptor ISDB-T 1-Seg através de rádio
definido por Software / Leonardo Muttoni. - 2017.
184 f. : il.

Orientador: Antônio Cláudio Paschoarelli da Veiga.
Dissertação (mestrado) - Universidade Federal de Uberlândia,
Programa de Pós-Graduação em Engenharia Elétrica.
Disponível em: <http://dx.doi.org/10.14393/ufu.di.2017.104>
Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Televisão - Receptores e recepção -
Teses. I. Veiga, Antônio Cláudio Paschoarelli da, 1963- II. Universidade
Federal de Uberlândia. Programa de Pós-Graduação em Engenharia
Elétrica. III. Título.

CDU: 621.3

Leonardo Muttoni

Desenvolvimento de um Receptor ISDB-T 1-Seg através de Rádio Definido por Software

Texto da dissertação apresentada à Universidade Federal de Uberlândia como parte dos requisitos necessários à obtenção do título de Mestre em Ciências.

Banca examinadora:

**Antônio Cláudio Paschoarelli da
Veiga, Dr.**
Orientador (UFU)

Gilberto Arantes Carrijo, Dr.
Prof. Convidado (UFU)

Cláriton Rodrigues Bernadelli, Dr.
Prof. Convidado (UFTM)

Uberlândia-MG
Dezembro de 2017

Agradecimentos

À Deus, pela minha vida, saúde e por me guiar onde quer que eu esteja.

À minha família, Neuza, Celso, Laura e Nattália, por me apoiarem na busca por novos horizontes.

À minha esposa, Marília, pelo apoio diário e incondicional nesta jornada.

Agradeço ao meu orientador, Prof. Paschoarelli, pelo apoio no desenvolvimento deste trabalho desde o início, pelas palavras motivadoras que permitiram a realização deste sonho.

“Se vi mais longe foi por estar de pé sobre ombros de gigantes”

Isaac Newton, 1676

Resumo

Este trabalho apresenta o desenvolvimento de um receptor de TV digital de um segmento (1 seg) para o padrão *Integrated Services Digital Broadcasting - Terrestrial* (ISDB-T), com base em Rádio Definido por Software (RDS). O receptor proposto funciona em tempo real e requer um computador e o RTL-SDR, uma plataforma de hardware RDS de baixo custo. O processamento de sinal executado em software fornece uma estrutura importante para o estudo do padrão ISDB-T na prática.

São apresentados os resultados do funcionamento do receptor, o desempenho das suas principais funções de rádio e a ocupação da CPU.

Palavras-chave: ISDB-T, RDS, SDR, OFDM, RTL-SDR.

Abstract

This work presents the development of a one-segment (1-seg) digital TV receiver for the Integrated Services Digital Broadcasting - Terrestrial (ISDB-T) standard, based on Software Defined Radio (SDR). The proposed receiver works in real time and requires a computer and the RTL-SDR, a low-cost SDR hardware platform. Signal processing performed in software provides an important framework for the study of the ISDB-T standard in practice.

The results of the receiver operation, the performance of its main radio functions and the workload imposed on the CPU are presented.

Keywords: ISDB-T, SDR, OFDM, RTL-SDR.

Lista de ilustrações

Figura 1 – Estrutura deste trabalho	19
Figura 2 – Arquitetura de RDS idealizada para a transmissão e recepção	21
Figura 3 – Receptor RDS típico	22
Figura 4 – Comparativo de processadores para o RDS: nível de reconfiguração x velocidade de desenvolvimento	26
Figura 5 – Placa de circuito impresso do RTL-SDR	30
Figura 6 – Diagrama de blocos do RTL-SDR com sinonizador R820T/T2	30
Figura 7 – Diagrama de blocos do RTL-SDR com sinonizador E4000	31
Figura 8 – Diagrama esquemático do circuito de um RTL-SDR - Parte 1: sintoni- zador R820T	32
Figura 9 – Diagrama esquemático do circuito de um RTL-SDR - Parte 2: C.I. principal RTL2832U	33
Figura 10 – Ruído de fundo do RTL-SDR	37
Figura 11 – Desvio de frequência entre o RTL-SDR e a emissora	39
Figura 12 – Desvio de frequência entre o RTL-SDR e a emissora - Ampliação do gráfico da Figura 11 nos 30 minutos iniciais	40
Figura 13 – Uso do GNU Radio como plataforma de RDS	41
Figura 14 – Matlab/Simulink como plataforma de RDS	42
Figura 15 – LabVIEW para o RDS	42
Figura 16 – Efeito do multipercurso	45
Figura 17 – Exemplo de modelo de canal para recepção portátil	47
Figura 18 – Interferência intersimbólica causada pela propagação multipercurso	48
Figura 19 – Efeito do desvanecimento plano	48
Figura 20 – Espalhamento temporal dos símbolos, efeito do desvanecimento seletivo em frequência, provocando ISI	49
Figura 21 – Efeito da propagação multipercurso no domínio da frequência para um receptor estático	50
Figura 22 – Efeito da propagação multipercurso no domínio da frequência para um receptor em movimento	51
Figura 23 – Comparação entre (a) sistema de portadora única e (b) OFDM	53
Figura 24 – Ortogonalidade das subportadoras OFDM	55
Figura 25 – Transmissor e receptor OFDM	56
Figura 26 – Inclusão do prefixo cíclico na transmissão	57
Figura 27 – Ilustração do LFSR equivalente	71
Figura 28 – $m'(x)$: Mensagem alongada para 239 bytes, com a inclusão de 51 bytes nulos no início	74

Figura 29 – $c'(x)$: Mensagem alongada processada pelo codificador RS(255, 239), que gerou 16 bytes de paridade, resultando em um palavra-código de 255 bytes	75
Figura 30 – $c(x)$: Palavra-código transmitida, com a remoção dos 51 bytes nulos que foram inseridos no início da mensagem antes da codificação	75
Figura 31 – Fluxo codificador \rightarrow canal \rightarrow decodificador	76
Figura 32 – Exemplo de um codificador convolucional com $K = 3$ e taxa $R = 1/2$.	84
Figura 33 – Codificador convolucional da Figura 32 representado em um diagrama de transição de estados	86
Figura 34 – Exemplo de codificação através da treliça	87
Figura 35 – Codificador convolucional utilizado no ISDB-T	89
Figura 36 – Um trecho da treliça	91
Figura 37 – Exemplo de decodificação através do algoritmo de Viterbi - parte 1 . .	94
Figura 38 – Exemplo de decodificação através do algoritmo de Viterbi - parte 2 . .	95
Figura 39 – Traceback simples e duplo	96
Figura 40 – Comparação do desempenho entre o método do traceback simples e do duplo	97
Figura 41 – Comparação do desempenho do VA para diferentes níveis de quantização da entrada	99
Figura 42 – Espectro de dois canais adjacentes, sendo um analógico e o outro digital	102
Figura 43 – Arranjo dos segmentos e exemplo de configuração para duas camadas hierárquicas	103
Figura 44 – Posicionamento da portadoras OFDM do ISDB-T dentro de um canal de televisão	104
Figura 45 – Opções de intervalo de guarda	108
Figura 46 – Diagrama de blocos do transmissor ISDB-T	109
Figura 47 – Gerador PRBS utilizado no dispersor de energia	110
Figura 48 – Funcionamento do intercalador de byte	111
Figura 49 – Diagrama do modulador DQPSK	112
Figura 50 – Constelação DQPSK	113
Figura 51 – Diagrama do modulador QPSK	113
Figura 52 – Mapeamento dos bits na constelação QPSK	113
Figura 53 – Diagrama do modulador 16QAM	114
Figura 54 – Mapeamento dos bits na constelação 16QAM	114
Figura 55 – Diagrama do modulador 64QAM	115
Figura 56 – Mapeamento dos bits na constelação 64QAM	116
Figura 57 – Funcionamento do intercalador de tempo	117
Figura 58 – Espalhamento do atraso de símbolo provocado pelo intercalador de tempo	117
Figura 59 – Etapas do intercalador de frequência	118

Figura 60 – Gerador PRBS utilizado para a síntese do sinal piloto	119
Figura 61 – Informações contidas no quadro TMCC	119
Figura 62 – Informações contidas nos blocos <code>info_atual</code> e <code>info_proxima</code>	121
Figura 63 – Exemplo de quadro utilizado na modulação diferencial (DQPSK) no modo 1	124
Figura 64 – Exemplo de quadro utilizado nas modulações síncronas (QPSK, 16QAM e 64QAM) no modo 1	125
Figura 65 – Visão geral do hardware do receptor desenvolvido	126
Figura 66 – Cadeia de processamento do transmissor ISDB-T e do receptor ISDB-T 1-seg	128
Figura 67 – Diagrama dos componentes do receptor ISDB-T 1-seg desenvolvido	131
Figura 68 – Gráficos de $f_1(k)$ para cada modo de transmissão e intervalo de guarda possíveis	132
Figura 69 – Diagrama do estimador do início do símbolo e do desvio de frequência fracionária	134
Figura 70 – Sinal OFDM e seu prefixo cíclico	135
Figura 71 – Gráfico para obtenção das estimativas $\hat{\theta}$ e $\hat{\mathcal{E}}$ com 3 símbolos OFDM	136
Figura 72 – Gráfico para obtenção das estimativas $\hat{\theta}$ e $\hat{\mathcal{E}}$ utilizando a média dos últimos 32 símbolos OFDM	137
Figura 73 – Funcionamento do desintercalador de tempo	142
Figura 74 – Funcionamento do demodulador QPSK	142
Figura 75 – Função $D(X)$ para o desmapeamento QPSK em 3 diferentes níveis de quantização	144
Figura 76 – Funcionamento do desintercalador de byte	145
Figura 77 – Receptor em operação e ferramenta de monitoração	151
Figura 78 – Magnitude do sinal antes do equalizador	152
Figura 79 – Magnitude do sinal depois do equalizador	153
Figura 80 – Desempenho do decodificador Viterbi comparado com o do Matlab	154
Figura 81 – Diagrama de Venn para algumas estruturas algébricas	165

Lista de tabelas

Tabela 1	– Comparativo dos processadores para RDS	27
Tabela 2	– Comparativo de dispositivos RDS	28
Tabela 3	– Comparativo de sintonizadores do RTL-SDR	31
Tabela 4	– Modelos de canais multipercuro	46
Tabela 5	– M : Exemplo de mensagem a ser codificada	77
Tabela 6	– C : Mensagem da Tabela 5 codificada	78
Tabela 7	– R : Vetor da Tabela 6 com erros acrescentados	79
Tabela 8	– R' : R estendido com 51 zeros no início	80
Tabela 9	– Padrões de puncionamento do código convolucional do ISDB-T	90
Tabela 10	– Parâmetros OFDM para diferentes modos de transmissão	105
Tabela 11	– Cálculos dos parâmetros do ISDB-T	106
Tabela 12	– Parâmetros de transmissão do ISDB-T	107
Tabela 13	– Calculador de fase DQPSK	112
Tabela 14	– Parâmetros para o intercalador de tempo	115
Tabela 15	– Configuração das portadoras OFDM para o segmento zero na modulação diferencial	122
Tabela 16	– Configuração das portadoras OFDM para o segmento zero nas modula- ções síncronas	123
Tabela 17	– Processo de escolha da frequência de amostragem f_s do receptor desen- volvido	129
Tabela 18	– Canais sintonizados em Uberlândia com o receptor desenvolvido	148
Tabela 19	– Uso relativo da CPU pelos blocos do receptor	155
Tabela 20	– Tabela adição para $GF(3)$	169
Tabela 21	– Tabela multiplicação para $GF(3)$	169
Tabela 22	– Tabela adição para $GF(2)$	170
Tabela 23	– Tabela multiplicação para $GF(2)$	170
Tabela 24	– Tabela multiplicação para a tentativa de construção do $GF(2^2)$	170
Tabela 25	– Elementos do campo $GF(2^2)$	172
Tabela 26	– Tabela soma para o $GF(2^2) = GF(2)[x]/\langle x^2 + x + 1 \rangle$	173
Tabela 27	– Tabela multiplicação para o $GF(2^2) = GF(2)[x]/\langle x^2 + x + 1 \rangle$	173
Tabela 28	– Geração de todos os elementos não zero do campo $GF(16) = GF(2)[x]/\langle x^4 +$ $x + 1 \rangle$ através das potências da raiz α de $p(x)$	174
Tabela 29	– Tentativa de se gerar todos os elementos não zero do campo $F =$ $GF(2)[x]/\langle x^4 + x^3 + x^2 + x + 1 \rangle$ através das potências da raiz α de $p(x)$	175
Tabela 30	– Geração de todos os elementos não zero do campo $F = GF(2)[x]/\langle x^4 +$ $x^3 + x^2 + x + 1 \rangle$ através das potências do elemento $\beta = x + 1$	176

Tabela 31 – Elementos do $GF(2^8) = GF(2)[x]/\langle p(x) = x^8 + x^4 + x^3 + x^2 + 1 \rangle$. . . 177

Sumário

	Introdução	17
1	RÁDIO DEFINIDO POR SOFTWARE	20
1.1	Introdução	20
1.2	Evolução do RDS	24
1.3	Dispositivos de processamento	25
1.4	Plataformas de Hardware	27
1.4.1	RTL-SDR	28
1.4.1.1	Funcionamento do RTL-SDR	31
1.4.1.2	Sintonizador E4000	34
1.4.1.3	Sintonizador R820T/T2	34
1.4.1.4	Chip principal RTL2832U	35
1.4.1.5	Sistemas de clock e de alimentação	36
1.4.1.6	Ruído de fundo	36
1.4.1.7	Estabilidade do oscilador	37
1.5	Plataformas de Software	41
1.5.1	GNU Radio	41
1.5.2	Matlab/Simulink	41
1.5.3	LabVIEW	42
1.5.4	API	42
2	SISTEMAS OFDM	44
2.1	Propagação Multipercurso	44
2.2	OFDM	52
2.2.1	Histórico	53
2.2.2	Ortogonalidade	54
2.2.3	Implementação	55
2.2.4	Prefixo ciclico	57
3	CÓDIGOS CORRETORES DE ERRO	59
3.1	Códigos Reed-Solomon	59
3.1.1	Introdução	59
3.1.2	Abordagem original dos códigos RS	60
3.1.3	Abordagem clássica dos códigos RS	62
3.1.3.1	Codificação sistemática	63
3.1.4	Decodificação	64

3.1.4.1	Decodificador Peterson-Gorenstein-Zierler (PGZ)	64
3.1.4.2	Algoritmo de Berlekamp-Massey	70
3.1.4.3	Algoritmo de Forney	71
3.1.5	Aplicação dos códigos RS no ISDB-T	73
3.1.5.1	Encurtamento do código RS(255, 239) para RS(204, 188)	73
3.1.5.2	Codificação	76
3.1.5.3	Decodificação	78
3.2	Códigos Convolucionais	84
3.2.1	Distância livre	87
3.2.2	Puncionamento	88
3.2.3	Aplicação dos códigos convolucionais no ISDB-T	89
3.2.4	Decodificação pelo Algoritmo de Viterbi	89
4	SISTEMA ISDB-T	101
4.1	Histórico	101
4.2	Segmentação do canal	102
4.3	Recepção parcial	103
4.4	Canalização	103
4.5	Parâmetros de transmissão	104
4.6	Transmissão ISDB-T	108
4.6.1	Codificador externo RS(204, 188)	110
4.6.2	Dispensor de energia	110
4.6.3	Intercalador de byte	110
4.6.4	Codificador convolucional	111
4.6.5	Modulador e intercalador de bit	111
4.6.5.1	DQPSK	111
4.6.5.2	QPSK	112
4.6.5.3	16QAM	112
4.6.5.4	64QAM	113
4.6.6	Intercalador de tempo	114
4.6.7	Intercalador de frequência	115
4.6.7.1	Intercalamento entre diferentes segmentos	116
4.6.7.2	Rotação de portadoras dentro do segmento	117
4.6.7.3	Aleatorização de portadoras dentro do segmento	118
4.6.8	Sinal piloto	118
4.6.9	TMCC	119
4.6.10	Quadro OFDM	121
4.6.11	IFFT	121
4.6.12	Intervalo de guarda	122
4.6.13	Forma de onda na saída do transmissor	123

5	RECEPTOR DESENVOLVIDO	126
5.1	Hardware	129
5.1.1	Taxa de amostragem	129
5.2	Software	130
5.2.1	Arquitetura	130
5.2.2	Blocos funcionais	130
5.2.2.1	Detector do modo de transmissão e do intervalo de guarda	130
5.2.2.2	Estimador do início do símbolo e do desvio de frequência fracionária	133
5.2.2.3	Corretor do início do símbolo e do desvio de frequência	137
5.2.2.4	Remoção do intervalo de guarda	138
5.2.2.5	FFT	138
5.2.2.6	Desmontador do quadro OFDM	139
5.2.2.7	Decodificador TMCC e detector do desvio de frequência inteira	139
5.2.2.8	Estimador do canal e equalizador	140
5.2.2.9	Desintercalador de frequência	141
5.2.2.10	Desintercalador de tempo	141
5.2.2.11	Demodulador e desintercalador de bit	142
5.2.2.12	Decodificador Viterbi	144
5.2.2.13	Desintercalador de byte	144
5.2.2.14	Reversor da dispersão de energia	145
5.2.2.15	Decodificador Reed-Solomon	146
5.2.2.16	Regenerador MPEG-2 TS	147
5.3	Resultados	148
5.3.1	Emissoras sintonizadas	148
5.3.2	Sincronizadores	151
5.3.3	Equalizador	152
5.3.4	Decodificador Viterbi	152
5.3.5	Decodificador Reed-Solomon	153
5.3.6	Velocidade de processamento	153
6	CONCLUSÃO	156
	REFERÊNCIAS	158
	APÊNDICES	164
	APÊNDICE A – ÁLGEBRA ABSTRATA PARA OS CÓDIGOS REED-SOLOMON	165
A.1	Grupos	165

A.2	Anéis	167
A.3	Campos	168
A.4	Campos Finitos $GF(p)$	168
A.5	Campos Finitos $GF(p^n)$	170
A.5.1	Polinômios sobre $GF(p)$	171
A.5.2	Polinômio irreduzível	171
A.5.3	Extensão de campos	172
A.5.4	Polinômio primitivo e elemento gerador do campo	173
A.5.5	Elemento gerador de um campo formado com um polinômio não primitivo	175
APÊNDICE B – POTÊNCIAS DE α PARA O $GF(256)$		177

Introdução

Este trabalho descreve o desenvolvimento de um receptor de TV digital ISDB-T 1-seg completo e funcional através de Rádio Definido por Software – RDS. O receptor opera em tempo real e requer um computador pessoal e o *RTL-SDR*, um dispositivo USB de baixo custo (\approx US\$ 8,00) utilizado como plataforma de hardware RDS, conectado a uma antena.

No contexto em que os sistemas de telecomunicações se tornam cada vez mais ubíquos, acessíveis e ricos em conteúdo para o usuário final, também há um crescimento exponencial da complexidade destes sistemas para fazerem frente à necessidade crescente da conectividade em todo lugar, com grande largura de banda, baixa latência e boa confiabilidade, mesmo sob condições adversas do canal. Esta necessidade se traduziu na criação de sistemas como as redes celulares, a TV e o rádio digital, o WiFi, Bluetooth, GPS, ADSL, dentre outros.

A complexidade destes sistemas apresenta desafios importantes para os engenheiros, pesquisadores e professores da área. Por exemplo, a TV digital implantada há 10 anos no Brasil, possui complexidade incomparável ao seu sistema antecessor analógico, de 66 anos de idade. Com a exceção de poucos centros de pesquisa do país, geralmente ficamos de fora da rota das inovações e dos grandes desenvolvimentos tecnológicos globais. Para gerar inovação, se faz necessário o conhecimento do existente, e é nesta linha mestre que este trabalho se baseia: conhecer, na teoria e prática, parte do existente para suportar a inovação futura.

Dado este contexto desafiador, os principais motivadores deste trabalho foram:

Desafio : O sistema ISDB-T é relativamente complexo. Sua decodificação demandou a construção de 19 *funções de rádio* em software, dentre elas: sincronizador de tempo e frequência, demodulador OFDM, equalizador, desintercaladores, decodificador convolucional (Viterbi) e Reed-Solomon. O desafio de navegar por distintas áreas do conhecimento e culminar com a construção de um receptor completo e funcional que operasse em tempo real foi um dos motivadores importantes deste trabalho. Em um desafio semelhante, Matt Ettus¹ disse em entrevista à revista Wired Magazine (2006) sobre a construção de um receptor de TV digital ATSC²: “*because it was the Mount Everest... it was the biggest receive-only mountain*”.

¹ Fundador da Ettus Research em 2004 e criador do USRP, considerada a primeira plataforma de hardware comercial de baixo custo para RDS. Em 2010 a Ettus Research foi adquirida pela gigante National Instruments.

² Padrão de TV digital dos Estados Unidos

Conhecimento aplicável a outros sistemas de telecomunicação : Várias funções de rádio do ISDB-T são utilizadas em outros sistemas de telecomunicação. O OFDM, por exemplo, é utilizado em muitos dos principais sistemas de telecomunicação atuais (WiFi, rede móvel LTE, ADSL, WiMax, DVB-T, DAB, *Powerline Communications*) e futuros (como a rede móvel 5G), assim como o codificador convolucional e Reed-Solomon, e também as técnicas de equalização e sincronização. Portanto, os conhecimentos adquiridos e difundidos neste trabalho não se limitam especificamente ao ISDB-T, e podem servir de base para outras pesquisas e desenvolvimentos.

Disponibilidade de recursos à baixo custo : Em muitos casos, o tema da pesquisa é desafiador e rico em conhecimentos a serem apropriados e difundidos, porém o custo para sua realização é proibitivo, principalmente em um país que investe apenas 0,63% do PIB em ciência e tecnologia³. Quando isto ocorre, o trabalho geralmente fica restrito a simulações, não havendo a possibilidade do desenvolvimento de um protótipo e validação em um ambiente real. No caso deste trabalho, contou-se com a disponibilidade de um dispositivo SDR de baixíssimo custo, o RTL-SDR, além do acesso, sem custo, da norma técnica do ISDB-T (em particular a japonesa, ARIB B31).

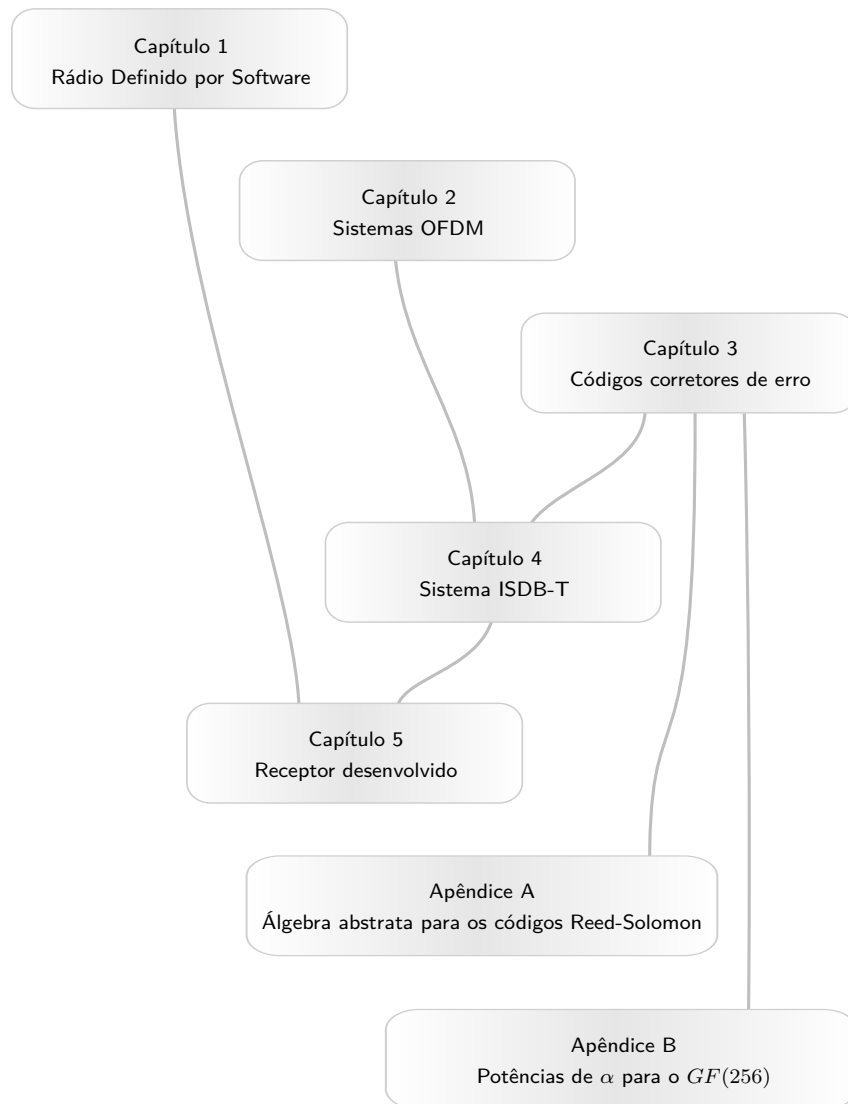
Aplicabilidade no ensino : Tratando-se de um receptor de baixo custo, funcional e desenvolvido de ponta a ponta, há a possibilidade de aplicar o sistema apresentado neste trabalho como laboratório de graduação do curso de Engenharia de Telecomunicações. A flexibilidade do RDS empregado permite o estudo do sistema ISDB-T na prática, incluindo todas as suas partes em detalhes, chegando até o nível de uma amostra de RF, bit ou byte.

A construção do receptor utilizou a premissa de se desenvolver as funções de rádio a partir do zero, limitando ao mínimo possível o uso de bibliotecas e *funções prontas* de terceiros, com o objetivo de atingir o maior nível de conhecimento experimental possível do sistema.

O núcleo deste trabalho é organizado em cinco capítulos e dois apêndices, distribuídos conforme a Figura 1. O Capítulo 1 apresenta o conceito do Rádio Definido por Software, e apresenta a plataforma RTL-SDR em detalhes. O Capítulo 2 discute as principais características dos sistemas OFDM. O Capítulo 3 apresenta os códigos corretores de erros convolucional e Reed-Solomon, utilizados na codificação de canal do ISDB-T. Já o Capítulo 4 apresenta o sistema de TV digital japonês adotado pelo Brasil, com ênfase na transmissão. O Capítulo 5, por sua vez, traz a implementação do receptor desenvolvido, função a função. Inclui também os resultados do seu funcionamento. Os apêndices A e B dão importante suporte ao Capítulo 3 - Códigos corretores de erro.

³ http://www.ipea.gov.br/portal/index.php?option=com_content&view=article&id=29255

Figura 1: Estrutura deste trabalho



Fonte: o autor

1 Rádio Definido por Software

Muitos dos principais engenheiros de hoje no setor de telecomunicações podem ter iniciado suas carreiras nas décadas de 1960 e 1970 ao construir um receptor de rádio de cristal, onde ouviam transmissões de rádio tarde da noite usando seu rádio caseiro. A capacidade de fazer um receptor de rádio portátil, barato e que realmente funcionou foi um motivador surpreendente para o aspirante a engenheiro. Talvez o elemento-chave para a popularização do rádio de cristal foi a sua simplicidade e que fazê-lo era gratificante. (STEWART et al., 2015, p. xiii, tradução nossa)

Assim como no passado era divertido e instrutivo construir um receptor de rádio de cristal (também conhecido rádio de galena) para ouvir emissoras AM, hoje há uma infinidade de sinais irradiados ao nosso redor para o *aspirante a engenheiro* conhecer e eventualmente, por curiosidade ou desafio, construir um receptor (ou transmissor) para aquele sinal. Porém, os sistemas de telecomunicações atuais são majoritariamente digitais com complexidade muito maior do que aquele simples receptor de Amplitude Modulada feito basicamente com três componentes discretos. No presente, a tecnologia do *Rádio Definido por Software* (RDS, ou SDR do inglês *Software Defined Radio*) é uma ferramenta importante tanto para o desenvolvimento de protótipos como para a implementação definitiva de um rádio.

1.1 Introdução

O RDS é uma tecnologia que permite a construção de diversas funções de um rádio¹ em software, executadas por um dispositivo de processamento programável. O RDS herda as vantagens inerentes de um software, principalmente a facilidade de mudanças.

Com o objetivo de uniformizar o vocabulário desta tecnologia relativamente recente, o SDR forum (2007) estabeleceu algumas definições importantes da terminologia no contexto do RDS e do Rádio Cognitivo. As principais são:

- *Software* - Instruções modificáveis executadas por um dispositivo de processamento programável;
- *Rádio* - Tecnologia para se transmitir ou receber informações através do uso de radiação eletromagnética;
- *Definido por Software* - refere-se ao uso do processamento de *software* em um sistema de *rádio* para implementar funções operacionais;

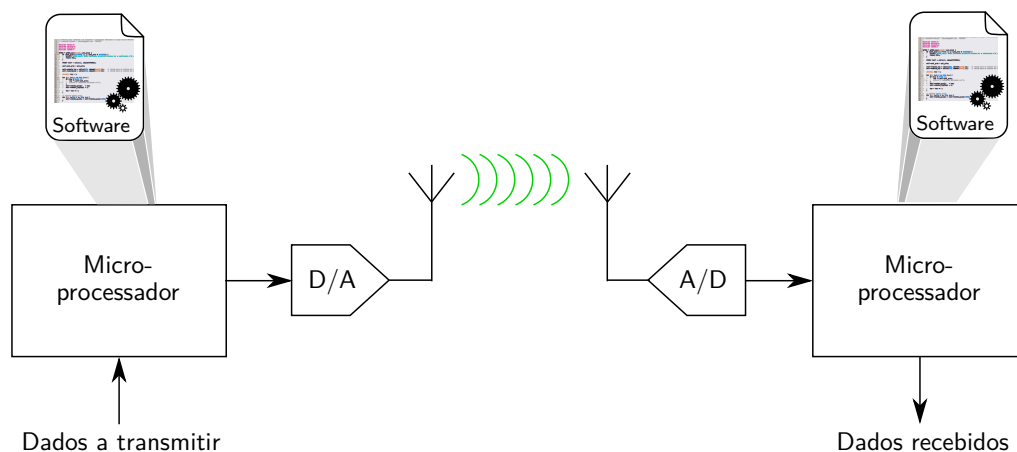
¹ O termo *rádio* é muitas vezes utilizado para descrever tanto um receptor quanto um transmissor de radiofrequência

- *Camada física* - É a camada mais baixa do modelo ISO, onde ocorrem a codificação de canal e o processamento de sinais de RF, FI² ou de banda base;
- *Rádio Definido por Software (RDS)* - Rádio no qual algumas ou todas as funções da camada física são definidas por software;
- *Forma de onda (waveform)* - Conjunto de transformações aplicadas na informação a ser transmitida e o conjunto correspondente de transformações para converter de volta os sinais recebidos em informação.

A Figura 2 apresenta o RDS ideal, composto por um microprocessador, um conversor A/D ou D/A e uma antena. Esta construção é considerada o *objetivo final* do RDS, onde o conversor A/D ou D/A é de faixa muito larga e fica o mais próximo possível da antena. O dispositivo de processamento é um GPP (*general purpose processor* - processador de propósito geral), totalmente programável e com alto poder de processamento. Esta idealização permitiria a construção de um rádio universal, capaz de receber ou transmitir virtualmente qualquer tipo de sinal em qualquer faixa de frequência.

Os obstáculos a se chegar no RDS ideal são: taxa de amostragem e resolução dos conversores A/D e D/A; consumo de energia e tamanho (importantes para aplicações móveis); disponibilidade de um GPP capaz de processar o sinal com a taxa de amostragem requerida pelo A/D e D/A; e o custo associado. Algumas aplicações, como as militares, permitem arcar com os custos de arquiteturas próximas do RDS ideal.

Figura 2: Arquitetura de RDS idealizada para a transmissão e recepção



Fonte: o autor

Uma arquitetura de um receptor³ RDS típico é apresentada na Figura 3. Sua

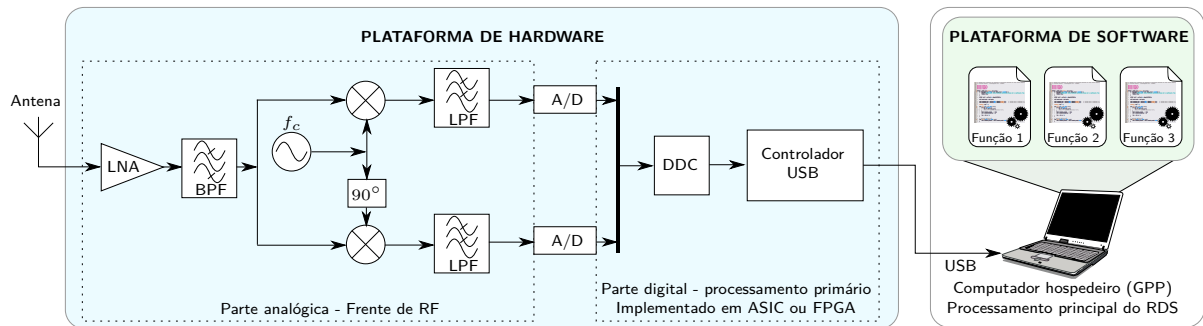
² Frequência Intermediária

³ Para um transmissor RDS, a arquitetura é semelhante, bastando inverter o fluxo do sinal, substituir o DDC por um DUC (*Digital Up-Converter*), os conversores A/D por D/A e o LNA por um PA (*Power Amplifier*)

composição básica é de uma *plataforma de hardware*⁴ e de um *computador hospedeiro* (ou *host*). Esta plataforma de hardware utiliza amplificadores, filtros e conversores de frequência *controlados por software*⁵ antes dos conversores A/D, com o objetivo de fazer a translação de parte do espectro desejado para a banda base ou mesmo para uma frequência intermediária mais baixa, havendo portanto uma drástica redução na taxa de amostragem dos conversores A/D em troca de uma pequena perda de flexibilidade, quando comparado com o RDS ideal.

Nesta arquitetura típica, logo após os conversores A/D ocorre parte do processamento digital do sinal (processamento primário). O processamento primário normalmente compreende a função DDC⁶ e geralmente é executado por um dispositivo de processamento do tipo FPGA. O FPGA não possui a facilidade e flexibilidade de programação tão grande quanto a de um GPP mas possui grande poder de processamento paralelo. As amostras são então encaminhadas para o computador hospedeiro (que contém um GPP) onde reside uma *plataforma de software* que abriga as demais funções do RDS, responsáveis pelo processamento principal do sinal.

Figura 3: Receptor RDS típico



Fonte: o autor

Quando comparada à implementação em hardware, uma função de um rádio implementada em software é vantajosa sob diversos aspectos. De acordo com Grayver (2013, p. 9), é possível enumerar os seguintes benefícios:

- Interoperabilidade - Um RDS pode ser construído para se comunicar, ou agir como ponte, entre múltiplos rádios distintos incompatíveis entre si;
- Uso eficiente dos recursos sob condições variáveis - Um RDS sob determinada condição pode alterar a sua forma de onda para maximizar uma métrica. Por exemplo, alterar

⁴ Também conhecida como *radio front-end*

⁵ Em contraste com o termo *definido por software*, aqui a função não é realizada em software, mas somente alguns de seus parâmetros podem ser alterados por software.

⁶ *Digital Down-Converter* ou Conversor de frequência digital

a modulação, codificação, ou mesmo o protocolo completo quando operando com nível fraco de bateria, com o objetivo de maximizar o tempo restante de bateria;

- Reuso oportunístico de frequências (rádio cognitivo) - Um RDS pode monitorar o espectro e utilizar as faixas subutilizadas de forma inteligente;
- Redução da obsolescência (futuro assegurado) - O RDS pode ser atualizado no campo para suportar os padrões de comunicações mais modernos disponíveis;
- Baixo custo - Como um mesmo rádio (hardware) poder ser utilizado em diversas aplicações, a economia de escala pode reduzir o custo destes dispositivos;
- Pesquisa e desenvolvimento - Um RDS pode ser utilizado na implementação de diversas formas de onda para análise de performance em tempo real. Estudos com diversas variações de parâmetros podem ser conduzidos muito mais rapidamente (e geralmente com mais fidelidade) do que através de simulações, havendo ainda a possibilidade da interação *real* do RDS em desenvolvimento com rádios existentes.

Algumas desvantagens apontadas pelo mesmo autor são:

- Custo - Mesmo apontada com uma potencial vantagem, em casos como o de um controle remoto para abertura de portões de garagem, onde o rádio utilizado é muito simples, de aplicação específica, com chips produzidos em larga escala, o RDS geralmente não é competitivo;
- Consumo de energia - Os algoritmos de processamento digital complexos do RDS requerem dispositivos de processamento (FPGA por exemplo) que (ainda) consomem muita energia. Quando comparados aos rádios digitais tradicionais implementados em ASICs⁷, o RDS consome facilmente 10 vezes mais energia. O uso de A/D e D/A com faixa larga também consome mais energia do que os de faixa estreita normalmente utilizados em rádios tradicionais.

Com todas estas virtudes, o mercado de dispositivos que utilizam RDS é crescente. De acordo com o relatório de mercado da Mobile Experts (2011), em 2011 havia a previsão de venda de mais de 1,1 bilhão de RDS naquele ano, principalmente nos segmentos de terminais móveis celulares, infraestrutura móvel, comunicações militares e rádios de segurança pública. No segmento de infraestrutura móvel, mais de 93% do mercado utilizava a tecnologia RDS em seus equipamentos.

⁷ *Application Specific Integrated Circuits* - ASIC: Circuitos integrados de aplicação específica, não configuráveis

1.2 Evolução do RDS

A história do RDS começa com pesquisas militares nos Estados Unidos na década de 1970. Na década de 1990, o projeto *SpeakEasy* utilizou o RDS para permitir a comunicação centralizada com diversos rádios militares legados que usavam diferentes formas de onda (LACKEY; UPMAL, 1995). Em 1992 o artigo pioneiro de Mitola (1992), intitulado *Software radios: Survey, critical evaluation and future directions* abriu as portas do mundo para o RDS, seguido em 1995 por outro artigo importante do mesmo autor *The Software Radio Architecture* (MITOLA, 1995).

Em 1996 foi criado o Fórum *Modular Multifunction Information Transfer System (MMITS)*⁸ por requisição das forças armadas dos Estados Unidos, como uma associação da indústria de telecomunicações focada no avanço e desenvolvimento do RDS.

No final da década de 1990, o RDS começou a se espalhar fora das aplicações militares. A aplicação comercial mais promissora considerada naquele momento eram as redes de telefonia celulares, para permitir a construção de estações rádio base interoperáveis com diversos padrões.

Em 2003, a empresa estadunidense *Vanu Inc.* lançou a *Anywave*, a primeira estação rádio base (ERB) GSM com todo o processamento de sinal executado inteiramente em software por servidores *de prateleira*, com processadores Pentium 3,4 GHz rodando o sistema operacional Linux. Em 2006 a mesma empresa lançou a primeira ERB que processava dois padrões celulares simultaneamente, o GSM e o CDMA (BOSE, 1999; KUMAGAI, 2007).

O acesso facilitado do RDS à comunidade acadêmica se deu no início do século 21, com a disponibilidade de plataformas de software e hardware acessíveis. Em 2001 Eric Blossom lançou o *GNU Radio*⁹, uma plataforma de software para RDS. Este software livre e de código aberto permite a construção de rádios RDS e a execução em um computador de mesa convencional (WIRED MAGAZINE, 2006).

Em 2004 Matt Ettus, que trabalhou com Eric Blossom no GNU Radio, fundou a empresa Ettus Research (ARS TECHNICA, 2012). Em 2005 lançou seu primeiro produto, o USRP1, a primeira versão do *Universal Software Radio Peripheral - USRP*, considerada a primeira plataforma de hardware comercial de baixo custo para RDS fácil de utilizar (CASS, 2006). Grande parte do processamento do RDS era executada pelo computador hospedeiro, que contava com o auxílio de um FPGA para operações intensivas como a filtragem inicial do sinal amostrado e a decimação. Em 2010 a Ettus Research foi adquirida pela National Instruments.

Em 2008 surgiu o *OpenBTS*¹⁰, um projeto RDS de código aberto de uma estação

⁸ Em 1998 passou a se chamar *SDR Forum*. Seu nome foi novamente alterado em 2009 para *Wireless Innovation Forum* - <<http://www.wirelessinnovation.org>>

⁹ <<https://www.gnuradio.org>>

¹⁰ <<http://openbts.org/>>

rádio base GSM rodando sob o sistema operacional Linux (SAMRA, 2008). Naquele mesmo ano, foi instalada uma ERB GSM temporária usando o OpenBTS no festival *Burning Man* que ocorreu no deserto de Black Rock no estado de Nevada (EUA), permitindo a comunicação com sucesso entre aparelhos celulares no raio de cobertura daquela estação. O OpenBTS atualmente é utilizado em laboratórios de universidades e também para prover acesso à telefonia celular em países dos 7 continentes, incluindo diversas ilhas (IEDEMA, 2015, p. vii).

Em 2011 surgiu uma nova plataforma de RDS de baixíssimo custo¹¹, o *RTL-SDR* (RTLSDR.ORG, 2013; CASS, 2013). Apesar de ser um dispositivo apenas receptor, com especificações modestas, sua ampla disponibilidade permite a qualquer pessoa interessada experimentar a tecnologia RDS. Com o surto de interesse nesta plataforma de baixo custo, houve o desenvolvimento amplo de suporte para este dispositivo nas plataformas de softwares como o Matlab (SERGIENKO, 2014) e o GNU Radio.

O RDS revolucionou as telecomunicações nos últimos ≈ 20 anos, principalmente pelos avanços importantes nos conversores A/D e D/A, nas tecnologias de processamento de sinais, e pela evolução das ferramentas de software e plataformas de hardware. Assim, a tecnologia RDS está conseguindo cumprir sua promessa em diversos sistemas de comunicação e ferramentas prototipagem, se tornando uma opção principal, poderosa e acessível para os sistemas de telecomunicação das próximas gerações.

1.3 Dispositivos de processamento

Um RDS pode ser classificado de acordo com o dispositivo de processamento programável (chamado daqui para frente simplesmente de *processador*) empregado para implementar as funções de rádio. Há basicamente 3 tipos de processadores a serem considerados para um RDS:

- DSP (*Digital Signal Processor* ou Processador digital de sinais) - São processadores otimizados para o processamento de sinais, contando com um conjunto eficiente de instruções para sua aplicação, por exemplo instruções do tipo *multiply-accumulate* e aritmética de saturação, úteis na filtragem digital de sinais, por exemplo;
- FPGA (*Field Programmable Gate Array* ou Arranjo de portas programáveis em campo) - Possui grande número de elementos lógicos (portas lógicas, flip-flops, memórias) reconfiguráveis no nível de seus elementos e interconexões. Possui grande poder de processamento paralelo;

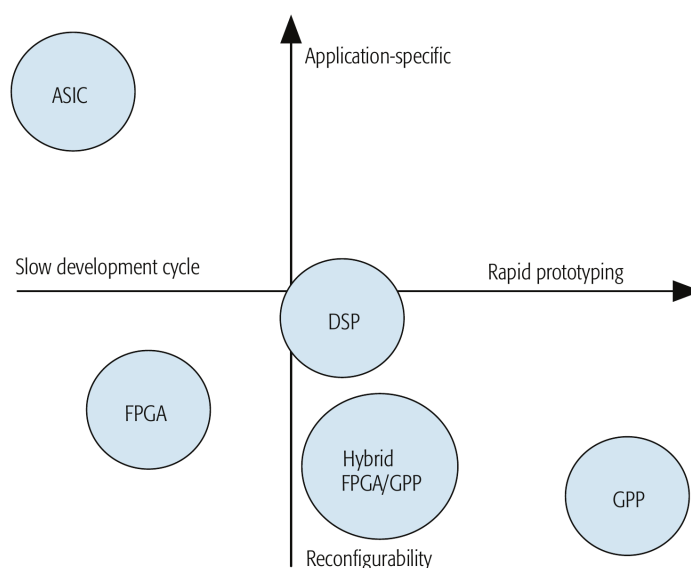
¹¹ Não é possível fazer uma comparação direta do custo com outras plataformas de RDS devido às funcionalidades distintas, mas, conforme a Tabela 2, o RTL-SDR é 14 vezes mais barato que a opção mais próxima

- GPP (*General Purpose Processor* ou Processador de propósito geral) - É a categoria, por exemplo, dos processadores utilizados nos computadores de mesa (como os Intel Core ou AMD Ryzen), ou os utilizados em *smartphones* (geralmente da família ARM). É projetado para executar uma vasta gama de softwares com diferentes requisitos, não sendo portanto otimizado especificamente para uma determinada aplicação. Atualmente seu conjunto de instruções engloba algumas instruções antes exclusivas dos DSPs. Possui amplo suporte de sistemas operacionais convencionais, com grande flexibilidade de programação, possibilidade de quantidade de memória e conectividade abundante com o mundo externo.

O ASIC (*Application Specific Integrated Circuit* ou Circuito integrado de aplicação específica), por não ser reprogramável, não é considerado como opção para o RDS, mas é citado como referência pelo baixo consumo de energia e alto desempenho de processamento, por ser projetado para uma função específica.

A Figura 4, reproduzida de Sklivanitis et al. (2016), classifica os processadores em um gráfico de nível de reconfiguração (eixo vertical, mais reconfigurável está na parte inferior) versus velocidade de desenvolvimento (eixo horizontal, o mais rápido de se desenvolver está à direita). O tipo *Hybrid FPGA/GPP* é um FPGA com parte de sua lógica programada para funcionar como um GPP. A Tabela 1 apresenta o desempenho, consumo de energia, a flexibilidade de programação e o custo de cada tipo de processador.

Figura 4: Comparativo de processadores para o RDS: nível de reconfiguração x velocidade de desenvolvimento



Fonte: Sklivanitis et al. (2016)

Para atender aos diferentes requisitos de flexibilidade, latência, memória e neces-

sidade de poder de processamento de cada função do RDS, pode-se dividir a execução das funções do RDS entre diferentes tipos de processadores, formando uma arquitetura heterogênea. Por exemplo, um FPGA pode ser utilizado para as funções com alta taxa de dados, que se beneficiam do processamento paralelo, e um GPP para as demais funções, em uma arquitetura semelhante a da Figura 3.

Tabela 1: Comparativo dos processadores para RDS

Tecnologia	Desempenho	Consumo de energia	Flexibilidade	Custo
ASIC	★★★★★	★☆☆☆☆	☆☆☆☆☆	\$ ^a
FPGA	★★★★☆	★★★★★	★★★★☆	\$\$\$
DSP	★★★☆☆	★★★☆☆	★★★★☆	\$\$\$
GPP	★★☆☆☆	★★★☆☆	★★★★★	\$

Fonte: Palicot (2011, p. 312)

^a depende da quantidade

1.4 Plataformas de Hardware

A plataforma de hardware é o dispositivo que fica entre a antena e o computador hospedeiro (que normalmente executa a maior parte do processamento do RDS). A plataforma de hardware contém componentes analógicos e digitais e normalmente implementa o processamento primário de sinal do RDS, que geralmente compreende a filtragem, amplificação, conversão de frequência, conversão A/D e/ou D/A, e DDC e/ou DUC. O processamento digital dentro desta plataforma normalmente é executado por um FPGA, conforme Figura 3. A plataforma de hardware pode possuir diversos formatos e tamanhos, como ser montável em *rack*, em mesa, ou mesmo ser portátil.

A Tabela 2 apresenta um comparativo de diversas plataformas de RDS comerciais disponíveis atualmente. É possível observar que, além das diversas faixas sintonizáveis, larguras de faixa e resolução das amostras, há opções para recepção somente, recepção e transmissão chaveada (*half-duplex*), simultânea (*full-duplex*) e também opções com mais de um canal de recepção e transmissão, úteis para o desenvolvimento de rádios MIMO¹².

Diante das opções listadas, o RTL-SDR é o de menor custo. Na próxima seção, esta plataforma será apresentada detalhadamente.

¹² *Multiple-Input, Multiple-Output* - Múltiplas entradas e múltiplas saídas

Tabela 2: Comparativo de dispositivos RDS

Plataforma de Hardware RDS	Faixa sintonizável (MHz)	Largura de Faixa (MHz)	Resolução A/D (Bits)	Canais de Recepção	Canais de Transmissão	Custo (US\$)
RTL-SDR ^a	24—1766	2,4	8	1	0	7—21
Airspy Mini/R2 ^b	24—1800	10	12	1	0	99/169
SDRPlay RSP1/RSP2 ^c	0,01/0,001—2000	8	12	1	0	100/169
ADALM-PLUTO ^d	325—3800	20	12	1	1	149
Funcube Pro+ ^e	0,15—260 410—2050	0,192	16	1	0	200
HackRF One ^f	1—6000	20	8	1	1 (half-duplex)	299
LimeSDR ^g	0,1—3800	61	12	2	2	299
BladeRF x40/x115 ^h	300—3800	28	12	2	2	420/650
USRP B200/B210 ⁱ	70—6000	56	12	1/2	1/2	745/1216
USRP N200/N210 ⁱ	0—6000	25	14	1	1	1673/1896
USRP X300/X310 ⁱ	0—6000	120	14	2	2	4296/5290
Crimson TNG ^j	0—6000	325	16	4	4	13500

Fonte: o autor

^a Vários modelos/fabricantes, por exemplo: <<http://www.nooelec.com/store/sdr/sdr-receivers/nesdr-smart-sdr.html>> e <<https://goo.gl/7WsY9r>>

^b <<http://airspy.com>>

^c <<http://www.sdrplay.com>>

^d <<http://www.analog.com/plutosdr>>

^e <<http://www.funcubedongle.com>>

^f <<https://greatscottgadgets.com/hackrf>>

^g <<http://www.limemicro.com/products/software-defined-radio>>

^h <<https://nuand.com>>

ⁱ <<https://www.ettus.com>>

^j <<https://www.pervices.com>>

1.4.1 RTL-SDR

O RTL-SDR é um receptor de RDS de baixo custo baseado no chip *Realtek RTL2832U*. É um dispositivo fabricado por diversas empresas, e dependendo do modelo considerado pode possuir diferenças na sua faixa sintonizável, na estabilidade de frequência e na qualidade construtiva. As principais características do RTL-SDR mais comum disponível

no mercado são:

- Faixa sintonizável: 24 MHz a 1766 MHz
- Impedância da entrada de antena: 75 Ohm
- Resolução: 8 bits
- Taxa de amostragem ajustável, até 2,4 MSPS^{13,14}

O RTL-SDR é peculiar pois ele não foi originalmente projetado e comercializado como um receptor RDS (STEWART et al., 2015, p. 10), mas como um adaptador USB para recepção de DVB-T (*Digital Video Broadcasting - Terrestrial*, padrão europeu de TV digital), DAB (*Digital Audio Broadcasting*, padrão europeu de rádio digital) e rádio FM no computador. Desta forma, o RTL-SDR é oficialmente apenas um dispositivo para o usuário comum assistir TV ou ouvir rádio no computador. Seu baixo custo pode ser explicado pela sua produção em massa, considerando a demanda pelo uso como receptor DVB-T.

O receptor é fornecido com os *drivers* para o sistema operacional Windows. O mesmo possui código fonte fechado, mas através de engenharia reversa foi descoberto que, dentre os tipos de sinais possíveis de se receber oficialmente (DAB, FM e DVB-T), apenas a demodulação do DVB-T é executada diretamente pelo hardware; a demodulação de FM e DAB são executadas pelo computador através de RDS. A engenharia reversa revelou como transferir, via USB em tempo real, as amostras brutas do conversor A/D interno ao chip RTL2832U¹⁵, permitindo o seu uso como um receptor RDS genérico. Cass (2013) atribui o crédito desta engenharia reversa ao engenheiro finlandês Antti Palosaari em 2012. Porém RTLSDR.ORG (2013) atribui o crédito ao Eric Fry em 2011.

Posteriormente à descoberta, Steve Markgraf, Dimitri Stolnikov, e Hoernchen desenvolveram a biblioteca de software *librtlsdr* e outros softwares para o sistema operacional Linux, que permitem o uso fácil do RTL-SDR como receptor RDS (OSMOCOM, 2014). Estes softwares foram publicados com a licença de código aberto GNU GPL e são a base para inúmeros trabalhos desenvolvidos com este hardware. Desde então o RTL-SDR se popularizou no meio científico, entre radioamadores e entusiastas do rádio definido por software, principalmente pelo seu baixo custo e ampla faixa de frequências sintonizáveis.

Uma foto do RTL-SDR é mostrada na Figura 5, com os principais componentes em destaque. A Figura 6 mostra o diagrama de blocos internos do RTL-SDR, especificamente

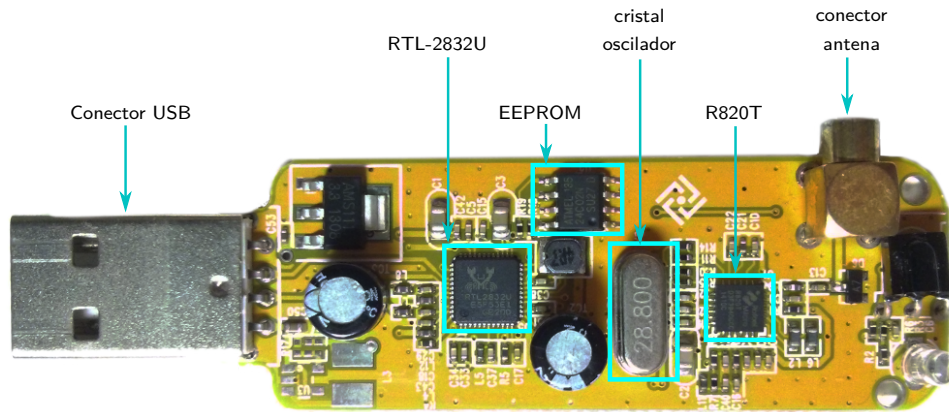
¹³ MSPS: *Megasamples per second* — milhões de amostras por segundo

¹⁴ A taxa de amostragem máxima possível é de 3,2 MSPS, porém acima de 2,4 MSPS geralmente há perda de amostras

¹⁵ Como será detalhado posteriormente, as amostras transferidas via USB não vêm diretamente do conversor A/D interno; as amostras antes passam por um processamento primário no chip RTL2832U (ASIC), na forma de funções de RDS controladas por software

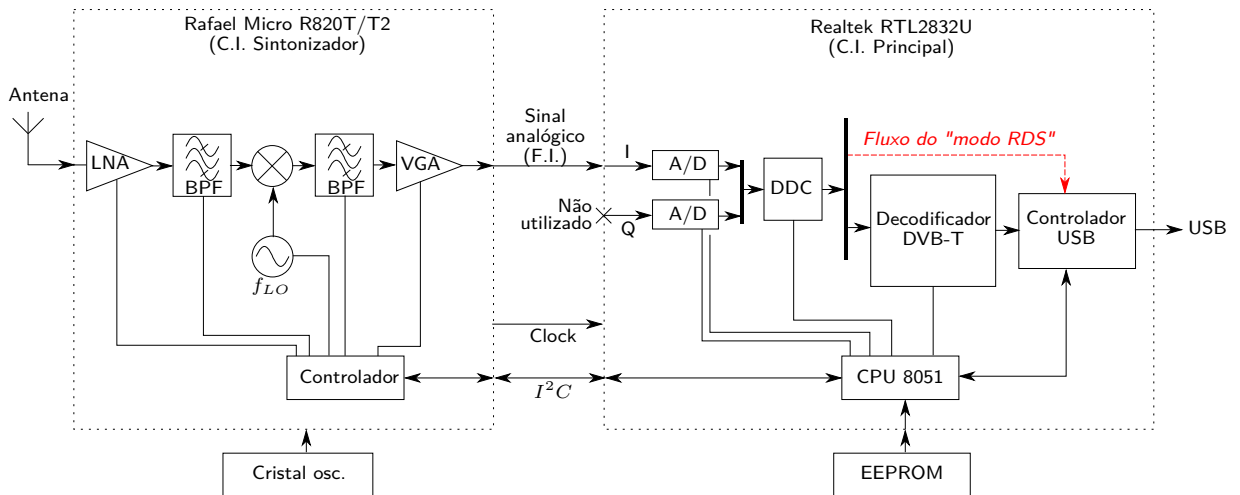
a versão equipada com o chip sintonizador R820T ou R820T2. A Figura 7 mostra a versão equipada com o chip sintonizador E4000. As figuras 8 e 9 mostram o diagrama esquemático do circuito de um RTL-SDR, desenhado a partir da sua placa de circuito impresso através de engenharia reversa, por Toshi (2014).

Figura 5: Placa de circuito impresso do RTL-SDR



Fonte: o autor

Figura 6: Diagrama de blocos do RTL-SDR com sintonizador R820T/T2

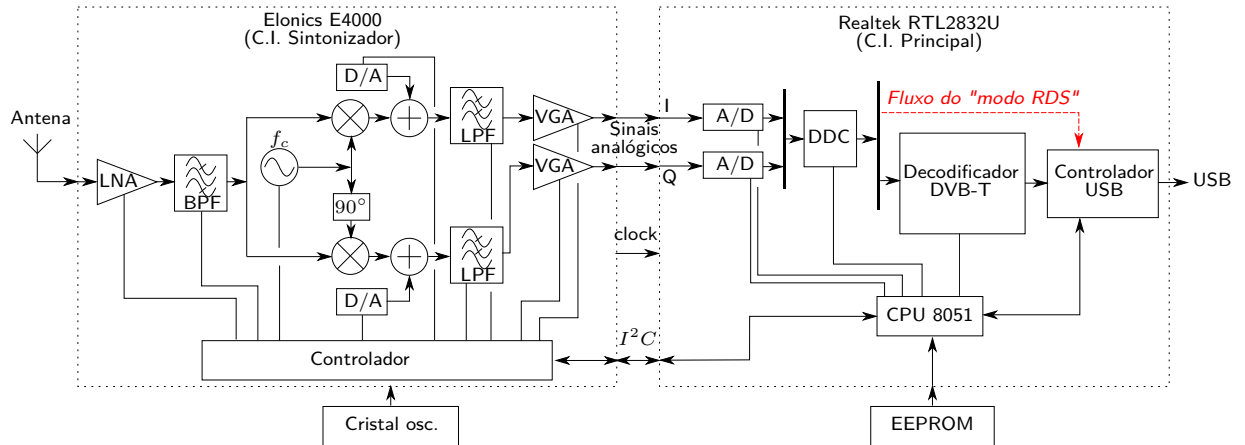


Fonte: o autor

É importante destacar que o decodificador DVB-T interno ao RTL2832U possui função fixa e não pode ser utilizado em nenhuma etapa da decodificação do sinal ISDB-T, mesmo onde um padrão de TV digital coincide com o outro. Neste desenvolvimento o bloco interno DVB-T é completamente desabilitado, e as amostras em fase e quadratura do sinal sintonizado são enviadas diretamente ao computador via USB, para processamento.

O RTL-SDR é muitas vezes comercializado sem marca ou com *marca genérica* (principalmente os fabricados na China), possuindo diferentes modelos com variações no

Figura 7: Diagrama de blocos do RTL-SDR com sintonizador E4000



Fonte: o autor

seu circuito de entrada (filtros e sintonizador), na qualidade construtiva (como blindagem eletromagnética) e na estabilidade do seu oscilador interno. A Tabela 3 mostra os diferentes sintonizadores encontrados em dispositivos RTL-SDR e as suas faixas de frequências sintonizáveis. O RTL-SDR mais comumente encontrado no mercado emprega o chip sintonizador *Rafael Micro R820T*, o mesmo utilizado neste trabalho.

Tabela 3: Comparativo de sintonizadores do RTL-SDR

Sintonizador	Faixa Sintonizável
Elonics E4000	52–1100 MHz e 1250–2200 MHz
Rafael Micro R820T	24–1766 MHz
Rafael Micro R828D	24–1766 MHz
Fitipower FC0013	22–1100 MHz
Fitipower FC0012	22–948.6 MHz
FCI FC2580	146–308 MHz e 438–924 MHz

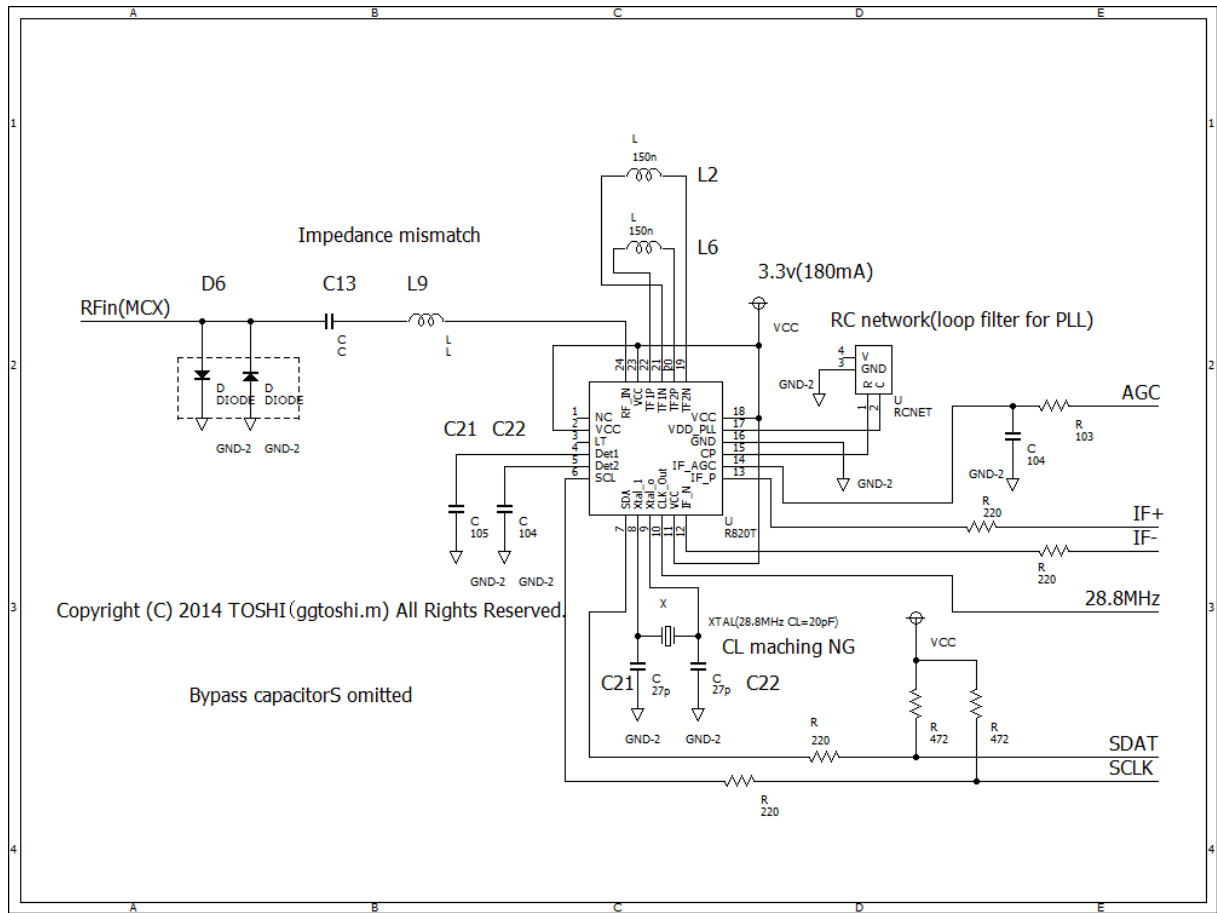
Fonte: Osmocom (2014)

Conforme RTL-SDR.COM (2016), não existe folha de dados abertamente disponível para o chip RTL2832U. A mesma é somente disponibilizada para fabricantes mediante a assinatura de um acordo de confidencialidade. Já a folha de dados dos sintonizadores R820T e do E4000 estão disponíveis na internet.

1.4.1.1 Funcionamento do RTL-SDR

Basicamente, o RTL-SDR é fisicamente composto por um C.I. sintonizador, pelo C.I. principal RTL2832U e pelos sistemas de clock e de alimentação. Conforme as figuras 6

Figura 8: Diagrama esquemático do circuito de um RTL-SDR - Parte 1: sintonizador R820T

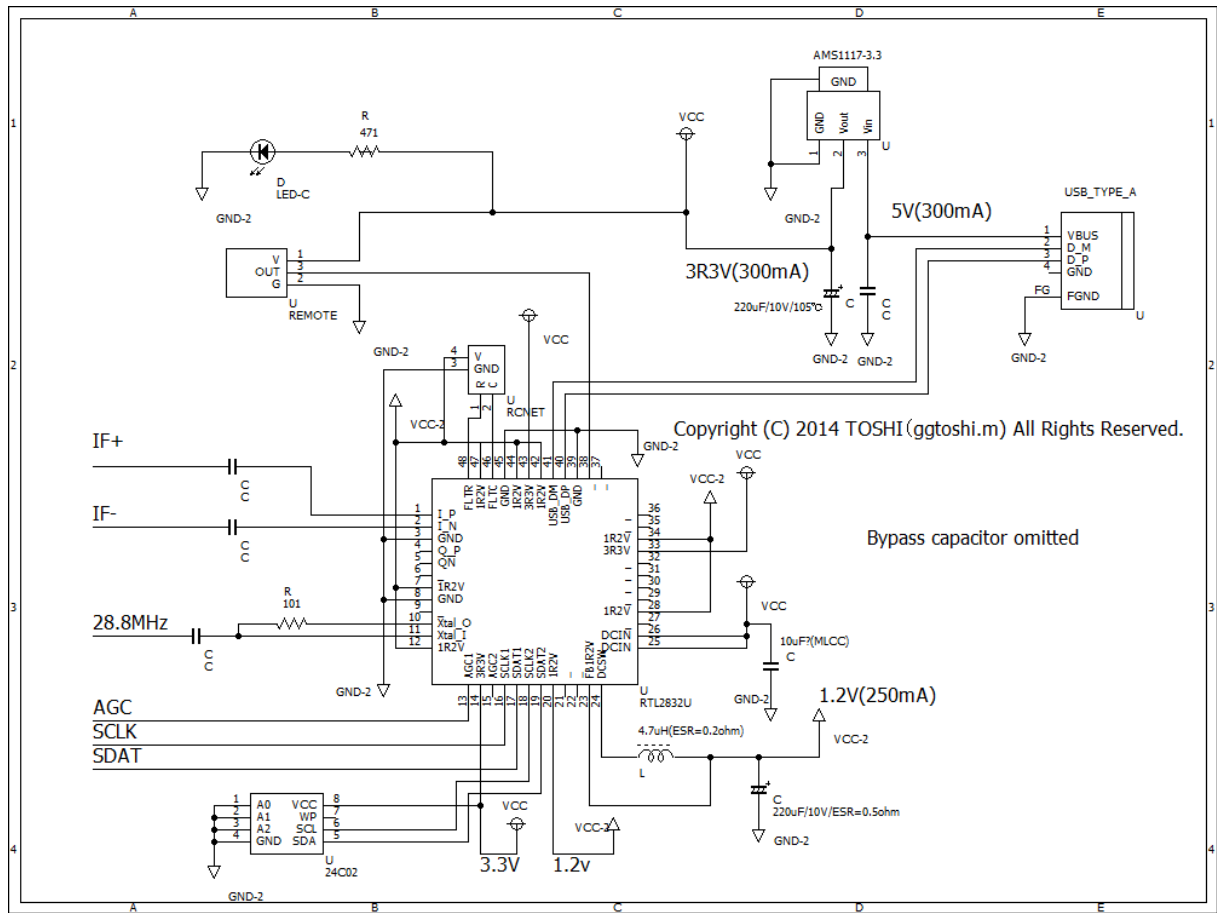


Fonte: Toshi (2014)

e 7, há diversas funções desta plataforma de hardware distribuídas entre o C.I. sintonizador e o C.I. principal. Segue a função de cada bloco:

- LNA (*Low Noise Amplifier*, Amplificador de baixo ruído): Primeira etapa de amplificação do receptor, com baixo ruído e ganho controlável por software;
- VGA (*Variable Gain Amplifier*, Amplificador de ganho variável): Amplificador com ganho controlável por software;
- BPF (*Band-Pass Filter*, Filtro passa-faixa): Filtro com frequência central e largura de faixa controláveis por software;
- LPF (*Low-Pass Filter*, Filtro passa-baixa): Filtro com frequência de corte controlável por software;
- f_{LO} : Oscilador local, com frequência controlável por software;

Figura 9: Diagrama esquemático do circuito de um RTL-SDR - Parte 2: C.I. principal RTL2832U



Fonte: Toshi (2014)

- Conversor de frequência ou *Mixer*: Faz o batimento entre o sinal do oscilador local (de frequência f_{LO}) e o sinal recebido (de frequência central f_{RF}), deslocando o centro do espectro do sinal recebido para a soma ($f_{LO} + f_{RF}$) e a diferença ($f_{LO} - f_{RF}$) das frequências;
- A/D: Conversor Analógico para Digital: Converte o sinal analógico em sua entrada em uma representação digital de 8 bits. Os conversores na entrada do RTL2832U operam com taxa de amostragem fixa de 28,8 MSPS;
- D/A: Conversor Digital para Analógico. Utilizado no sintonizador E4000 para gerar uma tensão contínua que permite compensar (eliminar) a frequência zero em cada um dos ramos I/Q do circuito;
- DDC (*Digital Down-Converter*, Conversor de frequência digital): Faz o deslocamento em frequência do espectro do sinal de entrada para frequências em torno de zero; faz a filtragem passa-baixas e reduz a taxa de amostragem. Opera totalmente no domínio digital;

- Decodificador DVB-T: Bloco mais complexo do RTL-SDR, utilizado para sua função original como receptor DVB-T. Este bloco é desativado quando a funcionalidade RDS é utilizada;
- Controlador USB: Faz a transferência de dados entre o RTL-SDR e um computador conectado a ele através de uma porta USB.

As seções a seguir descrevem o funcionamento do sintonizador e do C.I. principal. Para o sintonizador, serão abordados as duas variações mais comuns no mercado: o E4000 e o R820T/T2.

1.4.1.2 Sintonizador E4000

O E4000, fabricado pela empresa escocesa *Elonics*, faz a amplificação, filtragem e conversão do sinal passa-faixa de entrada para um sinal passa-baixa complexo (sinal em quadratura, I/Q), centralizado na frequência zero. Esta arquitetura é conhecida como *Zero-IF*. Este sintonizador, de acordo com a folha de dados de seu fabricante (ELONICS, s. d.), é capaz de sintonizar frequências na faixa 64–1700 MHz, mas na prática foi constatado que é possível sintonizar uma faixa mais ampla: 52–2200 MHz, porém com uma lacuna entre 1100 MHz a 1250 MHz.

O sinal proveniente da antena é amplificado pelo amplificador de baixo ruído, passa por um filtro passa-faixas e depois é dividido em dois ramos I e Q, onde o sinal de cada ramo é submetido ao batimento com o oscilador local (no ramo Q o oscilador local é previamente defasado em 90°). Os sinais então passam pelos compensadores de frequência zero (compostos por conversores D/A e somadores), cujo objetivo é eliminar a componente de corrente contínua do sinal. Os sinais passam por filtros passa-baixa, e por fim são levados aos amplificadores de ganho variável. Os sinais analógicos I e Q são então disponibilizados para fora do C.I.

Quase todos os blocos possuem parâmetros que podem ser controlados eletronicamente, como por exemplo, o oscilador local possui frequência ajustável em uma ampla faixa, os blocos LNA e VGA possuem ganhos ajustáveis, os filtros BPF e LPF possuem ajuste eletrônico da frequência de corte e do *Roll-off*. O gerenciamento destes parâmetros é feito por um controlador interno que se comunica com o meio externo através do protocolo *I²C*.

1.4.1.3 Sintonizador R820T/T2

O chip sintonizador R820T, do fabricante taiwanês *Rafael Micro*, e seu modelo mais recente R820T2 possuem uma arquitetura mas simples se comparados ao E4000: a conversão de frequência é feita para uma F.I. (Frequência Intermediária), usando apenas um *Mixer*, entregando na saída um único sinal passa-faixa. De acordo com a folha de dados

do C.I. (RAFAEL MICROELECTRONICS, s. d.), ele é capaz de sintonizar frequências na faixa 42–1002 MHz, mas na prática foi observado que a sintonia é possível na faixa de 24–1766 MHz, sem lacunas.

O sinal da antena é levado ao amplificador de baixo ruído, e depois a um um filtro passa-faixas. Depois é submetido ao batimento com o oscilador local, onde é gerado o sinal de F.I. Este sinal então passa por um filtro passa-faixa e é levado a um amplificador de ganho variável. O sinal analógico de sua saída é então disponibilizado para fora do C.I.

Como no E4000, os blocos possuem parâmetros que são gerenciados por um controlador que possui interface I^2C .

1.4.1.4 Chip principal RTL2832U

O RTL2832U do fabricante taiwanês *Realtek* (REALTEK, s. d.), é o coração do RTL-SDR. Ele possui em sua entrada dois conversores A/D com resolução de 8 bits, operando com a taxa de amostragem $f_{A/D}$ fixa de 28,8 MHz. Dependendo do tipo do sintonizador utilizado, somente um dos A/Ds é utilizado. Os sinais digitais das saídas dos conversores A/D são levados ao bloco DDC que, se configurado, faz o batimento dos sinais com dois osciladores numéricos operando em quadratura com o objetivo de reduzir a taxa (e a largura de faixa) do sinal. Estes sinais passam então por filtros passa-baixa e por fim suas taxas são reduzidas através da *decimação*.

Apesar da taxa de amostragem dos conversores A/D do RTL2832U serem fixas em $f_{A/D} = 28,8$ MHz, as amostras na saída do RTL-SDR entregues ao computador hospedeiro possuem taxa f_s configurável. De acordo com Osmocom (2017, linhas 1102 a 1116), f_s deve estar nos intervalos válidos $0,225 < f_s \leq 0,3$ e $0,9 < f_s \leq 3,2$ MSPS, que são os intervalos suportados pelo DDC do dispositivo. Geralmente o uso de taxas acima de 2,4 MSPS provocam perda de amostras.

Quando o C.I. opera no modo receptor de TV digital DVB-T, o bloco *Decodificador DVB-T* recebe as amostras I/Q da saída do DDC a uma taxa de cerca de 9 MSPS, suficiente para amostrar um canal de TV DVB-T de até 8 MHz de largura de faixa. Este bloco então faz todas as etapas de decodificação do sinal DVB-T no hardware, tais como: sincronização de frequência e de símbolo, remoção do intervalo de guarda, FFT, equalização, desentrelaçamento, decodificação Viterbi e Reed-Solomon e inversão da dispersão de energia. A saída deste bloco é um fluxo MPEG-2 TS com taxa de até 31 Mbits/s, que é entregue ao controlador USB para ser enviado ao computador, onde o vídeo e o áudio podem ser decodificados e reproduzidos. Este bloco de hardware é inflexível, não podendo ser configurado para decodificar outro padrão de TV digital, como por exemplo o ISDB-T, ou mesmo para receber um simples sinal de FM analógico.

Por outro lado, quando o C.I. é configurado para operar no modo RDS, o bloco

Decodificador DVB-T é desativado e as amostras digitais I/Q da saída do DDC são encaminhadas diretamente e de forma contínua para o controlador USB, desviando do bloco Decodificador DVB-T. Estas amostras podem ser disponibilizadas ao computador a uma taxa máxima estável de 2,4 MSPS. Cada amostra contém 8 bits do canal I e mais 8 bits do canal Q, compondo uma taxa de transferência de até 38,4 Mbits/s no barramento USB 2.0. Esta taxa é limitada pela capacidade do controlador USB interno ao C.I., e não pelos A/Ds ou pelo protocolo USB 2.0.

1.4.1.5 Sistemas de clock e de alimentação

Conforme as figuras 8 e 9, para a versão do RTL-SDR equipada com o R820T o sinal de clock é gerado por um cristal de 28,8 MHz, conectado aos pinos 8 e 9 do R820T. Este C.I. replica este clock em seu pino 10 e o envia para os pinos 10 e 11 do RTL2832U.

O cristal normalmente utilizado nas versões de custo mais baixo do RTL-SDR possui tolerância de 20 ppm, já as versão mais caras deste dispositivo empregam cristais com compensação de temperatura com tolerância de apenas 0,5 ppm.

O sinal de clock é utilizado como frequência de amostragem nos conversores A/D do RTL2832U e também serve de referência para o oscilador local contido no R820T. Portanto o desvio (erro) de frequência do clock causa desvios na frequência sintonizada e na taxa de amostragem. Em aplicações sensíveis a estes desvios, deve haver a estimativa e a compensação contínua e em tempo real do erro de frequência por software.

A alimentação do RTL-SDR é proveniente do computador hospedeiro, que fornece 5 volts via barramento USB. Esta tensão é abaixada para 3,3 V pelo regulador de tensão linear AMS1117-3.3, que alimenta os dois CIs e também a memória EEPROM 24C02. O RTL2832U requer uma segunda alimentação, de 1,2V. Esta tensão é obtida por um conversor de tensão chaveado interno ao RTL2832U, através de seus pinos 23 e 24.

Nas versões mais sofisticadas do RTL-SDR, o conversor chaveado é desabilitado e no seu lugar é utilizado um regulador de tensão linear discreto de 1,2 V. Isto elimina o ruído eletromagnético gerado pelo conversor chaveado que interfere no sinal amostrado.

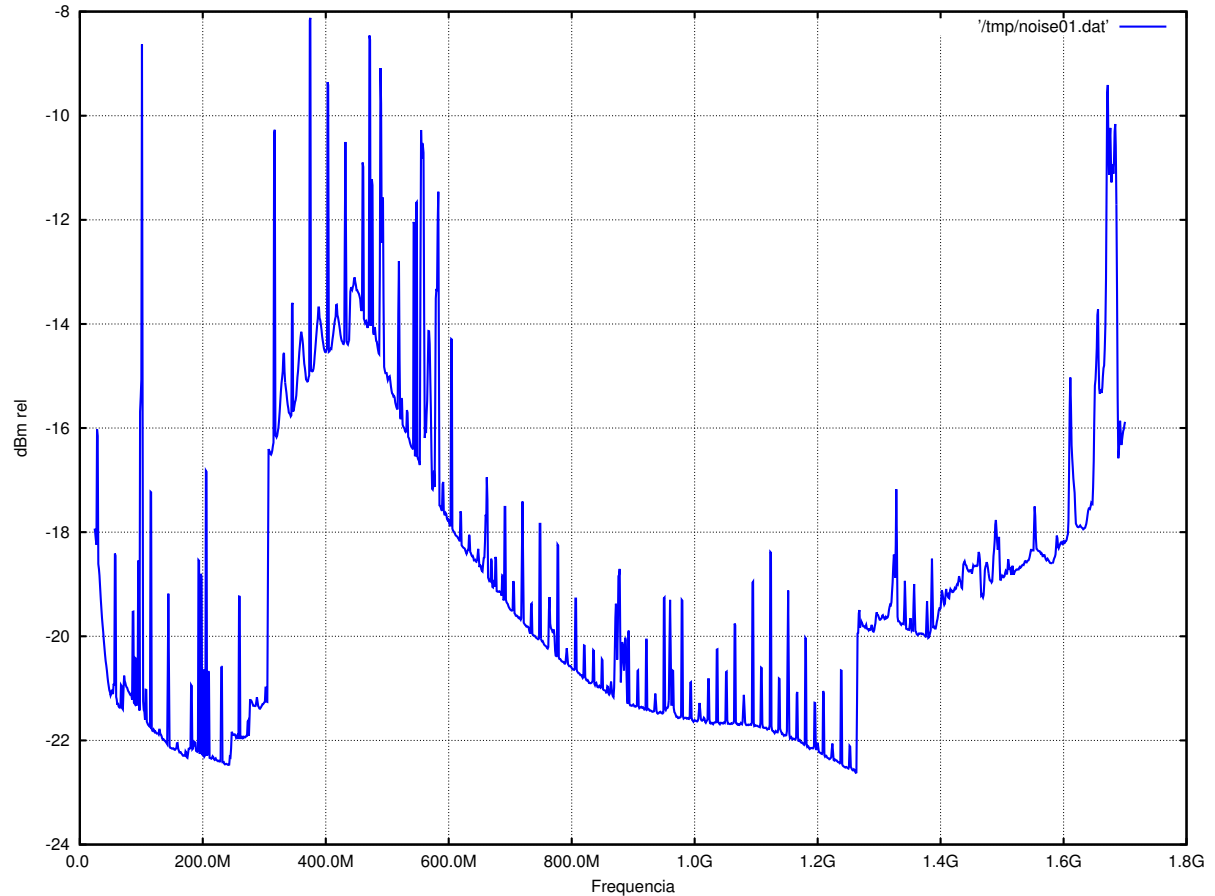
1.4.1.6 Ruído de fundo

Foi realizado um ensaio de *ruído de fundo*, com o objetivo de medir a interferência gerada pelo próprio RTL-SDR em toda a sua faixa de frequência sintonizável. Para tanto, a antena foi desconectada e o ganho configurado para o máximo, 50 dB.

A Figura 10 apresenta a magnitude relativa, em dB, do sinal sintonizado pelo RTL-SDR configurado para varrer praticamente toda a sua faixa de operação, de 24 a 1700 MHz, por 15 minutos.

É possível observar diversos picos localizados nas frequências múltiplas de 28,8 MHz — a frequência do cristal do dispositivo. Por se tratar de uma plataforma de RDS de baixo custo, isto é aceitável, mas deve ser conhecido do utilizador para não se ter surpresas ao se desenvolver um RDS nesta plataforma. Este resultado está de acordo com o encontrado por Keen (2017, seção *noise analysis*).

Figura 10: Ruído de fundo do RTL-SDR



Fonte: o autor

1.4.1.7 Estabilidade do oscilador

Outra perturbação que ocorre em um RDS é o desvio de frequência de seu oscilador. Com o objetivo de se obter a grandeza deste desvio para o RTL-SDR em mãos, foi realizado um ensaio com o receptor RDS ISDB-T descrito no Capítulo 5 sintonizado em uma emissora¹⁶ de sinal forte, durante o período de 96 horas. Não houveram erros de sincronização ou decodificação do receptor RDS ISDB-T durante este período.

O desvio de frequência $\hat{\mathcal{E}}$ (conhecido como *Carrier Frequency Offset – CFO*) entre o RTL-SDR e a emissora foi estimado pelo bloco do receptor *estimador do início do símbolo*

¹⁶ A emissora utilizada foi o canal 30 de Uberlândia — vide Tabela 18

e do desvio de frequência fracionária, cuja operação é descrita na subseção 5.2.2.2.

A Figura 11 apresenta o resultado deste ensaio. No período do ensaio, $\hat{\epsilon}$ apresentou variação de -504 Hz a 1883 Hz, o que equivale à faixa de $-0,89$ a $3,31$ ppm quando considerada a frequência da emissora (≈ 569 MHz).

De acordo com Beek, Sandell e Börjesson (1997), em um sistema OFDM o CFO máximo deve ser de 1 a 2% do espaçamento entre subportadoras, c_s , para manter a interferência entre as subportadoras (ICI - *Inter Carrier Interference*) sob controle.

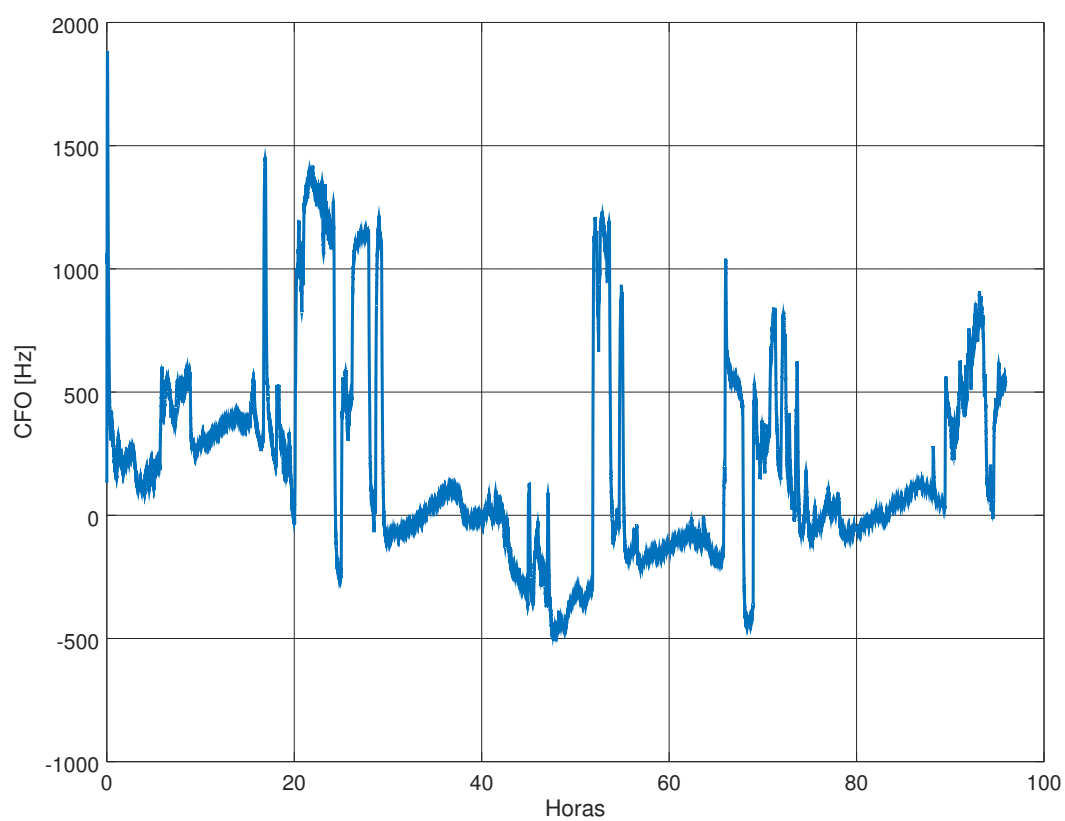
Conforme a Tabela 12, para o modo 3 do ISDB-T, $c_s \approx 992$ Hz, portanto o CFO para este sistema deve ficar no máximo entre 9,9 a 19,8 Hz. Isto é muito menor do que o CFO que ocorre na prática com o RTL-SDR, apresentado na Figura 11.

A causa deste CFO é a instabilidade do cristal oscilador do RTL-SDR, provocado principalmente pela sua variação da temperatura¹⁷. O cristal gera o *clock* do RTL-SDR, que é utilizado em diversas funções, dentre elas em seu conversor de frequência. Portanto, em um projeto de um receptor RDS de um sistema sensível ao CFO, como o OFDM, deve haver um cuidado importante na construção de um estimador e corretor de CFO adequado, principalmente quando a plataforma de hardware é de baixo custo, como o RTL-SDR.

A Figura 12 apresenta uma ampliação do gráfico da Figura 11 nos 30 minutos iniciais, mostrando que o maior desvio de CFO ocorre nos primeiros minutos de operação devido ao aquecimento do RTL-SDR.

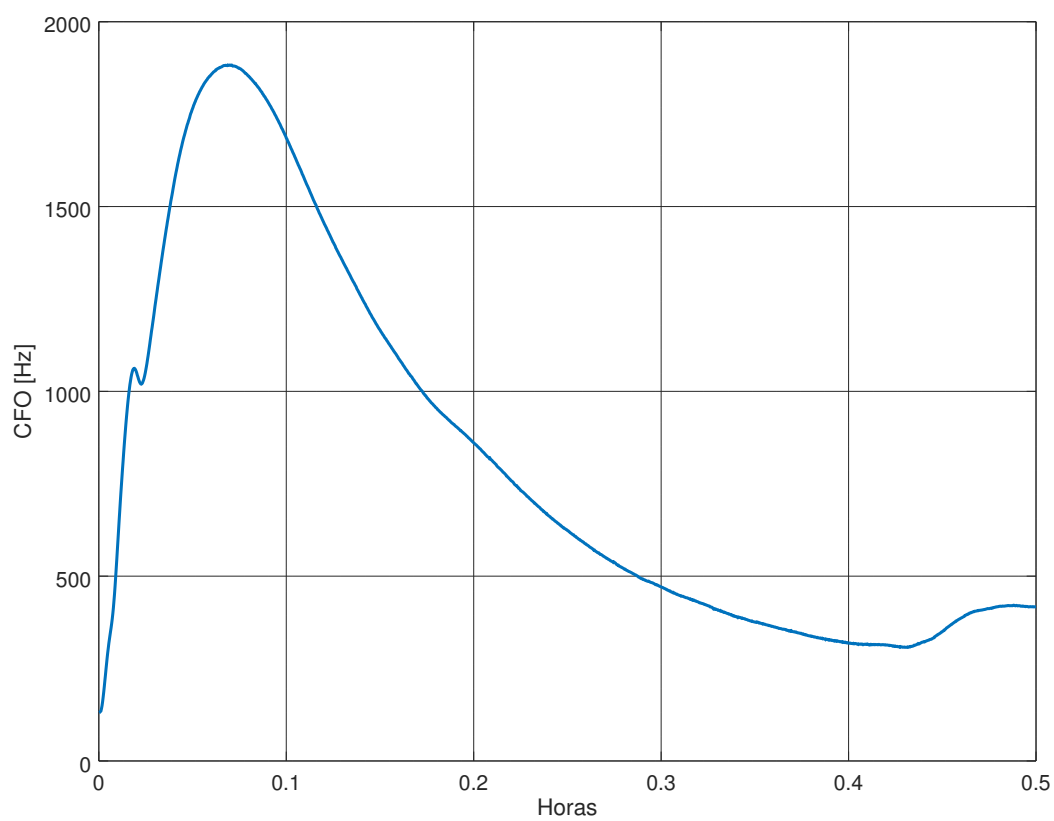
¹⁷ É fácil notar esta dependência do CFO com a temperatura ao *soprar* diretamente sobre o cristal

Figura 11: Desvio de frequência entre o RTL-SDR e a emissora



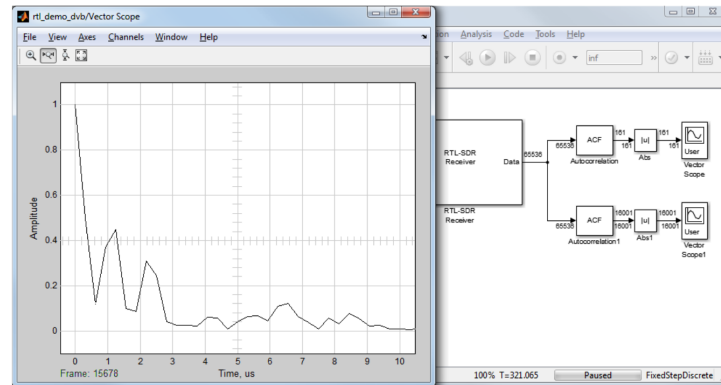
Fonte: o autor

Figura 12: Desvio de frequência entre o RTL-SDR e a emissora - Ampliação do gráfico da Figura 11 nos 30 minutos iniciais



Fonte: o autor

Figura 14: Matlab/Simulink como plataforma de RDS

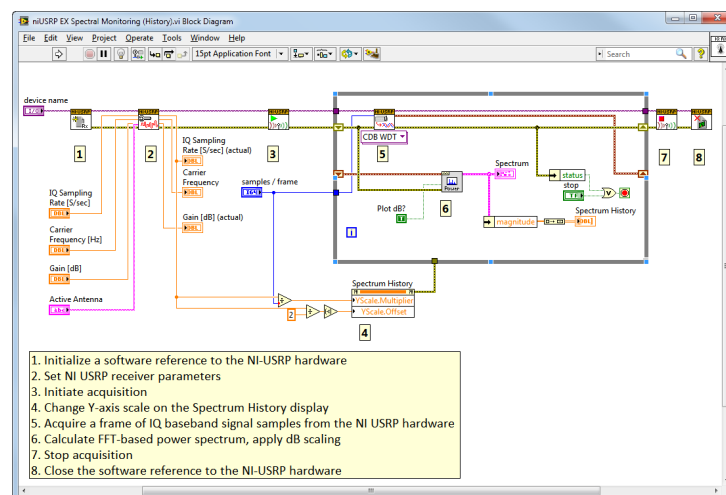


Fonte: Sergienko (2014)

1.5.3 LabVIEW

O LabVIEW é uma plataforma de desenvolvimento proprietária para a programação gráfica, desenvolvida pela *National Instruments*. Para o RDS, tem função e uso similares ao Matlab/Simulink. A Figura 15 apresenta uma tela de exemplo do LabVIEW sendo utilizado para a programação de um RDS.

Figura 15: LabVIEW para o RDS



Fonte: Rutgers University School of Engineering (s. d.)

1.5.4 API

É possível programar um RDS sem o uso de uma plataforma de software específica, como as apresentadas nas seções anteriores. Isto pode ser feito através de uma API (*Application Programming Interface* ou Interface de programação de aplicações) que tem acesso direto à plataforma de hardware RDS utilizado.

Por exemplo, para a plataforma de hardware RTL-SDR, há a biblioteca *librtlsdr* que disponibiliza uma API em C com diversas funções para sintonia, escolha da taxa de amostragem, ganho e leitura das amostras do A/D. Através desta biblioteca é possível construir um software que acessa facilmente as amostras obtidas pelo RTL-SDR, e deste ponto em diante, constrói-se (ou se utiliza de outras bibliotecas existentes) as funções do RDS.

O receptor RDS para a TV digital ISDB-T 1seg desenvolvido neste trabalho utilizou esta API do RTL-SDR mencionada.

2 Sistemas OFDM

A crescente demanda por conteúdos interativos e multimídia sem-fio requer sistemas de telecomunicações com grande capacidade de transmissão. Como exemplo, redes WiFi atualmente chegam a 867 Mb/s ¹, a rede móvel LTE possui taxa de até 300 Mb/s ² e um fluxo de vídeo em alta definição comprimido requer 18 Mb/s ³.

Uma capacidade de transmissão elevada pode ser obtida usando uma modulação com maior número de bits por símbolo (b) e/ou símbolos de menor duração (T_s), já que a capacidade de transmissão é dada por $R = b/T_s$ [bits/s] ⁴. A variável b é limitada principalmente pela relação sinal-ruído na entrada do receptor, e em sistemas típicos atinge 6 (para 64QAM) ou 8 (para 256QAM). O período do símbolo T_s não pode ser arbitrariamente pequeno principalmente devido ao problema causado pela *propagação multipercurso*.

Muitos sistemas de telecomunicações atuais, como o ISDB-T, utilizam o *Orthogonal frequency-division multiplexing* – OFDM (Multiplexação por Divisão em Frequências Ortogonais). O OFDM possui diversas virtudes que o destacam especialmente em canais sujeitos à propagação multipercurso severa.

Este capítulo é dividido em duas seções principais, a seção *Propagação Multipercurso* que trata do principal motivador para o emprego do OFDM, e a seção *OFDM* onde são abordados seus fundamentos, propriedades e a sua implementação, com ênfase no sistema de TV digital ISDB-T.

2.1 Propagação Multipercurso

O estudo e caracterização do canal são etapas muito importante em um projeto de sistemas de telecomunicações, pois norteiam diversos requisitos de projeto, tais como: a escolha da modulação ideal, a necessidade de equalizadores, o uso de intercaladores temporais/em frequência, a escolha de códigos corretores de erro.

O modelo clássico do canal AWGN (*Additive White Gaussian Noise* ou Ruído branco Gaussiano aditivo) não é suficiente para sistemas de telecomunicações digitais faixa larga, pois não consegue explicar grandes desvios na performance quando a propagação não se dá através de um caminho direto entre a antena do transmissor e a antena do

¹ Padrão IEEE 802.11ac, usando fluxo espacial simples.

² Downlink, Modulação 64QAM, taxa de codificação 3/4, MIMO 4x4

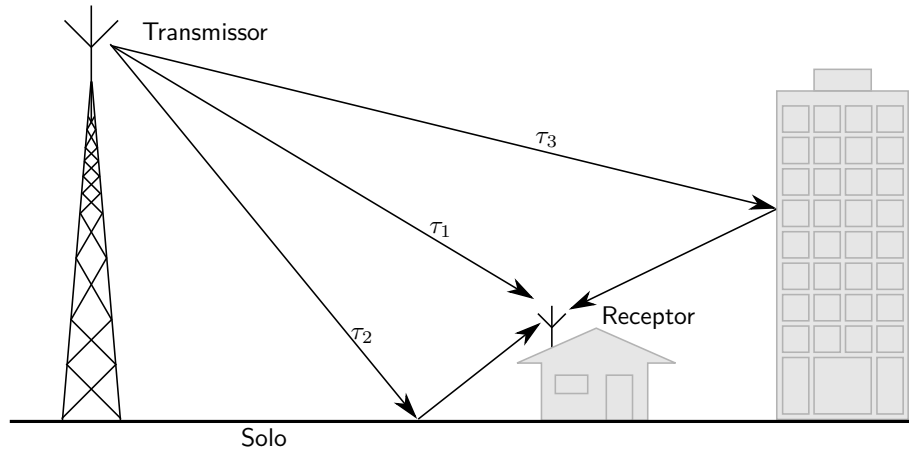
³ Sistema ISDB-T com 13 segmentos modulados em 64QAM, taxa de codificação convolucional 3/4 e intervalo de guarda de 1/8.

⁴ Considerando modulação de portadora única, sem codificação.

receptor (SKLAR, 2001, p. 945).

A propagação multipercurso ou *multipath propagation* é um fenômeno que ocorre quando o sinal de radiofrequência (RF) transmitido encontra diversos percursos com diferentes comprimentos d até o receptor, em razão de reflexões, refrações e difrações em obstáculos como edificações, solo, corpos d'água, vegetação, veículos, montanhas, etc, conforme ilustrado na Figura 16.

Figura 16: Efeito do multipercurso



Fonte: o autor

Neste sentido, o sinal na entrada do receptor $r(t)$ é a soma de diversas cópias do sinal transmitido $s(t)$, cada uma com atraso igual ao tempo de propagação do sinal $\tau = d/c$, sendo d o comprimento do percurso e c a velocidade de propagação da onda eletromagnética no meio ($\approx 3 \times 10^8$ m/s). Desconsiderando o ruído, o sinal $r(t)$ pode ser descrito matematicamente através da convolução de $s(t)$ com a resposta impulsiva do canal $h_c(t)$, conforme a Equação 2.1.

$$r(t) = s(t) * h_c(t) \quad (2.1)$$

O modelo para a resposta impulsiva $h_c(\tau)$ de um canal multipercurso pode ser representado pela Equação 2.2, onde N é o número de percursos, α_n , τ_n e θ_n são a atenuação, o atraso e a alteração na fase do sinal provocado pelo n -ésimo percurso, respectivamente. Em um modelo mais completo e fidedigno, os parâmetros α_n , τ_n e θ_n são variantes no tempo devido à mudanças na posição dos obstáculos (exemplo: movimento de veículos e pessoas), do transmissor e/ou do receptor.

$$h_c(\tau) = \sum_{n=1}^N \alpha_n e^{-j\theta_n} \delta(\tau - \tau_n) \quad (2.2)$$

O ETSI (2009, anexo B, p. 43) apresenta um modelo de canal com $N = 20$ percursos para a recepção portátil, utilizado em simulações de sistemas de televisão digital. Os gráficos deste modelo são mostrados na Figura 17. Já ITU-R (2008, anexo 4) apresenta 10 modelos de canais, reproduzidos na Tabela 4, cujos modelos *Brasil A/B/C/D/E* são frequentemente encontrados na literatura técnica brasileira sobre TV digital.

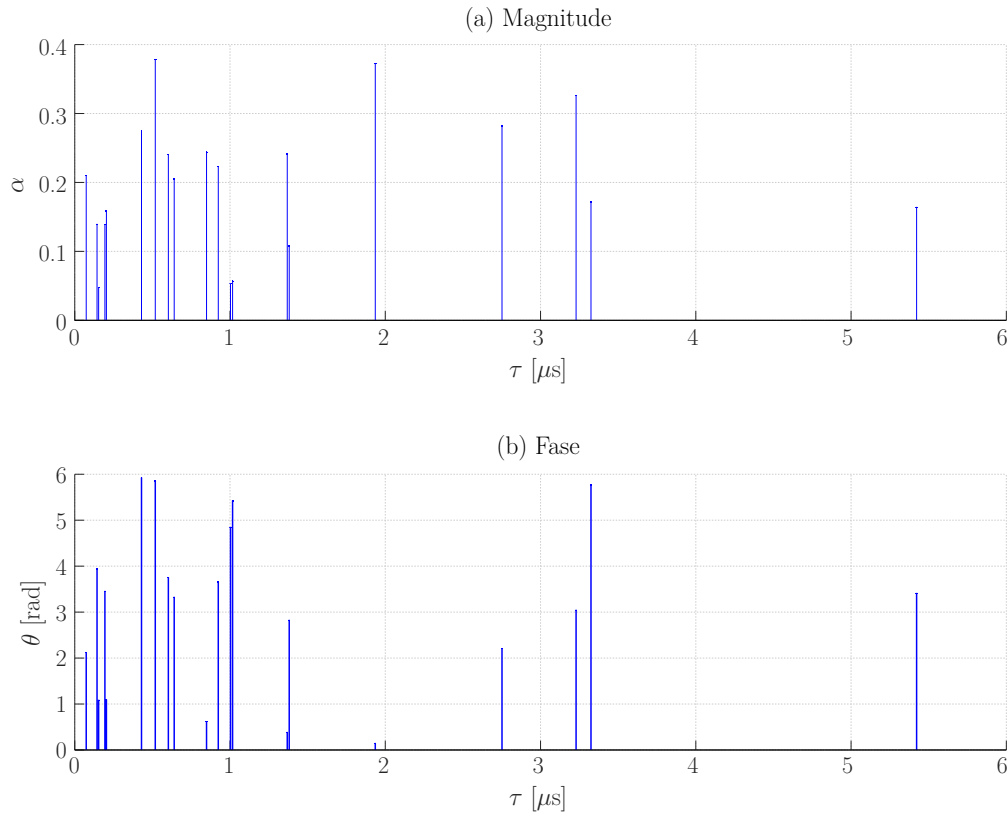
Tabela 4: Modelos de canais multipercurso

Canal	Parâmetro	Caminho					
		1	2	3	4	5	6
UK atraso curto	Atraso (μs)	0	0,05	0,4	1,45	2,3	2,8
	Atenuação (dB)	2,8	0	3,8	0,1	2,6	1,3
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
UK atraso longo	Atraso (μs)	0	5	14	35	54	75
	Atenuação (dB)	0	9	22	25	27	28
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
DVB-T (recepção portátil)	Atraso (μs)	0,5	1,95	3,25	2,75	0,45	0,85
	Atenuação (dB)	0	0,1	0,6	1,3	1,4	1,9
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	336	9	175	127	340	36
CRC	Atraso (μs)	0	-1,8	0,15	1,8	5,7	35
	Atenuação (dB)	0	11	11	1	variável	9
	Doppler (Hz)	0	0	0	0	5	0
	Fase (graus)	0	125	80	45	0	90
Brasil A	Atraso (μs)	0	0,15	2,22	3,05	5,86	5,93
	Atenuação (dB)	0	13,8	16,2	14,9	13,6	16,4
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
Brasil B	Atraso (μs)	0	0,3	3,5	4,4	9,5	12,7
	Atenuação (dB)	0	12	4	7	15	22
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
Brasil C	Atraso (μs)	0	0,089	0,419	1,506	2,322	2,799
	Atenuação (dB)	2,8	0	3,8	0,1	2,5	1,3
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
Brasil D	Atraso (μs)	0,15	0,63	2,22	3,05	5,86	5,93
	Atenuação (dB)	0,1	3,8	2,6	1,3	0	2,8
	Doppler (Hz)	0	0	0	0	0	0
	Fase (graus)	0	0	0	0	0	0
Brasil E	Atraso (μs)	0	1	2	-	-	-
	Atenuação (dB)	0	0	0	-	-	-
	Doppler (Hz)	0	0	0	-	-	-
	Fase (graus)	0	0	0	-	-	-
GSM Típico Urbano	Atraso (μs)	0	0,2	0,5	1,7	2,3	5
	Atenuação (dB)	13	10	12	16	18	20
	Desvanecimento	Rayleigh					

Fonte: ITU-R (2008, anexo 4)

Um dos efeitos mais importantes da propagação multipercurso é a interferência intersimbólica (*Intersymbol Interference* - *ISI*), que consiste no espalhamento temporal do símbolo anterior sobre o símbolo atual, conforme ilustrado na Figura 18. Uma maneira simplista de se evitar a ISI é através da inserção de uma pausa (tempo de guarda) entre os símbolos, em troca de uma redução na capacidade de transmissão.

Figura 17: Exemplo de modelo de canal para recepção portátil



Fonte: ETSI (2009, anexo B, p. 43)

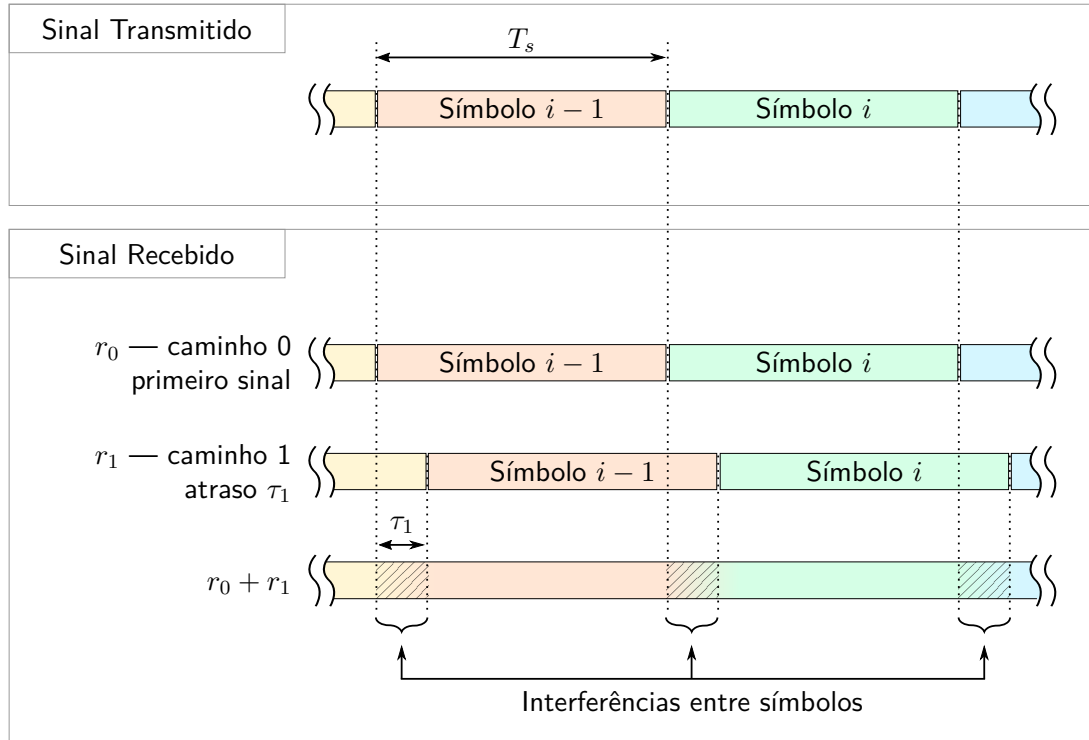
Uma métrica importante em canais multipercurso é o *atraso excedente máximo* τ_{max} , que é calculado pela subtração do instante da recepção do último impulso pelo instante do primeiro impulso detectado. Conforme Sklar (1997, p. 94), o último impulso recebido a ser considerado pode ser aquele que se enquadra acima do nível de -10 dB ou -20 dB em relação ao impulso mais forte recebido. A relação entre τ_{max} e o período do símbolo T_s pode categorizar o canal em um dos tipos:

1. Desvanecimento plano, se $\tau_{max} \leq T_s$;
2. Desvanecimento seletivo em frequência, se $\tau_{max} > T_s$.

Outra métrica importante para um canal multipercurso é o *espalhamento do atraso RMS* σ_τ , definido pela Equação 2.3 (SKLAR, 1997, p. 95). Como exemplo, considerando a propagação em ambientes urbanos, σ_τ pode chegar a $25\mu s$, já a propagação em ambientes internos, σ_τ assume valores como 10 a $100ns$ (RAPPAPORT, 2002, p. 162).

$$\sigma_\tau = \sqrt{\tau^2 - (\bar{\tau})^2} \quad (2.3)$$

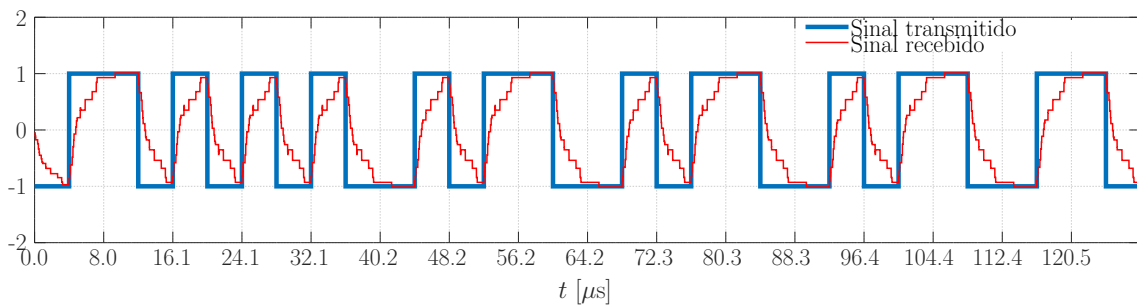
Figura 18: Interferência intersimbólica causada pela propagação multipercurso



Fonte: o autor

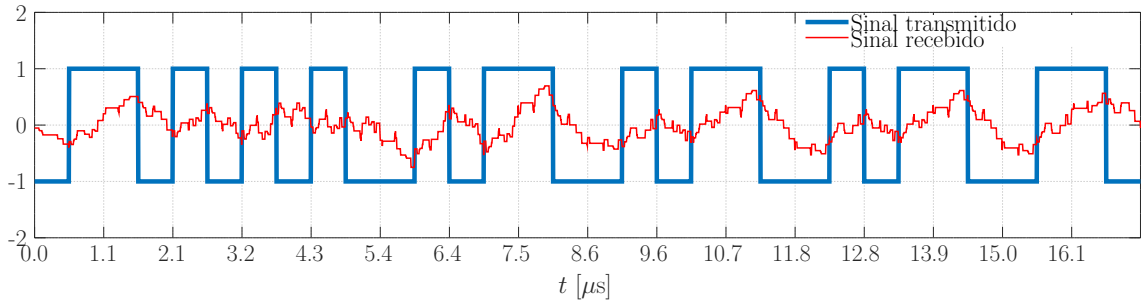
As figuras 19 e 20 apresentam resultados de simulações de transmissões de um sinal banda-base hipotético utilizando o canal ETSI (2009, anexo B, p. 43) (Figura 17). Este modelo de canal possui *atraso excedente máximo* $\tau_{max} = 5,3 \mu s$. Na primeira figura, o período do símbolo é $T_s = 1,5 \times \tau_{max}$ ($\tau_{max} \leq T_s$, caracterizando o desvanecimento plano), e a forma de onda na saída do canal apresenta distorção moderada. Já na segunda figura $T_s = 0,2 \times \tau_{max}$ ($\tau_{max} > T_s$, o que caracteriza o desvanecimento seletivo em frequência), sendo possível observar uma distorção severa na forma de onda do sinal na saída do canal, causada pela ISI.

Figura 19: Efeito do desvanecimento plano



Fonte: o autor

Figura 20: Espalhamento temporal dos símbolos, efeito do desvanecimento seletivo em frequência, provocando ISI



Fonte: o autor

Os efeitos do canal multipercurso também podem ser analisados no domínio da frequência. Neste caso, a resposta em frequência do canal pode ser obtida através da transformada de Fourier da resposta impulsiva do canal. Quando a resposta em frequência do canal não é plana o suficiente para a faixa de frequência do sinal transmitido, ocorre a distorção do sinal. A *largura de faixa coerente* B_c é uma medida estatística da largura de faixa do canal onde a resposta é considerada plana. Existem duas aproximações estatísticas para a relação entre σ_τ e B_c : caso a correlação em frequência seja 0,9, a largura de faixa coerente é dada pela Equação 2.4. Caso a correlação seja relaxada para 0,5, a Equação 2.5 pode ser utilizada (RAPPAPORT, 2002, p. 163).

$$B_{c,90} \approx \frac{1}{50\sigma_\tau} \quad (2.4)$$

$$B_{c,50} \approx \frac{1}{5\sigma_\tau} \quad (2.5)$$

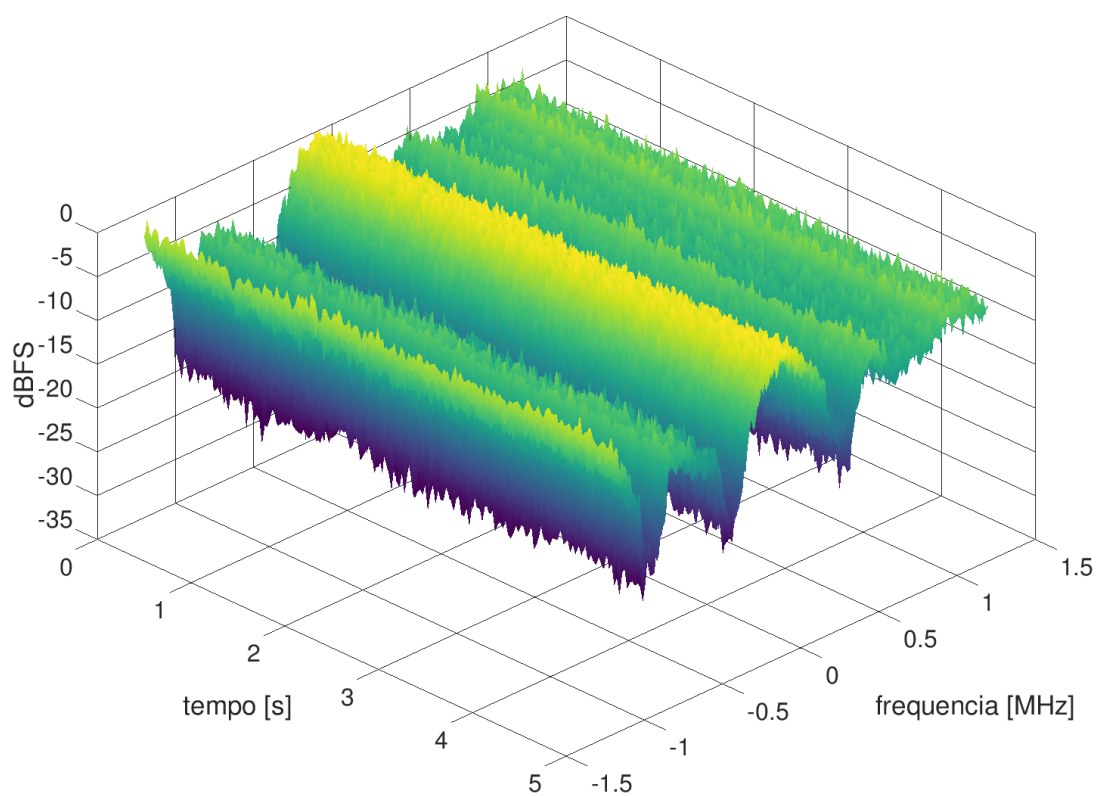
A Figura 21 apresenta a parte central (largura de faixa de 2,4 MHz) do espectro obtido pela recepção de um sinal de TV digital ISDB-T⁵ com antena interna. Neste espectro é possível observar claramente algumas frequências fortemente atenuadas, efeito do desvanecimento em frequência causado pela propagação multipercurso⁶. A amplitude das atenuações chegam a 30 dB quando comparadas à maior amplitude do espectro.

O espectrograma da Figura 22 foi obtido nas mesmas condições do anterior, porém com o receptor em movimento. Nesta figura é possível observar grandes variações na resposta do canal ao longo do tempo, mesmo com o movimento limitado a menos de meio metro.

⁵ Canal 30 de Uberlândia, frequência ≈ 569 MHz

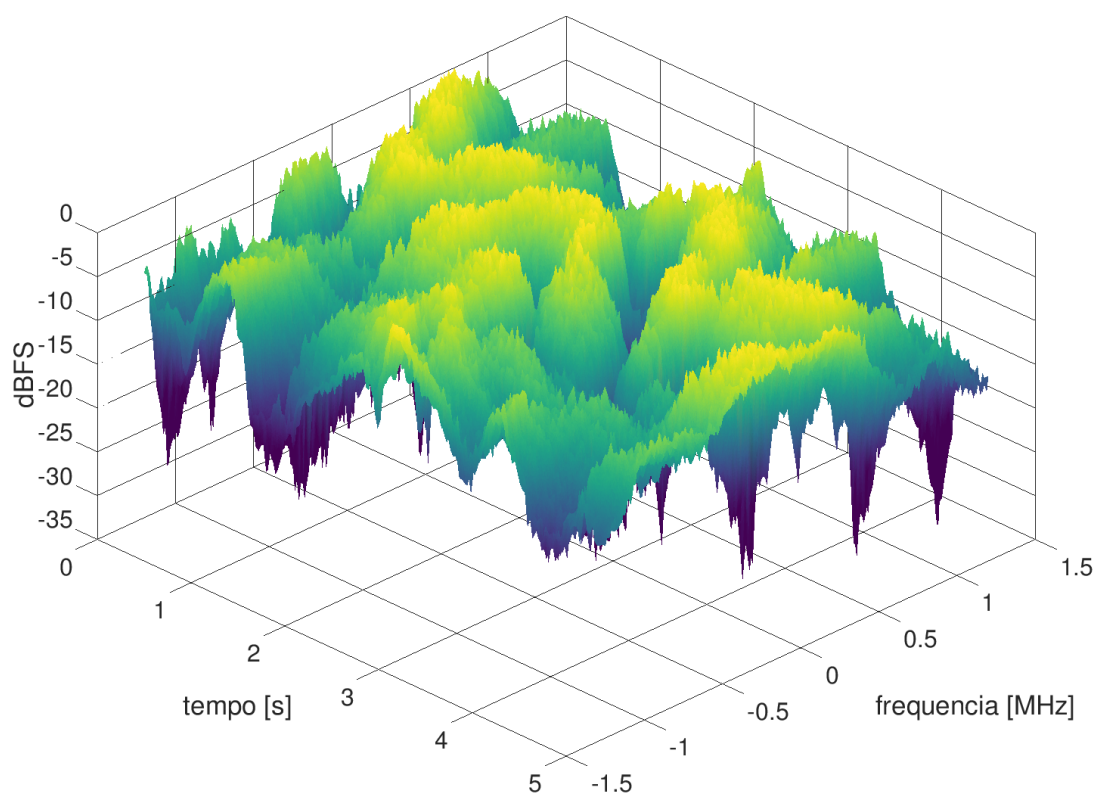
⁶ O espectro esperado para um canal ideal (não multipercurso) seria um espectro basicamente plano, devido às características do sinal ISDB-T

Figura 21: Efeito da propagação multipercurso no domínio da frequência para um receptor estático



Fonte: o autor

Figura 22: Efeito da propagação multipercurso no domínio da frequência para um receptor em movimento



Fonte: o autor

2.2 OFDM

O OFDM (*Orthogonal Frequency Division Multiplexing* ou Multiplexação pela Divisão em Frequências Ortogonais) é um método de se codificar dados digitais em múltiplas portadoras estreitamente espaçadas e ortogonais. Em síntese, O OFDM permite a troca de um único sinal com largura de faixa B e período de símbolo T_s por N sinais paralelos, ortogonais e estreitamente espaçadas em frequência, cada um com largura de faixa $B_{sc} = B/N$ e período de símbolo $T_{\text{OFDM}} = N \cdot T_s$.

A Figura 23 compara a ocupação de 16 símbolos (S_1, S_2, \dots, S_{16}) na frequência e no tempo para: (a) um sistema de única portadora; (b) um sistema OFDM hipotético com $N = 8$. É possível observar neste exemplo que o OFDM, quando comparado ao sistema de portadora única, possui as seguintes propriedades:

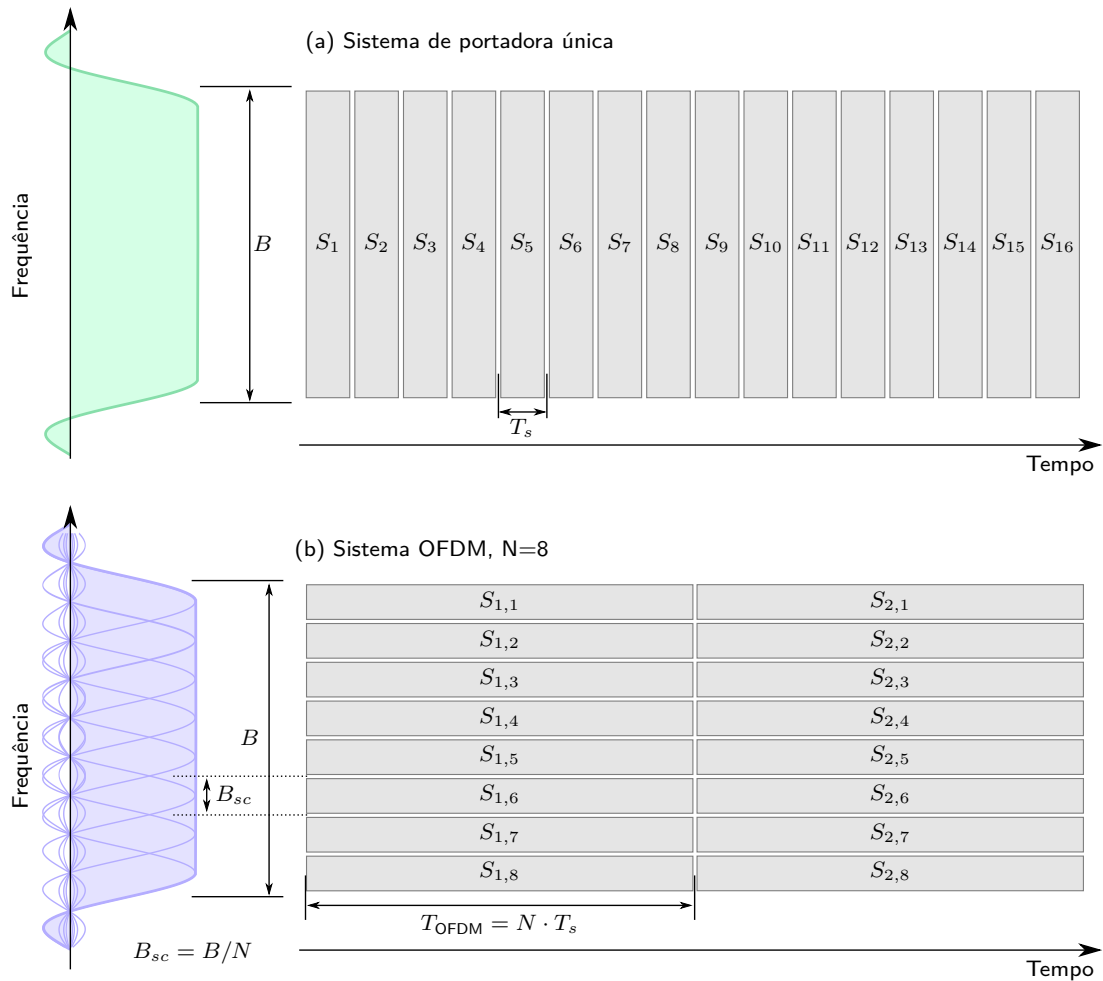
- Cada símbolo possui duração 8 vezes maior;
- São transmitidos 8 símbolos simultaneamente ($S_{x,1}, S_{x,2}, \dots, S_{x,8}$) em frequências adjacentes (chamadas de *subportadoras*);
- A largura da faixa de frequência de cada suportadora é 8 vezes menor;
- As subportadoras do OFDM estão *minimamente* espaçadas mas não se interferem, pois são ortogonais entre si.

Como cada portadora OFDM codifica seus símbolos em intervalos de tempo N vezes mais longos (quando comparado a um sistema convencional de portadora única), a dispersão temporal do sinal provocada pelo canal multipercurso torna-se menos prejudicial. Assim, o OFDM permite transformar um canal largo — que como um todo estaria sujeito ao desvanecimento seletivo em frequência — em N canais estreitos paralelos, cada um sujeito ao desvanecimento plano, muito mais simples de se mitigar.

Em um sistema de portadora única com alta taxa de transmissão, a largura de faixa do sinal transmitido é geralmente maior que a largura de faixa coerente (B_c), tornando necessária a incorporação de um equalizador no receptor para corrigir as distorções no sinal provocadas pelo canal. A complexidade deste equalizador é proporcional à máxima dispersão temporal do canal utilizada no projeto do sistema. Esta complexidade pode tornar o sistema inviável, por isso muitos sistemas de telecomunicação modernos recorrem a outras técnicas para combater os efeitos da propagação multipercurso, como o OFDM.

O OFDM encontra seu campo principal de uso nos sistemas de telecomunicação sujeitos a propagação multipercurso, onde a taxa de transmissão é elevada e a resposta impulsiva do canal possui um comprimento essencialmente maior que o tempo de símbolo original. É notável que em alguns tipos de sistemas, como os via satélite que utilizam

Figura 23: Comparação entre (a) sistema de portadora única e (b) OFDM



Fonte: o autor

antenas direcionais (parabólicas, por exemplo) e sistemas a cabo coaxial (televisão e modem a cabo) o OFDM não é muito empregado. O motivo é que tais sistemas são praticamente livres da propagação multipercurso (LATHI; DING, 2009, p. 810).

2.2.1 Histórico

O OFDM foi primeiro descrito por Chang em 1966. Em 1971 Weinstein e Ebert sugeriram o uso da transformada discreta de Fourier (DFT) em substituição aos bancos de geradores senoidais, o que reduziria de forma significativa a complexidade de implementação dos modems OFDM (HANZO; KELLER, 2006, p. 5).

Com o avanço da eletrônica e da computação, microprocessadores e DSPs potentes se tornaram disponíveis a custos acessíveis para uma vasta gama de aplicações comerciais. O OFDM se tornou realizável na esteira destes avanços, encontrando uso em diversas tecnologias de uso cotidiano pelo mundo, como o DAB (1995), ADSL (1996), DVB-T

(1997), IEEE 802.11a/g/n (WiFi - 1999, 2002, 2004), IEEE 802.16d (WiMax, 2004), LTE (2005), dentre outros (LATHI; DING, 2009, p. 811). O ISDB-T (2001), objeto de detalhamento desta dissertação, também faz uso do OFDM (ARIB, 2014, p. 31).

2.2.2 Ortogonalidade

Considere duas subportadoras cujas frequências f_a e f_b estejam espaçadas por $\Delta f = f_a - f_b$, e o período dos símbolos que as modulam seja T . A condição para que as duas subportadoras não interfiram entre si é a condição da ortogonalidade. Por definição, duas funções f e g são ortogonais se obedecerem a relação⁷:

$$\langle f, g \rangle = \int_{x=-\infty}^{+\infty} f(x)g(x)^* dx = 0$$

Aplicando-se o princípio da ortogonalidade para as duas subportadoras $e^{j2\pi f_1 t}$ e $e^{j2\pi f_2 t}$, tem-se:

$$\begin{aligned} \langle e^{j2\pi f_1 t}, e^{j2\pi f_2 t} \rangle &\equiv \int_{t=0}^T e^{j2\pi f_1 t} \cdot e^{-j2\pi f_2 t} dt = 0 \\ \frac{-j}{2\pi(f_1 - f_2)} \left(e^{jT2\pi(f_1 - f_2)} - 1 \right) &= 0 \\ e^{jT2\pi(f_1 - f_2)} &= 1 \\ T2\pi(f_1 - f_2) &= 2\pi k, \quad k \in \mathbb{Z}^+ \end{aligned}$$

E por fim:

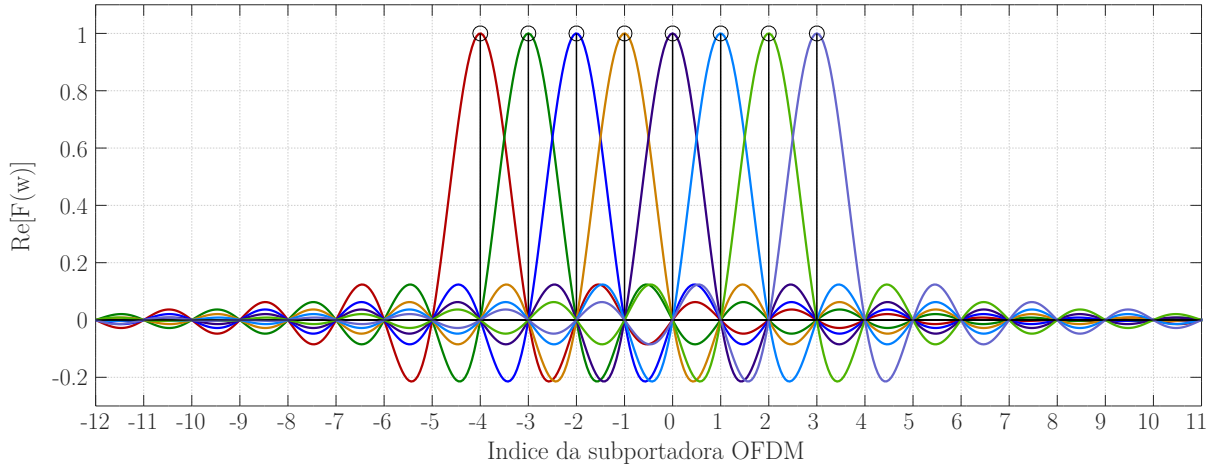
$$\Delta f = f_1 - f_2 = \frac{k}{T}, \quad k \in \mathbb{Z}^+ \quad (2.6)$$

A conclusão importante da Equação 2.6 é que o espaçamento entre portadoras $\Delta f = k/T$ (onde k é um inteiro positivo) garante a ortogonalidade. O menor espaçamento possível é $\Delta f = 1/T$ ($k=1$), o utilizado pelo OFDM.

Como todas as subportadoras do OFDM possuem comprimento limitado T_s , o espectro de um símbolo OFDM é a somatória de funções *sinc*, deslocadas na frequência em $1/T_s$ em relação à subportadora vizinha. A Figura 24 apresenta o espectro de um sinal OFDM hipotético de 8 subportadoras. É possível observar que o pico de qualquer portadora OFDM coincide exatamente com os nulos de todas as outras subportadoras.

⁷ O asterisco denota o complexo conjugado

Figura 24: Ortogonalidade das subportadoras OFDM



Fonte: o autor

2.2.3 Implementação

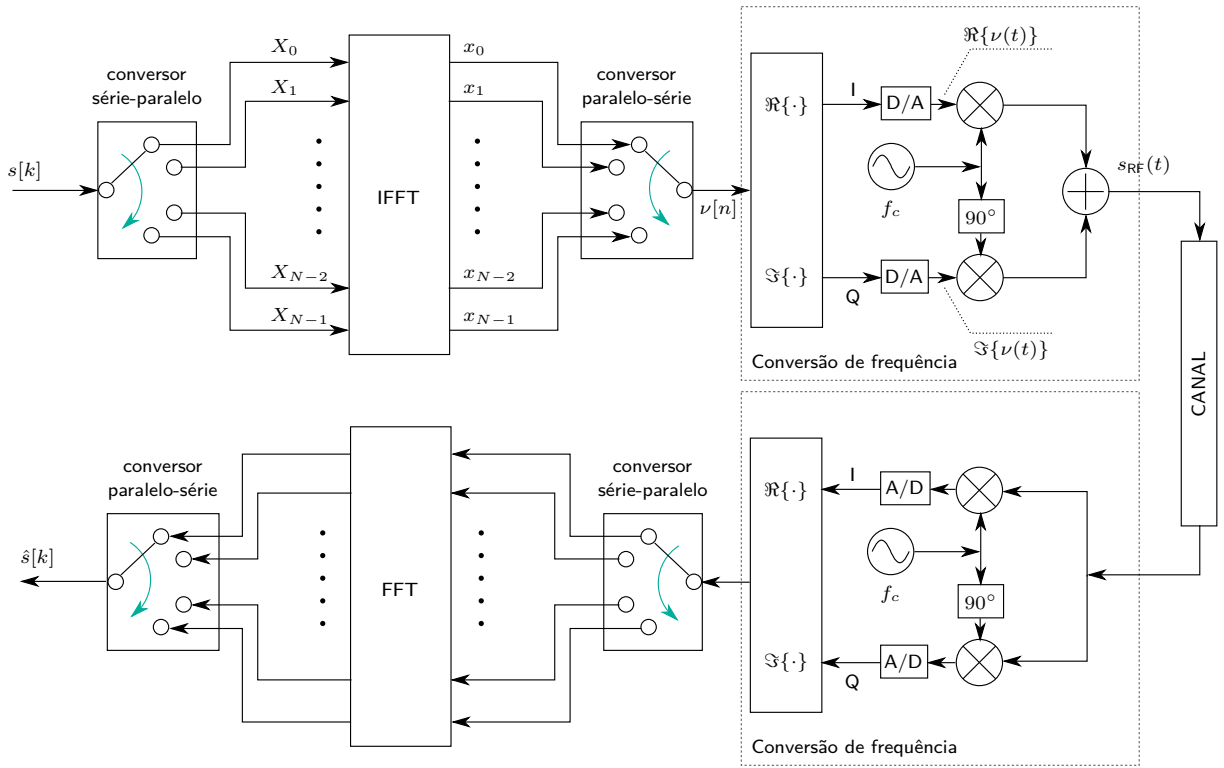
A Figura 25 apresenta um transmissor OFDM e um receptor simplificado. Na transmissão, ocorrem os seguintes processos para geração do sinal de RF ($s_{\text{RF}}(t)$) a partir dos símbolos complexos de entrada $s[k]$:

1. Dado o feixe de símbolos complexos $s[k]$ que se deseja transmitir com taxa R [símbolos/segundo] e período de símbolo $T_s = 1/R$, a primeira etapa é a divisão deste em N subfeixes paralelos $X_0, X_1 \dots X_{N-2}, X_{N-1}$, cada um com taxa $R_{\text{OFDM}} = R/N$. Esta divisão é feita através de um conversor série-paralelo, de modo que $X_m = \{s[m], s[N+m], s[2N+m], s[3N+m], \dots\}$;
2. O vetor $X = [X_0, X_1 \dots X_{N-2}, X_{N-1}]$ forma a entrada de um bloco IFFT (*Inverse Fast Fourier Transform* ou Transformada Rápida de Fourier Inversa) de tamanho N . A cada final de ciclo do conversor série-paralelo, o bloco IFFT é executado, gerando o vetor complexo $x = [x_0, x_1 \dots x_{N-2}, x_{N-1}]$, chamado de *símbolo OFDM*. O símbolo OFDM possui taxa R_{OFDM} e duração $T_{\text{OFDM}} = 1/R_{\text{OFDM}}$;
3. A saída complexa x da IFFT é serializada através de um conversor paralelo-série, gerando o sinal complexo em banda-base $\nu[n] = \{x_0[0], x_1[0] \dots x_{N-1}[0], x_0[1], x_1[1] \dots x_{N-1}[1] \dots\}$, de taxa R ;
4. O sinal $\nu[n]$ tem suas partes real e imaginária separadas, dando origem aos sinais I (*In-phase* ou em fase) e Q (*Quadrature* ou em quadratura). Os sinais IQ $\Re\{\nu[n]\}$ e $\Im\{\nu[n]\}$ são então convertidos para sinais analógicos $\Re\{\nu(t)\}$ e $\Im\{\nu(t)\}$ através de conversores D/A para serem entregues a um modulador de RF;

5. O sinal *em fase* modula uma portadora de frequência f_c (através de sua multiplicação pela portadora), já o sinal *em quadratura* modula a mesma portadora defasada de 90° . Os dois sinais modulados são somados para dar origem ao sinal passa-faixa $s_{\text{RF}}(t)$, que pode enfim ser amplificado para a transmissão através de uma antena. Esta é descrita matematicamente através da Equação 2.7.

$$s_{\text{RF}}(t) = \Re \left\{ e^{-j2\pi f_c t} \cdot \nu(t) \right\} \quad (2.7)$$

Figura 25: Transmissor e receptor OFDM



Fonte: o autor

A forma de onda do sinal *banda base equivalente* de um símbolo OFDM $\nu(t)$ com duração $T_{\text{OFDM}} = 1/R_{\text{OFDM}}$ é definido através da Equação 2.8. Note que $\nu(t)$ é um sinal contínuo obtido após a conversão D/A do sinal discreto $\nu[n]$ no transmissor. Analisando-se de outra forma, ao amostrar $\nu(t)$ a cada período T_s , obtemos $\nu[n] = \nu(t = nT_s)$ conforme a Equação 2.9.

$$\nu(t) = \sum_{k=0}^{N-1} s[k] \cdot e^{j2\pi k \frac{t}{T_{\text{OFDM}}}}, \quad \text{para } 0 \leq t < T_{\text{OFDM}} \quad (2.8)$$

$$\nu[n] = \sum_{k=0}^{N-1} s[k] \cdot e^{j2\pi k \frac{n}{N}}, \quad \text{para } 0 \leq n \leq N-1 \quad (2.9)$$

É relevante destacar que a Equação 2.9 é a transformada discreta de Fourier inversa⁸, que na prática é implementada de maneira eficiente através do algoritmo IFFT (*Inverse Fast Fourier Transform*, ou Transformada Rápida de Fourier Inversa). Outra constatação pertinente é que o alfabeto do símbolo OFDM $\nu(t)$ possui M^N símbolos distintos (M é o número de símbolos diferentes do alfabeto do feixe de entrada $s[k]$ e N é o número de subportadoras ou o tamanho da FFT). Em outras palavras, existirão M^N formas de ondas diferentes possíveis para cada símbolo OFDM emitido.

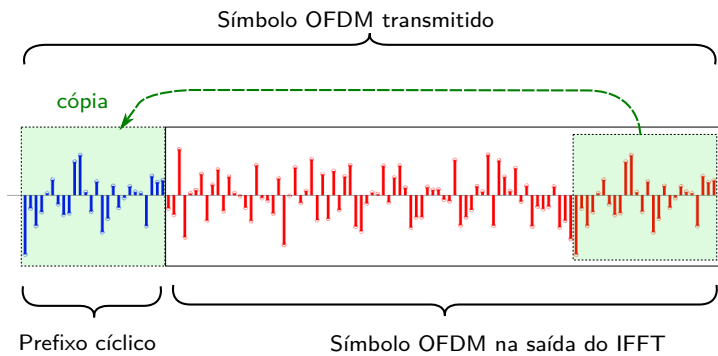
O funcionamento do receptor OFDM é basicamente o inverso do transmissor, conforme⁹ a parte inferior da Figura 25. A principal alteração no receptor é o uso do bloco FFT, que *desfaz* o efeito do bloco IFFT do transmissor.

2.2.4 Prefixo cíclico

O aumento da duração do símbolo em um sistema OFDM por si só, quando comparado a um sistema de portadora única, melhora mas não garante a imunidade contra a ISI causada pelos efeitos do multipercurso.

Uma técnica utilizada nos sistemas OFDM para eliminar a ISI, é a adição de um intervalo de guarda contendo um prefixo cíclico para cada símbolo OFDM gerado pela IFFT. O prefixo cíclico é obtido simplesmente através da cópia de uma parte do final do símbolo OFDM original, conforme ilustrado pela Figura 26.

Figura 26: Inclusão do prefixo cíclico na transmissão



Fonte: o autor

O comprimento do prefixo cíclico, T_g , é configurado para absorver a maior parte da energia da dispersão do símbolo OFDM imediatamente anterior, por exemplo através do critério de projeto $T_g > \tau_{max}$.

Enquanto o espalhamento provocado pelo canal possuir duração menor que a do prefixo cíclico, não haverá interferência intersimbólica. No padrão ISDB-T, a duração do

⁸ Exceto pelo fator de escala $1/N$ (o mais comumente utilizado).

⁹ Na figura foram omitidas as funções de sincronismo necessárias ao receptor, por simplificação

prefixo cíclico é especificada como uma fração da duração útil do símbolo, e pode variar de $7,875 \mu s$ até $252 \mu s$ dependendo dos parâmetros de transmissão.

No receptor, o prefixo cíclico é removido antes da FFT e o processamento segue normalmente. A cópia das amostras do final do símbolo para o início, permite que a convolução linear que ocorre no canal multipercurso seja modelada com uma *convolução circular*. Isto é conveniente, pois, como a transformada discreta de Fourier é utilizada no receptor OFDM, a equalização é simplificada.

3 Códigos corretores de erro

3.1 Códigos Reed-Solomon

3.1.1 Introdução

Os códigos Reed-Solomon (RS) são códigos de bloco cíclicos e pertencem a uma subclasse dos códigos Bose-Chaudhuri-Hocquenghem (BCH) não binários. A família dos códigos RS é extensa: existem códigos RS para diversos comprimentos de bloco e a quantidade de símbolos de paridade (e o número de erros possíveis de se corrigir) pode ser definida livremente. A flexibilidade de parâmetros permite seu uso em diversas aplicações com requisitos específicos. Além disso, por utilizar símbolos compostos por múltiplos bits, possui grande poder de correção de erros, especialmente os do tipo rajada.

É utilizado, por exemplo, em diversos padrões de televisão digital (ISDB-T, DVB-T, DVB-S, ATSC), discos compactos (CD, DVD, Blu-ray), ADSL, WiMAX, nas comunicações pelo espaço profundo (sondas espaciais Voyager) e nos *QR codes*. Em muitas aplicações, o RS é concatenado com um código convolucional para obter um poder de correção de erros ainda maior. Atualmente há uma tendência gradual em novos sistemas do uso de códigos mais modernos, como o Turbo e o LDPC.

A invenção dos códigos Reed-Solomon é atribuída a Irving Stoy Reed e Gustave Solomon. O trabalho foi publicado inicialmente em 1960 em um artigo de 5 páginas intitulado *Polynomial Codes Over Certain Finite Fields* da revista científica *Journal of the Society for Industrial and Applied Mathematics* (REED; SOLOMON, 1960). Os códigos RS são contemporâneos dos códigos BCH e são considerados uma subclasse destes, apesar de serem desenvolvidos independentemente.

A abordagem original dos códigos RS pelos autores exposta naquele artigo de 1960 não é a mesma utilizada atualmente, mas os fundamentos daquele código revolucionário permanece. Naquela publicação original, o método de decodificação era impraticável, devido a complexidade computacional. Somente oito anos depois, Berlekamp (1968) publicou um algoritmo de decodificação eficiente, que foi posteriormente simplificado por Massey (1969). Wicker e Bhargava (1999, cap. 2) apresentam com uma surpreendente riqueza de detalhes o contexto histórico da invenção dos códigos RS.

A nomenclatura $RS(n, k)$ é normalmente utilizada para se referir aos códigos Reed-Solomon com comprimento de bloco n e comprimento da mensagem k . O RS é definido em um campo de Galois de ordem $q = p^m$, representado pela notação $GF(p^m)$ ou $GF(q)$. Os principais conceitos matemáticos necessários ao entendimento dos códigos Reed-Solomon

são apresentados pelo Apêndice A.

Um código RS(n, k) possui distância mínima $d_{min} = n - k + 1$, portanto pode corrigir até $t = \lfloor (n - k + 1)/2 \rfloor$ erros e até $2t$ rasuras (chamados também de *erasures*, símbolos marcados previamente como incorretos). Assim o RS atinge o *limite de Singleton*, a igualdade da inequação $d_{min} \leq n - k + 1$, sendo portanto um *código de distância máxima* (*Maximum Distance Separable - MDS*), provendo notável capacidade de correção de erros.

Os principais parâmetros de um código RS com mensagem e código $\in GF(2^m)$ são:

- Comprimento do bloco: $n = 2^m - 1$ símbolos
- Comprimento da mensagem: k símbolos
- Comprimento da paridade: $n - k$ símbolos
- Capacidade de correção de erros: até $t = \lfloor (n - k + 1)/2 \rfloor$ símbolos errados
- Distância mínima: $d_{min} = n - k + 1$ símbolos

3.1.2 Abordagem original dos códigos RS

Esta seção esboça a perspectiva original dos códigos RS apresentada por Reed e Solomon (1960). A abordagem original, em síntese, trata os problemas da codificação e decodificação como uma interpolação polinomial.

O código $RS(n, k)$ faz o mapeamento de uma mensagem \mathbf{m} , de comprimento k em uma palavra código \mathbf{c} de comprimento n , ambos os vetores com símbolos pertencentes ao campo $F = GF(2^l)$. Formalmente $\mathcal{C} : F^k \rightarrow F^n$, $k < n \leq 2^l$.

Considere a mensagem do vetor \mathbf{m} , composto por k símbolos $\in F$:

$$\mathbf{m} = [m_0, m_1, m_2, \dots, m_{k-1}]$$

Na abordagem original dos códigos RS, os símbolos da mensagem \mathbf{m} a ser codificada são considerados como sendo os coeficientes de um polinômio $p(x)$ de grau até $k - 1$:

$$p(x) = m_0 + m_1x + m_2x^2 + \dots + m_{k-1}x^{k-1}$$

A palavra código $\mathbf{c} \in F$ é o vetor resultante do cálculo de $p(x)$ em n posições $(0, \alpha, \alpha^2, \dots, \alpha^{n-2}, 1)$ conhecidas tanto pelo transmissor como pelo receptor, onde α é um elemento primitivo de F :

$$\mathbf{c} = [p(0), p(\alpha), p(\alpha^2), \dots, p(\alpha^{n-2}), p(1)]$$

Obviamente, esta construção dá origem a um código não-sistemático, pois a mensagem original não faz parte da palavra código.

Considere a palavra-código recebida $\mathbf{r} = \mathbf{c} + \mathbf{e}$, onde \mathbf{e} é um vetor de erros provocados pelo canal:

$$\mathbf{r} = [r(0), r(\alpha), r(\alpha^2), \dots, r(\alpha^{n-2}), r(1)]$$

A partir de \mathbf{r} , o decodificador tenta recuperar a mensagem $\hat{\mathbf{m}} = \mathbf{m}$, através da resolução do sistema de n equações:

$$\begin{aligned} r(0) &= \hat{m}_0 \\ r(\alpha) &= \hat{m}_0 + \hat{m}_1\alpha + \hat{m}_2\alpha^2 + \dots + \hat{m}_{k-1}\alpha^{k-1} \\ r(\alpha^2) &= \hat{m}_0 + \hat{m}_1\alpha^2 + \hat{m}_2\alpha^4 + \dots + \hat{m}_{k-1}\alpha^{2(k-1)} \\ &\vdots \\ r(1) &= \hat{m}_0 + \hat{m}_1 + \hat{m}_2 + \dots + \hat{m}_{k-1} \end{aligned}$$

Note que este sistema de equações dá origem a uma matriz de Vandermonde, e para o caso de não haver erros de transmissão ($\mathbf{r} = \mathbf{c}$), é garantido que o sistema será determinado para qualquer conjunto k equações escolhidas dentre as n disponíveis. Há portanto $\binom{n}{k}$ sistemas diferentes possíveis de se montar.

O algoritmo de decodificação descrito no artigo original propõe que os $\binom{n}{k}$ sistemas sejam resolvidos, e cada sistema, quando determinado, tem como solução uma *mensagem candidata* $\hat{\mathbf{m}}_i$. Junto com cada $\hat{\mathbf{m}}_i$ deve ser armazenada também a quantidade de *votos* acumulados até então para aquela mensagem, ou seja, a quantidade de sistemas resolvidos que chegaram àquela mesma solução.

Após a conclusão do cálculo de todos os $\binom{n}{k}$ sistemas, a mensagem recuperada $\hat{\mathbf{m}}$ é definida como sendo aquela com a maior quantidade de votos. É provado que um código RS pode corrigir até t erros, desde que $t \leq \lfloor (n - k + 1)/2 \rfloor$ (REED; SOLOMON, 1960, p. 302).

Este procedimento de decodificação é muito ineficiente pois requer a avaliação de $\binom{n}{k}$ sistemas de equação. Por exemplo, no código RS utilizado no ISDB-T, tem-se $n = 255$ e $k = 239$, e $\binom{255}{239} = 9,45 \cdot 10^{24}$, uma quantidade muito grande de sistemas de equações para se calcular. Portanto, o método original de decodificação é considerado apenas teórico, pois poderia ser aplicado somente em códigos muito curtos, os quais não possuem interesse prático.

A despeito da complexidade de decodificação, é possível modificar o procedimento original para dar origem a um código sistemático. Basicamente, ao invés de tratar \mathbf{m} como *coeficientes* de um $p(x)$, considera-se um outro polinômio $p'(x)$ onde os símbolos m_i ($0 \leq i \leq k - 1$) da mensagem são os *valores* deste novo polinômio quando calculado nos pontos x_i ($0 \leq i \leq k - 1$): $p'(x_i) = m_i$. Através de regressão polinomial, obtém-se

os coeficientes de $p'(x)$. Calculando-se $p'(x)$ nos primeiros k valores de x_i , obtém-se a mensagem, e continuando a avaliação para outros $n - k$ valores diferentes de x_i obtém-se os símbolos de paridade. Assim, $\mathbf{c} = [p'(x_0), p(x_1), \dots, p(x_{k-1}), p(x_k), p(x_{k+1}), \dots, p(x_{n-1})]$ forma um código sistemático.

3.1.3 Abordagem clássica dos códigos RS

Na abordagem clássica, a construção de um código RS se baseia nos códigos cíclicos que utilizam um polinômio auxiliar $g(x)$, chamado de *polinômio gerador*, para gerar a palavra código a partir da mensagem.

A teoria dos códigos cíclicos evoluiu inicialmente de forma independente dos códigos RS, mas em 1961 os pesquisadores Gorenstein and Zierler generalizaram os códigos binários BCH para campos abitrários do tipo $GF(p^m)$. Naquele momento percebeu-se que havia sido criada nova maneira de descrever os códigos RS.

Esta então nova perspectiva dos códigos RS é a mais comum atualmente na literatura dos códigos corretores de erro. Welch (1997) e Wicker e Bhargava (1999, p. 6) apresentam mais detalhes sobre as duas diferentes perspectivas dos códigos Reed-Solomon.

Um código cíclico é um código de bloco linear com uma estrutura matemática inerente bem definida que torna a sua codificação mais fácil e a decodificação eficiente. Um código cíclico obedece duas propriedades básicas:

- Propriedade da *linearidade*: a soma de duas quaisquer palavras código resulta em outra palavra código;
- Propriedade *cíclica*: qualquer deslocamento cíclico de uma palavra código resulta em outra palavra código.

A propriedade da linearidade advém dos códigos de bloco lineares, pois um código cíclico também é um código de bloco linear. Já a propriedade cíclica nos diz que, se o vetor $\mathbf{c}_0 = (c_0, c_1, c_2, \dots, c_{n-2}, c_{n-1})$ de comprimento n for uma palavra código, quaisquer deslocamentos cíclicos de seus elementos formam outras palavras códigos, conforme exemplos abaixo:

$$\mathbf{c}_{-1} = (c_1, c_2, c_3, \dots, c_{n-1}, c_0) \quad (3.1)$$

$$\mathbf{c}_1 = (c_{n-1}, c_0, c_1, \dots, c_{n-3}, c_{n-2}) \quad (3.2)$$

$$\mathbf{c}_2 = (c_{n-2}, c_{n-1}, c_0, \dots, c_{n-4}, c_{n-3}) \quad (3.3)$$

Interpretando os elementos (símbolos) da palavra código de comprimento n como os coeficientes de um polinômio $c(x)$ de grau $n - 1$, temos o seguinte polinômio código:

$$c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1}$$

O deslocamento cíclico pode então ser modelado como a multiplicação de $c(x)$ por x^d seguida da redução módulo $x^n - 1$, onde d é o deslocamento desejado, por exemplo:

$$x \cdot c(x) \mod x^n - 1 = c_{n-1} + c_0x + c_1x^2 + \cdots + c_{n-2}x^{n-1}$$

Portanto, se $c(x)$ for uma palavra código (representada por um polinômio), $x^d \cdot c(x) \mod x^n - 1$ também será uma palavra código para qualquer deslocamento cíclico d .

Uma palavra código de comprimento n , com símbolos $\in F_q = GF(q)$ forma um espaço vetorial F_q^n . Considerando a representação polinomial, um código cíclico \mathcal{C} é um subconjunto do anel $R_q^n = F_q[x]/(x^n - 1)$, que é a classe de resíduos (as operações são calculadas módulo $x^n - 1$) do anel de polinômios $F_q[x]$. Mais especificamente, o referido subconjunto é um ideal principal I do anel R_q^n . Este ideal principal é gerado por um polinômio $g(x)$ chamado de *polinômio gerador do código*, portanto $\mathcal{C} = I = \langle g(x) \rangle$.

O polinômio gerador $g(x)$ de um código \mathcal{C} é uma palavra código representada por um polinômio mônico, de grau mínimo $n - k$ e fator de $x^n - 1$ em $F_q[x]$. Qualquer palavra código $c(x) \in \mathcal{C}$ pode ser gerada por um múltiplo do polinômio gerador:

$$c(x) = m(x) \cdot g(x)$$

Definindo o comprimento da palavra código para n símbolos, temos que o polinômio $c(x)$ deve possuir no máximo o grau $n - 1$. Como $g(x)$ por definição possui grau $n - k$, o múltiplo $m(x)$ deve possuir no máximo o grau $k - 1$. Os coeficientes de $m(x)$ podem então ser representados por um vetor de k símbolos. O polinômio $m(x)$, então, é a *mensagem* de comprimento k a ser codificada, e $c(x)$ é a palavra código correspondente de comprimento n .

Da forma com que foi apresentada, a codificação pela multiplicação direta da mensagem pelo polinômio gerador não dá origem a um código sistemático, pois o vetor da mensagem não está contido no vetor da palavra código.

3.1.3.1 Codificação sistemática

Para se construir um código cíclico sistemático a partir de uma mensagem, deve-se utilizar o seguinte procedimento (HAYKIN, 2014, p. 595):

- Multiplique a mensagem $m(x)$ por x^{n-k}
- Divida $m(x) \cdot x^{n-k}$ pelo polinômio gerador do código $g(x)$, obtendo o resto $b(x)$

- Some $b(x)$ com $m(x) \cdot x^{n-k}$, obtendo-se assim a palavra código $c(x)$ sistemática

Portanto, escrevendo em uma única equação, o código $c(x)$ na forma sistemática é dado por:

$$c(x) = [m(x) \cdot x^{n-k} \bmod g(x)] + m(x) \cdot x^{n-k}$$

Assim como para os códigos RS (considerando a abordagem clássica), o procedimento de codificação descrito acima se aplica a qualquer código cíclico. O polinômio gerador de um código Reed-Solomon segue a forma da Equação 3.4, onde α é um elemento primitivo do $GF(2^m)$ e raiz do polinômio primitivo que define o campo, t é a capacidade de correção de erros do código.

$$g(x) = \prod_{j=b}^{b+2t-1} (x - \alpha^j) = (x - \alpha^b)(x - \alpha^{b+1})(x - \alpha^{b+2}) \cdots (x - \alpha^{b+2t-1}) \quad (3.4)$$

Pode-se observar que o polinômio gerador possui $2t$ raízes consecutivas, as potências de α . A constante b permite a construção de códigos RS diferentes, através do deslocamento das sequências de potências de α . Quando $b = 1$, é dito que se forma um código RS de sentido estrito (*narrow-sense*). Os parâmetros do polinômio gerador utilizado no codificador devem ser conhecidos no decodificador.

3.1.4 Decodificação

Como visto na subseção 3.1.2, o algoritmo de decodificação proposto no artigo de Reed e Solomon em 1960 era impraticável. Mesmo com a reinterpretação dos códigos RS como cíclicos, ainda levou alguns anos até surgirem algoritmos mais eficientes, permitindo aplicações com comprimento de blocos e t (número de erros corrigíveis) maiores.

O algoritmo de Peterson-Gorenstein-Zierler (PGZ), que leva o nome de três pesquisadores que o deram origem, faz a decodificação algébrica. Sua operação requer a inversão de matrizes, não sendo muito eficiente, mas foi um avanço extraordinário quando comparado ao algoritmo original. O PGZ possui relevância histórica e didática por ser conceitualmente simples.

3.1.4.1 Decodificador Peterson-Gorenstein-Zierler (PGZ)

Esta seção aborda o decodificador Peterson-Gorenstein-Zierler (PGZ), cujo nome é uma homenagem aos três pesquisadores que o construíram. O problema da decodificação de um código RS pode ser dividido em duas partes: 1. encontrar as posições dos erros; 2. descobrir as magnitudes dos erros. Nesta seção, a primeira parte pode ser resolvida

através do algoritmo de Peterson, e a segunda com o algoritmo de Gorenstein–Zierler. Blahut (2003, p. 159), Wicker e Bhargava (1999, p. 79) aprofundam mais o assunto e apresentam demonstrações de alguns dos fatos aqui apresentados.

Considere um código RS(n, k) que, no transmissor, gerou uma palavra código $c(x)$ sistemática a partir de uma mensagem $m(x)$. No receptor, a palavra código recebida $r(x)$ é:

$$r(x) = c(x) + e(x)$$

Os coeficientes do polinômio $e(x)$ representam os possíveis erros aditivos de transmissão ocorridos. Caso não tenha ocorrido nenhum erro de transmissão, todos os coeficientes de $e(x)$ são zero. O polinômio $e(x)$ tem a seguinte forma:

$$e(x) = e_0 + e_1x + e_2x^2 + \cdots + e_{n-1}x^{n-1}$$

A tarefa mais importante do decodificador é descobrir $e(x)$ a partir de somente $r(x)$. Para isto, o primeiro passo do decodificador RS(n, k) é calcular a *síndrome* S , um polinômio de grau $2t - 1$ (com $2t$ coeficientes), que possui a forma geral mostrada na Equação 3.5.

$$S(x) = \sum_{j=1}^{2t} S_j x^{j-1} = S_1 + S_2x + S_3x^2 + S_4x^3 + \cdots + S_{2t}x^{2t-1} \quad (3.5)$$

O cálculo dos coeficientes S_j do polinômio $S(x)$ consiste em avaliar $r(x)$ em cada uma das $2t$ raízes de $g(x)$, dadas por α^{b+i} para $0 \leq i < 2t$. Para simplificar a notação das equações subsequentes, foi considerado um código RS *narrow sense*, quando o $g(x)$ é construído com $b = 1$ (Equação 3.4), assim S_j pode ser calculado pela Equação 3.6. Como, por definição, toda palavra-código é múltipla de $g(x)$ (HAYKIN, 2014, p. 594), se $r(x)$ não contiver erros, as raízes de $g(x)$ também serão raízes de $r(x)$, resultando em uma síndrome zero.

$$S_j = r(\alpha^j), \quad 1 \leq j \leq 2t \quad (3.6)$$

Observe que:

$$S_j = r(\alpha^j) = c(\alpha^j) + e(\alpha^j), \quad 1 \leq j \leq 2t$$

Mas como $\{\alpha^j \mid 1 \leq j \leq 2t\}$ são raízes de $c(x)$, temos que $c(\alpha^j) = 0$, resultando na Equação 3.7:

$$S_j = e(\alpha^j), \quad 1 \leq j \leq 2t \quad (3.7)$$

Portanto, a síndrome depende unicamente do erro $e(x)$, qualquer que seja a palavra código $c(x)$. Esta é uma propriedade fundamental dos códigos de bloco lineares. Toda a informação necessária para se encontrar o erro $e(x)$ encontra-se na síndrome $S(x)$.

Considere que tenham ocorrido ν erros, $0 \leq \nu \leq t$, nas posições indeterminadas i_1, i_2, \dots, i_ν . O polinômio $e(x)$ pode então ser reescrito na forma da Equação 3.8.

$$e(x) = \sum_{l=1}^{\nu} e_{i_l} x^{i_l} = e_{i_1} x^{i_1} + e_{i_2} x^{i_2} + \dots + e_{i_\nu} x^{i_\nu} \quad (3.8)$$

Portanto, o coeficiente e_{i_l} é a magnitude do l -ésimo erro na posição desconhecida i_l . O objetivo final do decodificador é recuperar $\hat{m}(x)$, idêntica à mensagem transmitida $m(x)$, baseando-se apenas na palavra código recebida $r(x)$. Para isto, é necessária a identificação do número de erros ν , das localizações i_1, i_2, \dots, i_ν e das suas magnitudes $e_{i_1}, e_{i_2}, \dots, e_{i_\nu}$. Com estas três incógnitas resolvidas, teremos $e(x)$, podendo-se obter $c(x)$ pela subtração de $r(x)$ com $e(x)$. Enfim, a mensagem recuperada $\hat{m}(x)$ estará nas primeiras k posições de $c(x)$, considerando um código sistemático com os símbolos de paridade inseridos no final do código.

Substituindo a Equação 3.8 na Equação 3.7, temos:

$$S_j = \sum_{l=1}^{\nu} e_{i_l} (\alpha^j)^{i_l} = e_{i_1} (\alpha^j)^{i_1} + e_{i_2} (\alpha^j)^{i_2} + \dots + e_{i_\nu} (\alpha^j)^{i_\nu}, \quad 1 \leq j \leq 2t$$

Simplificando a notação, define-se $X_l = \alpha^{i_l}$ (*números localizadores de erros*) e $Y_l = e_{i_l}$ (magnitude dos erros), onde $1 \leq l \leq \nu$. Assim temos a Equação 3.9.

$$S_j = \sum_{l=1}^{\nu} Y_l X_l^j = Y_1 X_1^j + Y_2 X_2^j + \dots + Y_\nu X_\nu^j, \quad 1 \leq j \leq 2t \quad (3.9)$$

Reescrevendo a Equação 3.9 para todos os S_j , observa-se mais claramente que o problema da decodificação então foi reduzido ao seguinte sistema de equações 3.10.

$$\begin{aligned}
S_1 &= Y_1 X_1 + Y_2 X_2 + \cdots + Y_\nu X_\nu \\
S_2 &= Y_1 X_1^2 + Y_2 X_2^2 + \cdots + Y_\nu X_\nu^2 \\
S_3 &= Y_1 X_1^3 + Y_2 X_2^3 + \cdots + Y_\nu X_\nu^3 \\
&\vdots \\
S_{2t} &= Y_1 X_1^{2t} + Y_2 X_2^{2t} + \cdots + Y_\nu X_\nu^{2t}
\end{aligned} \tag{3.10}$$

O sistema possui $2t$ equações não lineares e 2ν incógnitas, X_l e Y_l para $1 \leq l \leq \nu$. As síndromes S_j para $1 \leq j \leq 2t$, são conhecidas do receptor. Como $0 \leq \nu \leq t$, o sistema é determinado porém sua solução direta é difícil, devido à não linearidade.

A solução do sistema 3.10 é simplificada através da definição de um novo polinômio auxiliar $\Lambda(x)$ da Equação 3.11, e subsequentes manipulações algébricas.

$$\Lambda(x) = 1 + \Lambda_1 x + \Lambda_2 x^2 + \cdots + \Lambda_\nu x^\nu \tag{3.11}$$

$\Lambda(x)$ é conhecido como *polinômio localizador de erros*, pois ele é definido como sendo o polinômio cujas raízes são os inversos dos *números localizadores de erros*, ou seja, suas raízes são X_l^{-1} para $1 \leq l \leq \nu$. $\Lambda(x)$, portanto, é construído como:

$$\Lambda(x) = \prod_{l=1}^{\nu} (1 - x \cdot X_l)$$

Definindo $x = X_l^{-1}$ para a equação Equação 3.11 temos:

$$\begin{aligned}
\Lambda(X_l^{-1}) &= 0 \\
0 &= 1 + \Lambda_1 X_l^{-1} + \Lambda_2 X_l^{-2} + \cdots + \Lambda_\nu X_l^{-\nu}
\end{aligned}$$

Multiplicando ambos os lados por $Y_l X_l^{j+\nu}$, temos:

$$\begin{aligned}
0 &= Y_l X_l^{j+\nu} + \Lambda_1 X_l^{-1} Y_l X_l^{j+\nu} + \Lambda_2 X_l^{-2} Y_l X_l^{j+\nu} + \cdots + \Lambda_\nu X_l^{-\nu} Y_l X_l^{j+\nu} \\
0 &= Y_l X_l^{j+\nu} + \Lambda_1 Y_l X_l^{j+\nu-1} + \Lambda_2 Y_l X_l^{j+\nu-2} + \cdots + \Lambda_\nu Y_l X_l^j \\
Y_l (X_l^{j+\nu} + \Lambda_1 X_l^{j+\nu-1} + \Lambda_2 X_l^{j+\nu-2} + \cdots + \Lambda_\nu X_l^j) &= 0
\end{aligned}$$

Esta equação é válida para cada l no intervalo $1 \leq l \leq \nu$ e cada j no intervalo $1 \leq j \leq 2t$. Somando as equações da forma anterior para $l = \{1, 2, \dots, \nu\}$, temos:

$$\sum_{l=1}^{\nu} Y_l (X_l^{j+\nu} + \Lambda_1 X_l^{j+\nu-1} + \Lambda_2 X_l^{j+\nu-2} + \cdots + \Lambda_{\nu} X_l^j) = 0$$

Distribuindo o somatório, temos:

$$\sum_{l=1}^{\nu} Y_l X_l^{j+\nu} + \Lambda_1 \sum_{l=1}^{\nu} Y_l X_l^{j+\nu-1} + \Lambda_2 \sum_{l=1}^{\nu} Y_l X_l^{j+\nu-2} + \cdots + \Lambda_{\nu} \sum_{l=1}^{\nu} Y_l X_l^j = 0$$

Observa-se a correspondência de cada somatório com a síndrome dada pela Equação 3.9. Assim temos:

$$S_{j+\nu} + \Lambda_1 S_{j+\nu-1} + \Lambda_2 S_{j+\nu-2} + \cdots + \Lambda_{\nu} S_j = 0$$

Movendo-se o primeiro termo para o final, temos a Equação 3.12, cujos $\nu + 1$ termos $\{S_u \mid j \leq u \leq j + \nu\}$ são as síndromes calculadas inicialmente pelo receptor. Como $0 \leq \nu \leq t$ e os subscritos de S_u precisam estar no intervalo $1 \leq u \leq 2t$ para se ter síndromes conhecidas, temos que $1 \leq j \leq 2t - \nu$.

$$\Lambda_1 S_{j+\nu-1} + \Lambda_2 S_{j+\nu-2} + \cdots + \Lambda_{\nu} S_j = -S_{j+\nu} \quad (3.12)$$

Portanto, agora podemos construir um sistema linear de $2t - \nu$ equações e ν incógnitas $\Lambda_1, \Lambda_2, \dots, \Lambda_{\nu}$, os coeficientes do polinômio localizador de erros da Equação 3.11. Como há ν incógnitas, são necessárias apenas ν dessas equações para termos o sistema determinado. Por exemplo, as ν primeiras ($1 \leq j \leq \nu$) equações dão origem ao sistema 3.13 representado na forma matricial.

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{\nu} \\ S_2 & S_3 & S_4 & \cdots & S_{\nu+1} \\ S_3 & S_4 & S_5 & \cdots & S_{\nu+2} \\ \vdots & & & & \vdots \\ S_{\nu} & S_{\nu+1} & S_{\nu+2} & \cdots & S_{2\nu-1} \end{bmatrix} \cdot \begin{bmatrix} \Lambda_{\nu} \\ \Lambda_{\nu-1} \\ \Lambda_{\nu-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ -S_{\nu+3} \\ \vdots \\ -S_{2\nu} \end{bmatrix} \quad (3.13)$$

A matriz 3.14 é obtida através da matriz mais à esquerda de 3.13, com $\mu = \nu$. É provado que para o caso de terem ocorrido $\mu = \nu$ erros, a matriz \mathbf{M}_{μ} é inversível (não singular) e por consequência o sistema correspondente é determinado, com uma solução

única.

$$\mathbf{M}_\mu = \begin{bmatrix} S_1 & S_2 & S_3 & \dots & S_\mu \\ S_2 & S_3 & S_4 & \dots & S_{\mu+1} \\ S_3 & S_4 & S_5 & \dots & S_{\mu+2} \\ \vdots & & & & \vdots \\ S_\mu & S_{\mu+1} & S_{\mu+2} & \dots & S_{2\mu-1} \end{bmatrix} \quad (3.14)$$

Como uma matriz é singular somente quando seu determinante é zero, o número de erros ocorridos na transmissão ν pode ser obtido através do cálculo do determinante para diferentes matrizes \mathbf{M}_μ , com $1 \leq \mu \leq t$, pois:

$$\det \mathbf{M}_\mu \begin{cases} = 0 & \text{se } \mu > \nu \\ \neq 0 & \text{se } \mu = \nu \end{cases}$$

O determinante de \mathbf{M}_μ é calculado começando por $\mu = t$ e seguindo em ordem decrescente: $\mathbf{M}_t, \mathbf{M}_{t-1}, \dots, \mathbf{M}_1$. O cálculo é interrompido quando $\mathbf{M}_\mu \neq 0$, neste momento $\mu = \nu$.

Com a determinação do número de erros ocorridos na transmissão ν , o sistema linear 3.13 pode então ser calculado através da inversão de sua matriz quadrada, obtendo-se $\Lambda_1, \Lambda_2, \dots, \Lambda_\nu$, os coeficientes do polinômio localizador de erros da Equação 3.11.

A partir do polinômio $\Lambda(x)$ montado, suas raízes são calculadas por exemplo através do procedimento de Chien (*Chien search*), que consiste em avaliar $\Lambda(x)$ através da força bruta, fazendo q tentativas de x para todos os elementos do campo $GF(q)$ em busca de $\Lambda(x) = 0$. Conforme a definição do $\Lambda(x)$, suas ν raízes são os inversos dos *números localizadores de erros* X_l^{-1} para $1 \leq l \leq \nu$.

Os X_l^{-1} são então invertidos, obtendo-se $X_l = \alpha^{i_l}$. Calcula-se então os logaritmos de α^{i_l} , obtendo-se finalmente i_l , $1 \leq l \leq \nu$, as posições dos erros na palavra código $c(x)$.

Podemos reescrever o sistema de equações 3.10 em forma de matriz, conforme 3.15. Substituindo os valores de X_l (onde $1 \leq l \leq \nu$) neste sistema matricial, teremos agora um sistema linear, com $2t$ equações e ν incógnitas $\{Y_1, Y_2, \dots, Y_\nu\}$. Escolhendo apenas as primeiras ν equações, é provado que este é um sistema determinado com solução única.

Resolvendo este sistema, teremos o conjunto $Y_l = e_{i_l}$, $1 \leq l \leq \nu$, as magnitudes dos erros.

$$\begin{bmatrix} X_1 & X_2 & X_3 & \dots & X_\nu \\ X_1^2 & X_2^2 & X_3^2 & \dots & X_\nu^2 \\ X_1^3 & X_2^3 & X_3^3 & \dots & X_\nu^3 \\ \vdots & & & & \vdots \\ X_1^{2t} & X_2^{2t} & X_3^{2t} & \dots & X_\nu^{2t} \end{bmatrix} \cdot \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_\nu \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_{2t} \end{bmatrix} \quad (3.15)$$

Com os valores de e_{i_l} e i_l , $1 \leq l \leq \nu$, podemos montar o polinômio $e(x)$ da Equação 3.8. Neste ponto, podemos corrigir a palavra código recebida $r(x)$ através da subtração dos erros $e(x)$, obtendo-se assim, finalmente, a palavra código corrigida $c(x) = r(x) - e(x)$.

3.1.4.2 Algoritmo de Berlekamp-Massey

O decodificador PGZ, apesar de representar um grande avanço em relação ao decodificador teórico do artigo original de Reed e Solomon, requer a inversão de matrizes, o que não é muito eficiente para sua implementação, pois sua complexidade é proporcional ao quadrado do número de erros ν , já a abordagem do algoritmo Berlekamp-Massey apresenta complexidade linear com ν (WICKER, 1994, p. 211).

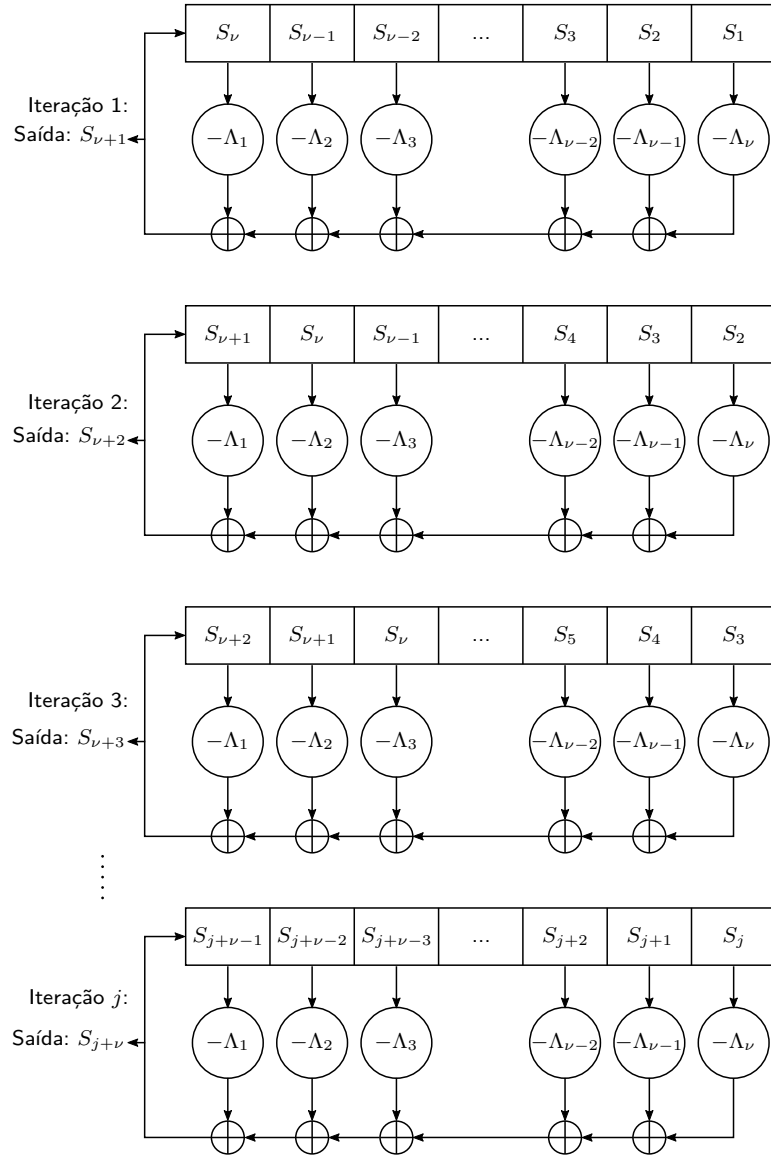
O algoritmo Berlekamp-Massey apresentado nesta seção se baseia na visão de que parte do problema da decodificação dos códigos RS (e dos BCH), a obtenção do polinômio localizador de erros $\Lambda(x)$, é equivalente ao problema de se sintetizar um *registrador de deslocamento com realimentação linear* (*linear feedback shift register - LFSR*) mais curto possível capaz de gerar uma determinada sequência em sua saída.

A Figura 27 ilustra um LFSR comparável ao sistema 3.13 e sua evolução no tempo para j ciclos. A resolução do referido sistema de equações é equivalente a encontrar um LFSR de comprimento mínimo ν , contendo os multiplicadores $-\Lambda_1, -\Lambda_2, -\Lambda_3, \dots, -\Lambda_\nu$ que gere em sua saída a sequência $S_{\nu+1}, S_{\nu+2}, S_{\nu+3}, \dots, S_{2t}$, considerando o estado inicial dos registradores como $S_\nu, S_{\nu-1}, S_{\nu-2}, \dots, S_1$.

O algoritmo 3.1 apresenta o cálculo iterativo do polinômio $\Lambda(x)$, os coeficientes do LFSR, conforme procedimento de Berlekamp-Massey. O algoritmo calcula a cada iteração a variável Δ , a discrepância entre a saída do LFSR e a saída desejada. Se esta discrepância for zero, o polinômio $\Lambda(x)$ calculado até então é mantido, senão, é corrigido. Ao final de $2t$ iterações, obtém-se ν e $\Lambda(x)$.

Para um maior aprofundamento do assunto, pode-se consultar o artigo de Massey (1969), que apresenta a visão original de um dos criadores do algoritmo. Na literatura mais recente, Blahut (2003, p. 183) aborda o funcionamento deste algoritmo de forma mais didática. Outras referências importantes sobre este assunto são: (WICKER; BHARGAVA,

Figura 27: Ilustração do LFSR equivalente



Fonte: o autor, baseada em Stanford University (2015, p. 8)

1999, p. 85), (BERLEKAMP, 2015, p. 179), (WICKER, 1994, p. 219), (LIN; COSTELLO, 2004, p. 209) e (MOON, 2005, p. 253).

3.1.4.3 Algoritmo de Forney

Com a determinação de $\Lambda(x)$ e da consequente obtenção dos números localizadores de erros X_l , $1 \leq l \leq \nu$, é necessário calcular a magnitude dos erros. Como já visto no procedimento de Gorenstein–Zierler, apresentado como parte do decodificador PGZ na subseção 3.1.4.1, a determinação das magnitudes dos erros, Y_l , $1 \leq l \leq \nu$ se dá através da resolução do sistema linear 3.15 que requer a inversão da matriz dos números localizadores de erros X_l .

Algoritmo 3.1 Berlekamp-Massey**Entrada(s):** $t, \{S_1, S_2, \dots, S_{2t}\}$ **Saída(s):** $\nu, \Lambda(x)$

```

1:  $\Lambda(x) \leftarrow 1$ 
2:  $T(x) \leftarrow x$  {polinômio temporário}
3:  $L \leftarrow 0$ 
4: for  $k = 1$  to  $2t$  do
5:    $\Delta \leftarrow \sum_{i=0}^L \Lambda_i S_{k-i}$  {obtem discrepância}
6:   if  $\Delta = 0$  then
7:      $N(x) \leftarrow \Lambda(x)$  {mantém  $\Lambda(x)$ }
8:   else
9:      $N(x) \leftarrow \Lambda(x) - \Delta T(x)$  {calcula novo  $\Lambda(x)$ }
10:    if  $k > 2L$  then
11:       $L \leftarrow k - L$ 
12:       $T(x) \leftarrow \Delta^{-1} \Lambda(x)$ 
13:    end if
14:  end if
15:   $T(x) \leftarrow xT(x)$ 
16:   $\Lambda(x) \leftarrow N(x)$ 
17: end for
18:  $\nu \leftarrow L$ 

```

O algoritmo de Forney (FORNEY, 1965) apresenta uma alternativa mais eficiente para a resolução deste sistema, uma solução fechada que não requer a inversão de matrizes. O procedimento para o cálculo de Y_l , $1 \leq l \leq \nu$, conforme o procedimento de Forney é apresentado a seguir, sem derivações ou provas. Blahut (2003, p. 196), Lin e Costello (2004, p. 247) e Moon (2005, p. 262) discorrem detalhadamente sobre a derivação e o funcionamento deste algoritmo.

Seguem os passos do algoritmo de Forney para o cálculo de Y_l , $1 \leq l \leq \nu$ a partir de $\Lambda(x)$, $S(x)$, ν e t :

1. Calcula-se o polinômio avaliador de erros $\Omega(x)$, conforme a Equação 3.16;
2. Calcula-se $\Lambda'(x)$, a derivada formal de $\Lambda(x)$, definida pela Equação 3.17;
3. Obtém-se a magnitude dos erros Y_i através da Equação 3.18.

$$\Omega(x) = S(x)\Lambda(x) \pmod{x^{2t}} \quad (3.16)$$

$$\Lambda'(x) = \sum_{i=1}^{\nu} i\Lambda_i x^{i-1}; \quad (3.17)$$

$$Y_i = -\frac{X_i^{1-b}\Omega(X_i^{-1})}{\Lambda'(X_i^{-1})} \quad (3.18)$$

3.1.5 Aplicação dos códigos RS no ISDB-T

Esta seção apresenta as especificidades do código RS utilizado pelo padrão ISDB-T, com exemplos de codificação e decodificação.

O codificador e decodificador Reed-Solomon do ISDB-T utiliza o polinômio primitivo da Equação 3.19 para definir o campo de Galois $GF(2^8)$.

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \quad (3.19)$$

A Tabela 31 do Apêndice B apresenta todos os elementos do campo $GF(2^8) = GF(2)[x]/\langle x^8 + x^4 + x^3 + x^2 + 1 \rangle$ gerados através das potências de α , a raiz do polinômio primitivo da Equação 3.19. Conforme a Tabela 31, $\alpha = 2$ na representação decimal.

No padrão ISDB-T, o alfabeto de entrada e saída do RS é o byte. Um byte é composto por 8 bits (zero ou um), portanto, no alfabeto de um byte há $q = 2^8 = 256$ símbolos possíveis. Deste modo, convenientemente, os códigos RS do ISDB-T são baseados no campo de Galois $GF(2^8)$ ($p = 2$ e $m = 8$) de ordem $q = 256$.

Os parâmetros dos códigos RS utilizados pelo ISDB-T são os seguintes:

- Alfabeto de entrada e saída: byte, $q = 2^8 = 256$
- $n = 255$
- $k = 239$
- Comprimento da paridade: 16 símbolos
- Capacidade de correção de erros: até $t = 8$ símbolos errados
- Distância mínima: $d_{min} = 17$
- Encurtamento: $l = 51$

No ISDB-T, a mensagem a ser codificada é um quadro MPEG-2 TS de 188 bytes de comprimento. O poder de correção de erros utilizado no projeto deste sistema de telecomunicação foi $t = 8$. Como não existem códigos RS com $k = 188$, $t = 8$ e $q = 256$, a solução utilizada foi o *encurtamento* do código RS(255, 239) para o RS(204, 188).

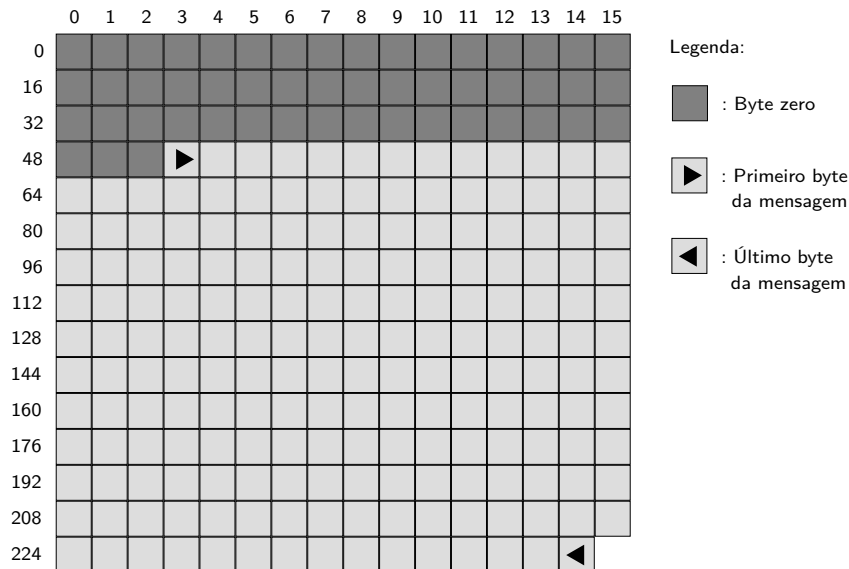
3.1.5.1 Encurtamento do código RS(255, 239) para RS(204, 188)

Quando um código e/ou sua mensagem correspondente não possuem comprimento adequado para sua utilização em uma aplicação específica, uma alternativa utilizada é o encurtamento do código (n, k) para (n', k') , onde $n' = n - l$ e $k' = k - l$ (LIN; COSTELLO, 2004, p. 179).

No codificador RS utilizado no transmissor ISDB-T, o encurtamento sucede a etapa de codificação e também exige a modificação da mensagem de entrada, conforme as etapas a seguir:

1. Alongamento da mensagem: São acrescentados $l = 51$ símbolos *zero* no início da mensagem a ser codificada $m(x)$, formando a mensagem alongada $m'(x)$, conforme ilustrado na Figura 28;
2. Codificação de $m'(x)$: A mensagem alongada (agora com 239 bytes) é submetida ao codificador RS(255, 239), gerando-se o código sistemático $c'(x)$ com 255 bytes, que contém os 16 bytes de paridade no final, conforme a Figura 29;
3. Encurtamento do código: Os 51 zeros que foram inseridos na etapa inicial agora são removidos de $c'(x)$, formando assim o código encurtado $c(x)$ com 204 bytes, conforme Figura 30. Este é o código utilizado na transmissão.

Figura 28: $m'(x)$: Mensagem alongada para 239 bytes, com a inclusão de 51 bytes nulos no início

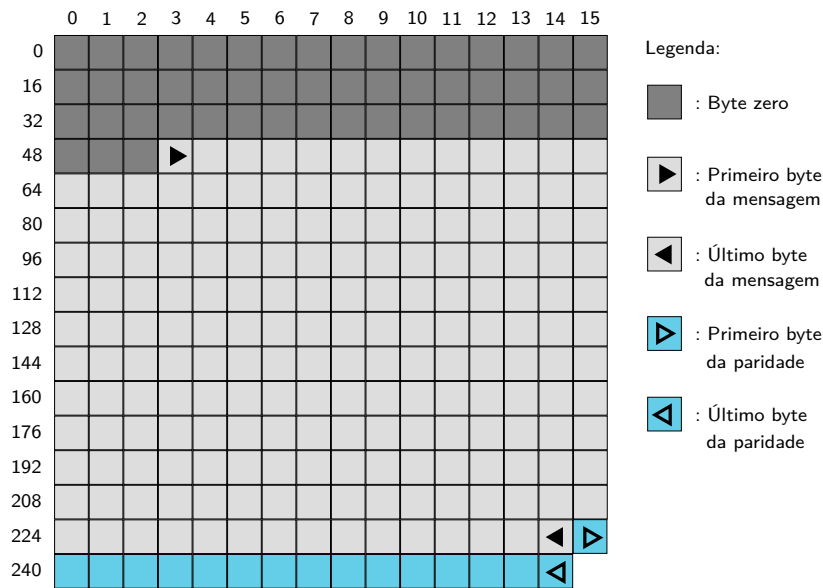


Fonte: o autor

No receptor, o código encurtado (de 204 bytes) recebido $r(x)$ deve seguir as seguintes etapas para poder ser processado pelo decodificador RS(255, 239):

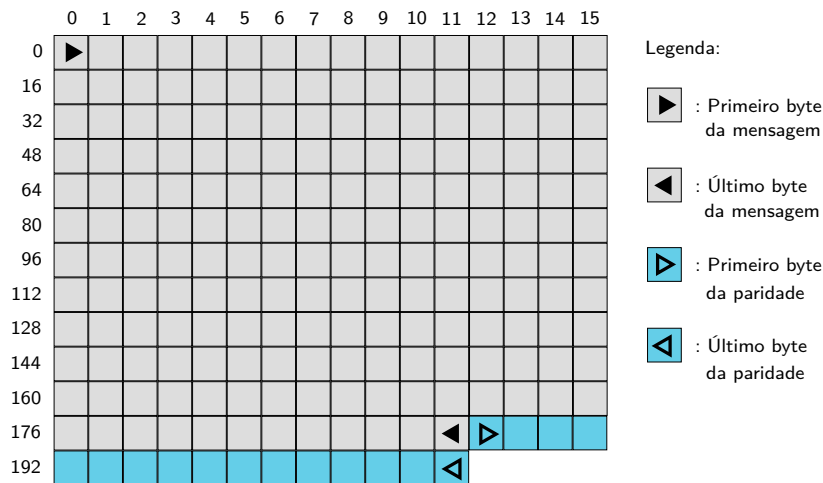
1. Alongamento do código recebido: São acrescentados 51 símbolos *zero* no início do código encurtado (de 204 bytes) recebido $r(x)$, formando o código recebido alongado $r'(x)$, de 255 bytes;

Figura 29: $c'(x)$: Mensagem alongada processada pelo codificador RS(255, 239), que gerou 16 bytes de paridade, resultando em um palavra-código de 255 bytes



Fonte: o autor

Figura 30: $c(x)$: Palavra-código transmitida, com a remoção dos 51 bytes nulos que foram inseridos no início da mensagem antes da codificação



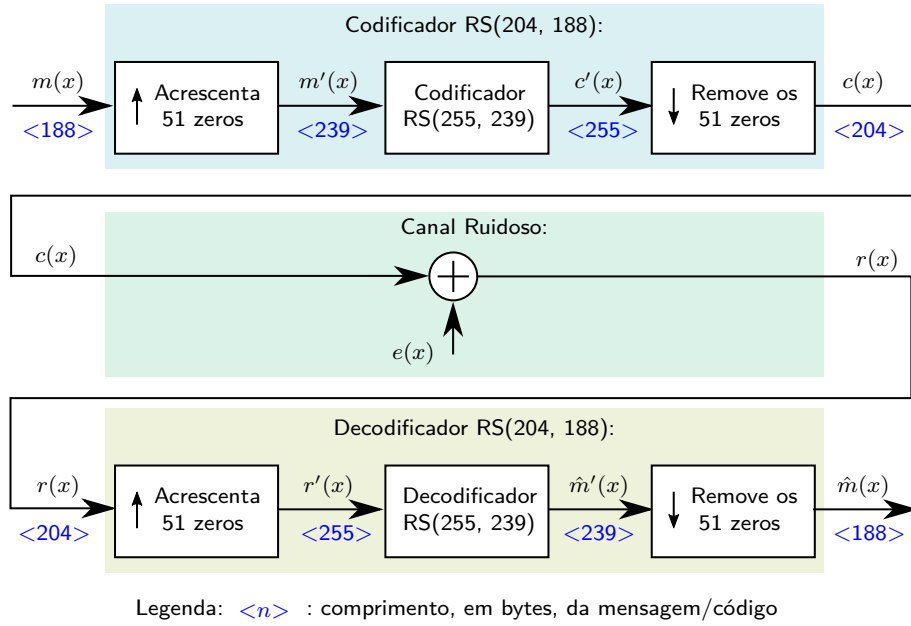
Fonte: o autor

2. Decodificação de $r'(x)$: O código recebido alongado $r'(x)$ é processado pelo decodificador RS(255, 239), gerando-se a mensagem recuperada alongada $\hat{m}'(x)$, de 239 bytes de comprimento;
3. Encurtamento da mensagem recuperada alongada: Os primeiros 51 bytes (zeros) são removidos de $\hat{m}'(x)$, formando assim a mensagem recuperada $\hat{m}(x)$, de 188 bytes.

A Figura 31 resume a geração e a decodificação de um código encurtado Reed-

Solomon do ISDB-T, com a nomenclatura e o comprimento dos vetores utilizados. A figura também apresenta um canal que insere um ruído aditivo $e(x)$ nas palavras código transmitidas. Este canal representa de forma condensada tudo o que há entre o codificador RS no transmissor e o decodificador correspondente no receptor.

Figura 31: Fluxo codificador \rightarrow canal \rightarrow decodificador



Fonte: o autor

3.1.5.2 Codificação

O código RS(255, 239) do ISDB-T utiliza o polinômio gerador $g(x)$ da Equação 3.4 com os parâmetros $2t = n - k = 16$ (número de símbolos de paridade), $b = 0$ (expoente inicial de α) e $\alpha = 2$ (representação decimal da raiz do polinômio primitivo que define o campo). Fazendo estas substituições na referida equação, temos:

$$\begin{aligned}
 g(x) = & (x - 2^0)(x - 2^1)(x - 2^2)(x - 2^3) \\
 & (x - 2^4)(x - 2^5)(x - 2^6)(x - 2^7) \\
 & (x - 2^8)(x - 2^9)(x - 2^{10})(x - 2^{11}) \\
 & (x - 2^{12})(x - 2^{13})(x - 2^{14})(x - 2^{15})
 \end{aligned}$$

Resolvendo as potências de 2 (considerando o campo $GF(256)$), temos:

$$\begin{aligned}
 g(x) = & (x-1)(x-2)(x-4)(x-8) \\
 & (x-16)(x-32)(x-64)(x-128) \\
 & (x-29)(x-58)(x-116)(x-232) \\
 & (x-205)(x-135)(x-19)(x-38)
 \end{aligned} \tag{3.20}$$

Os zeros de $g(x)$ são, portanto: 1, 2, 4, 8, 16, 32, 64, 128, 29, 58, 116, 232, 205, 135, 19 e 38. De forma alternativa, o mesmo polinômio pode ser reescrito na forma da Equação 3.21.

$$\begin{aligned}
 g(x) = & 59 + 36x^1 + 50x^2 + 98x^3 + 229x^4 + 41x^5 + 65x^6 + 163x^7 + \\
 & 8x^8 + 30x^9 + 209x^{10} + 68x^{11} + 189x^{12} + 104x^{13} + 13x^{14} + 59x^{15} + x^{16}
 \end{aligned} \tag{3.21}$$

Tabela 5: **M**: Exemplo de mensagem a ser codificada

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	47	02	21	19	88	c6	df	11	d0	84	4e	8c	9a	37	51	9b
16	16	e9	0a	c6	4f	3d	1c	68	eb	7c	2f	82	5c	98	7c	fc
32	69	30	db	14	2d	6b	ed	b7	92	cd	62	52	bd	9e	49	38
48	cb	e0	67	92	1d	41	53	7f	32	b1	8e	a6	1c	bf	03	0d
64	91	75	ed	bd	44	0e	da	62	f6	cd	19	4c	e3	e5	f6	8e
80	d8	ad	cc	ca	bc	ab	c1	fa	fc	c2	69	97	f1	84	cf	05
96	b8	9c	18	69	cc	cb	02	8f	a6	7d	6b	5f	b6	8e	83	7d
112	dd	4b	0f	fa	d7	4d	15	bb	b0	dd	1f	89	db	9e	99	f1
128	9e	e0	8e	ef	cb	65	2c	e1	5b	97	9b	72	28	a1	2c	ae
144	47	ff	64	7e	e1	56	be	b2	d1	5f	d2	7c	34	b7	8d	db
160	a7	c3	83	28	16	f3	93	27	9d	b7	2e	de	72	cc	28	a6
176	7b	14	75	31	70	eb	40	9e	85	22	e8	65				

Considere como exemplo o quadro MPEG-2 TS de $k' = 188$ bytes apresentado na Tabela 5 como uma sequência de bytes formando o vetor $\mathbf{M} = [M_0, M_1, M_2, \dots, M_{k'-1}]$, com os símbolos na forma hexadecimal. O codificador do ISDB-T associa este vetor \mathbf{M} com os coeficientes do polinômio-mensagem $m(x)$ na ordem inversa:

$$\mathbf{M} = [M_0, M_1, \dots, M_{k'-1}] \rightarrow m(x) = M_{k'-1} + M_{k'-2}x + \dots + M_0x^{k'-1}$$

A partir deste polinômio-mensagem de exemplo $m(x)$, o codificador Reed-Solomon do transmissor ISDB-T gera a palavra-código $c(x)$ apresentada na Tabela 6 na forma do vetor $\mathbf{C} = [C_0, C_1, C_2, \dots, C_{n'-1}]$. Esta sequência possui $n' = 204$ bytes, sendo os primeiros

188 bytes idênticos à mensagem e os $2t = 16$ bytes restantes finais são bytes de paridade, que protegem a mensagem de erros. O mapeamento de $c(x)$ para o vetor \mathbf{C} também é executado de forma inversa, assim como foi feito para o o vetor da mensagem:

$$c(x) = c_0 + c_1x + \cdots + c_{n'-1}x^{n'-1} \rightarrow \mathbf{C} = [c_{n'-1}, c_{n'-2}, \cdots, c_1, c_0] \quad (3.22)$$

A codificação utiliza o polinômio gerador $g(x)$ dado pela Equação 3.21, e consiste do procedimento apresentado na subseção 3.1.3.1 mais as etapas adicionais de alongamento e encurtamento apresentadas na subseção 3.1.5.1.

Tabela 6: \mathbf{C} : Mensagem da Tabela 5 codificada

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	47	02	21	19	88	c6	df	11	d0	84	4e	8c	9a	37	51	9b
16	16	e9	0a	c6	4f	3d	1c	68	eb	7c	2f	82	5c	98	7c	fc
32	69	30	db	14	2d	6b	ed	b7	92	cd	62	52	bd	9e	49	38
48	cb	e0	67	92	1d	41	53	7f	32	b1	8e	a6	1c	bf	03	0d
64	91	75	ed	bd	44	0e	da	62	f6	cd	19	4c	e3	e5	f6	8e
80	d8	ad	cc	ca	bc	ab	c1	fa	fc	c2	69	97	f1	84	cf	05
96	b8	9c	18	69	cc	cb	02	8f	a6	7d	6b	5f	b6	8e	83	7d
112	dd	4b	0f	fa	d7	4d	15	bb	b0	dd	1f	89	db	9e	99	f1
128	9e	e0	8e	ef	cb	65	2c	e1	5b	97	9b	72	28	a1	2c	ae
144	47	ff	64	7e	e1	56	be	b2	d1	5f	d2	7c	34	b7	8d	db
160	a7	c3	83	28	16	f3	93	27	9d	b7	2e	de	72	cc	28	a6
176	7b	14	75	31	70	eb	40	9e	85	22	e8	65	ea	cf	cd	0a
192	d7	79	b9	92	6c	48	f1	97	45	de	f1	98				

3.1.5.3 Decodificação

Considere agora um código $c(x)$ sendo transmitido por um canal ruidoso, cujo efeito sobre o código recebido $r(x)$ é modelado pela Equação 3.23.

$$r(x) = c(x) + e(x) \quad (3.23)$$

O polinômio $e(x)$ possui a forma geral dada pela Equação 3.24.

$$e(x) = e_0 + e_1x + e_2x^2 + e_3x^3 + \cdots + e_{n'-1}x^{n'-1} \quad (3.24)$$

De forma análoga como a palavra código foi mapeada em um vetor na Equação 3.22, os coeficientes do polinômio $e(x)$ podem ser representados na ordem inversa por um vetor $\mathbf{E} = [E_0, E_1, \dots, E_{n'-1}]$:

$$e(x) = e_0 + e_1x + \cdots + e_{n'-1}x^{n'-1} \rightarrow \mathbf{E} = [e_{n'-1}, e_{n'-2}, \cdots, e_1, e_0]$$

Com a definição dos vetores \mathbf{C} e \mathbf{E} , a Equação 3.23 pode ser reescrita para obtermos o vetor \mathbf{R} :

$$\mathbf{R} = \mathbf{C} + \mathbf{E}$$

Considere os seguintes componentes não zeros do vetor \mathbf{E} , e os respectivos coeficientes não zeros do polinômio $e(x)$:

$$\begin{aligned} e_{186} = E_{17} &= 150 \\ e_{178} = E_{25} &= 17 \\ e_{150} = E_{53} &= 34 \\ e_{149} = E_{54} &= 120 \\ e_{102} = E_{101} &= 79 \\ e_{12} = E_{191} &= 51 \end{aligned} \tag{3.25}$$

A expressão acima informa a posição dos $\nu = 6$ erros provocados por um canal ruidoso hipotético e suas magnitudes. A posição dos erros está nos subscritos de \mathbf{E} e e , considerando a representação vetorial e polinomial, respectivamente.

Considerando o código transmitido \mathbf{C} dado pela Tabela 6, e o canal ruidoso provocando erros \mathbf{E} em algumas posições de \mathbf{C} conforme Equação 3.25, no receptor teremos o código recebido \mathbf{R} mostrado na Tabela 7.

Tabela 7: \mathbf{R} : Vetor da Tabela 6 com erros acrescentados

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	47	02	21	19	88	c6	df	11	d0	84	4e	8c	9a	37	51	9b
16	16	7f	0a	c6	4f	3d	1c	68	eb	6d	2f	82	5c	98	7c	fc
32	69	30	db	14	2d	6b	ed	b7	92	cd	62	52	bd	9e	49	38
48	cb	e0	67	92	1d	63	2b	7f	32	b1	8e	a6	1c	bf	03	0d
64	91	75	ed	bd	44	0e	da	62	f6	cd	19	4c	e3	e5	f6	8e
80	d8	ad	cc	ca	bc	ab	c1	fa	fc	c2	69	97	f1	84	cf	05
96	b8	9c	18	69	cc	84	02	8f	a6	7d	6b	5f	b6	8e	83	7d
112	dd	4b	0f	fa	d7	4d	15	bb	b0	dd	1f	89	db	9e	99	f1
128	9e	e0	8e	ef	cb	65	2c	e1	5b	97	9b	72	28	a1	2c	ae
144	47	ff	64	7e	e1	56	be	b2	d1	5f	d2	7c	34	b7	8d	db
160	a7	c3	83	28	16	f3	93	27	9d	b7	2e	de	72	cc	28	a6
176	7b	14	75	31	70	eb	40	9e	85	22	e8	65	ea	cf	cd	39
192	d7	79	b9	92	6c	48	f1	97	45	de	f1	98				

Conforme a subseção 3.1.5.1, antes de ser processada pelo decodificador Reed-Solomon, a palavra-código de 204 bytes recebida $r(x)$ é estendida através do acréscimo de 51 zeros no início, formando assim o vetor \mathbf{R}' de 255 bytes mostrado na Tabela 8. Seu

polinômio equivalente $r'(x)$ é mostrado na Equação 3.26 (os coeficientes do polinômio são apresentados na notação hexadecimal). Observe que os 51 zeros inseridos recaem sobre os termos de maior grau de $r'(x)$, portanto o polinômio $r'(x)$ é igual ao $r(x)$. Isto ocorre devido ao mapeamento inverso entre a representação vetorial e a polinomial no ISDB-T.

Tabela 8: $\mathbf{R}' : \mathbf{R}$ estendido com 51 zeros no início

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
16	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
32	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
48	00	00	00	47	02	21	19	88	c6	df	11	d0	84	4e	8c	9a
64	37	51	9b	16	7f	0a	c6	4f	3d	1c	68	eb	6d	2f	82	5c
80	98	7c	fc	69	30	db	14	2d	6b	ed	b7	92	cd	62	52	bd
96	9e	49	38	cb	e0	67	92	1d	63	2b	7f	32	b1	8e	a6	1c
112	bf	03	0d	91	75	ed	bd	44	0e	da	62	f6	cd	19	4c	e3
128	e5	f6	8e	d8	ad	cc	ca	bc	ab	c1	fa	fc	c2	69	97	f1
144	84	cf	05	b8	9c	18	69	cc	84	02	8f	a6	7d	6b	5f	b6
160	8e	83	7d	dd	4b	0f	fa	d7	4d	15	bb	b0	dd	1f	89	db
176	9e	99	f1	9e	e0	8e	ef	cb	65	2c	e1	5b	97	9b	72	28
192	a1	2c	ae	47	ff	64	7e	e1	56	be	b2	d1	5f	d2	7c	34
208	b7	8d	db	a7	c3	83	28	16	f3	93	27	9d	b7	2e	de	72
224	cc	28	a6	7b	14	75	31	70	eb	40	9e	85	22	e8	65	ea
240	cf	cd	39	d7	79	b9	92	6c	48	f1	97	45	de	f1	98	

$$\begin{aligned}
r'(x) = & \mathbf{98} + \mathbf{f1}x + \mathbf{dex}^2 + \mathbf{45}x^3 + \mathbf{97}x^4 + \mathbf{f1}x^5 + \mathbf{48}x^6 + \mathbf{6cx}^7 \\
& + \mathbf{92}x^8 + \mathbf{b9}x^9 + \mathbf{79}x^{10} + \mathbf{d7}x^{11} + \mathbf{39}x^{12} + \mathbf{cd}x^{13} + \mathbf{cf}x^{14} + \mathbf{eax}^{15} \\
& + \mathbf{65}x^{16} + \mathbf{e8}x^{17} + \mathbf{22}x^{18} + \mathbf{85}x^{19} + \mathbf{9ex}^{20} + \mathbf{40}x^{21} + \mathbf{ebx}^{22} + \mathbf{70}x^{23} \\
& + \mathbf{31}x^{24} + \mathbf{75}x^{25} + \mathbf{14}x^{26} + \mathbf{7bx}^{27} + \mathbf{a6}x^{28} + \mathbf{28}x^{29} + \mathbf{cc}x^{30} + \mathbf{72}x^{31} \\
& + \mathbf{dex}^{32} + \mathbf{2ex}^{33} + \mathbf{b7}x^{34} + \mathbf{9dx}^{35} + \mathbf{27}x^{36} + \mathbf{93}x^{37} + \mathbf{f3}x^{38} + \mathbf{16}x^{39} \\
& + \mathbf{28}x^{40} + \mathbf{83}x^{41} + \mathbf{c3}x^{42} + \mathbf{a7}x^{43} + \mathbf{db}x^{44} + \mathbf{8dx}^{45} + \mathbf{b7}x^{46} + \mathbf{34}x^{47} \\
& + \mathbf{7cx}^{48} + \mathbf{d2}x^{49} + \mathbf{5fx}^{50} + \mathbf{d1}x^{51} + \mathbf{b2}x^{52} + \mathbf{be}x^{53} + \mathbf{56}x^{54} + \mathbf{e1}x^{55} \\
& + \mathbf{7ex}^{56} + \mathbf{64}x^{57} + \mathbf{ff}x^{58} + \mathbf{47}x^{59} + \mathbf{aex}^{60} + \mathbf{2cx}^{61} + \mathbf{a1}x^{62} + \mathbf{28}x^{63} \\
& + \mathbf{72}x^{64} + \mathbf{9bx}^{65} + \mathbf{97}x^{66} + \mathbf{5bx}^{67} + \mathbf{e1}x^{68} + \mathbf{2cx}^{69} + \mathbf{65}x^{70} + \mathbf{cb}x^{71} \\
& + \mathbf{ef}x^{72} + \mathbf{8ex}^{73} + \mathbf{e0}x^{74} + \mathbf{9ex}^{75} + \mathbf{f1}x^{76} + \mathbf{99}x^{77} + \mathbf{9ex}^{78} + \mathbf{db}x^{79} \\
& + \mathbf{89}x^{80} + \mathbf{1fx}^{81} + \mathbf{dd}x^{82} + \mathbf{b0}x^{83} + \mathbf{bb}x^{84} + \mathbf{15}x^{85} + \mathbf{4dx}^{86} + \mathbf{d7}x^{87} \\
& + \mathbf{fax}^{88} + \mathbf{0fx}^{89} + \mathbf{4bx}^{90} + \mathbf{dd}x^{91} + \mathbf{7dx}^{92} + \mathbf{83}x^{93} + \mathbf{8ex}^{94} + \mathbf{b6}x^{95} \\
& + \mathbf{5fx}^{96} + \mathbf{6bx}^{97} + \mathbf{7dx}^{98} + \mathbf{a6}x^{99} + \mathbf{8fx}^{100} + \mathbf{02}x^{101} + \mathbf{84}x^{102} + \mathbf{cc}x^{103} \\
& + \mathbf{69}x^{104} + \mathbf{18}x^{105} + \mathbf{9cx}^{106} + \mathbf{b8}x^{107} + \mathbf{05}x^{108} + \mathbf{cf}x^{109} + \mathbf{84}x^{110} + \mathbf{f1}x^{111} \\
& + \mathbf{97}x^{112} + \mathbf{69}x^{113} + \mathbf{c2}x^{114} + \mathbf{fc}x^{115} + \mathbf{fax}^{116} + \mathbf{c1}x^{117} + \mathbf{ab}x^{118} + \mathbf{bc}x^{119} \\
& + \mathbf{ca}x^{120} + \mathbf{cc}x^{121} + \mathbf{adx}^{122} + \mathbf{d8}x^{123} + \mathbf{8ex}^{124} + \mathbf{f6}x^{125} + \mathbf{e5}x^{126} + \mathbf{e3}x^{127} \\
& + \mathbf{4cx}^{128} + \mathbf{19}x^{129} + \mathbf{cd}x^{130} + \mathbf{f6}x^{131} + \mathbf{62}x^{132} + \mathbf{dax}^{133} + \mathbf{0ex}^{134} + \mathbf{44}x^{135} \\
& + \mathbf{bd}x^{136} + \mathbf{edx}^{137} + \mathbf{75}x^{138} + \mathbf{91}x^{139} + \mathbf{0dx}^{140} + \mathbf{03}x^{141} + \mathbf{bf}x^{142} + \mathbf{1cx}^{143} \\
& + \mathbf{a6}x^{144} + \mathbf{8ex}^{145} + \mathbf{b1}x^{146} + \mathbf{32}x^{147} + \mathbf{7fx}^{148} + \mathbf{2bx}^{149} + \mathbf{63}x^{150} + \mathbf{1dx}^{151} \\
& + \mathbf{92}x^{152} + \mathbf{67}x^{153} + \mathbf{e0}x^{154} + \mathbf{cb}x^{155} + \mathbf{38}x^{156} + \mathbf{49}x^{157} + \mathbf{9ex}^{158} + \mathbf{bd}x^{159} \\
& + \mathbf{52}x^{160} + \mathbf{62}x^{161} + \mathbf{cd}x^{162} + \mathbf{92}x^{163} + \mathbf{b7}x^{164} + \mathbf{edx}^{165} + \mathbf{6bx}^{166} + \mathbf{2dx}^{167} \\
& + \mathbf{14}x^{168} + \mathbf{db}x^{169} + \mathbf{30}x^{170} + \mathbf{69}x^{171} + \mathbf{fc}x^{172} + \mathbf{7cx}^{173} + \mathbf{98}x^{174} + \mathbf{5cx}^{175} \\
& + \mathbf{82}x^{176} + \mathbf{2fx}^{177} + \mathbf{6dx}^{178} + \mathbf{eb}x^{179} + \mathbf{68}x^{180} + \mathbf{1cx}^{181} + \mathbf{3dx}^{182} + \mathbf{4fx}^{183} \\
& + \mathbf{c6}x^{184} + \mathbf{0ax}^{185} + \mathbf{7fx}^{186} + \mathbf{16}x^{187} + \mathbf{9bx}^{188} + \mathbf{51}x^{189} + \mathbf{37}x^{190} + \mathbf{9ax}^{191} \\
& + \mathbf{8cx}^{192} + \mathbf{4ex}^{193} + \mathbf{84}x^{194} + \mathbf{d0}x^{195} + \mathbf{11}x^{196} + \mathbf{df}x^{197} + \mathbf{c6}x^{198} + \mathbf{88}x^{199} \\
& + \mathbf{19}x^{200} + \mathbf{21}x^{201} + \mathbf{02}x^{202} + \mathbf{47}x^{203}
\end{aligned} \tag{3.26}$$

A partir do polinômio $r'(x)$, o primeiro passo do decodificador RS(255, 239) é calcular os 16 coeficientes da síndrome $S = (S_0, S_1, \dots, S_{15})$, através da avaliação de $r'(x)$ nas 16 raízes de $g(x)$: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, conforme a Equação 3.27

$$\begin{aligned}
S_0 &= r'(\alpha^0) = r'(2^0) = r'(1) \\
S_1 &= r'(\alpha^1) = r'(2^1) = r'(2) \\
S_2 &= r'(\alpha^2) = r'(2^2) = r'(4) \\
&\vdots \\
S_{15} &= r'(\alpha^{15}) = r'(2^{15}) = r'(32768)
\end{aligned} \tag{3.27}$$

Calculando a síndrome para o $r'(x)$ de exemplo dado pela Equação 3.26, temos $S = (161, 218, 19, 85, 111, 91, 204, 16, 228, 162, 109, 213, 255, 19, 210, 7)$. A Equação 3.28

apresenta esta mesma síndrome na forma polinomial, $S(x)$.

$$S(x) = 161 + 218x + 19x^2 + 85x^3 + 111x^4 + 91x^5 + 204x^6 + 16x^7 + 228x^8 + 162x^9 + 109x^{10} + 213x^{11} + 255x^{12} + 19x^{13} + 210x^{14} + 7x^{15} \quad (3.28)$$

Como a síndrome calculada não é uma sequência de zeros, sabemos que no vetor de exemplo $r'(x)$ (da Tabela 8 e da Equação 3.26) há erros. A síndrome contém toda a informação necessária para a obtenção do polinômio de erros $e(x)$.

A partir da síndrome S , o algoritmo de Berlekamp-Massey é executado conforme a subseção 3.1.4.2, gerando o polinômio localizador de erros $\Lambda(x)$ e a quantidade de erros de transmissão ν detectados, dados a seguir:

$$\Lambda(x) = 1 + 189x + 78x^2 + 49x^3 + 212x^4 + 188x^5 + 205x^6 + 0x^7 + 0x^8 \quad (3.29)$$

$$\nu = 6$$

As raízes do polinômio localizador de erros $\Lambda(x)$ encontrado são os inversos dos *números localizadores de erros* X_l^{-1} , $1 \leq l \leq \nu$. Calculando-se as referidas raízes de $\Lambda(x)$ através do procedimento de Chien, temos:

$$\begin{aligned} X_1^{-1} &= 26 \\ X_2^{-1} &= 47 \\ X_3^{-1} &= 52 \\ X_4^{-1} &= 60 \\ X_5^{-1} &= 125 \\ X_6^{-1} &= 146 \end{aligned} \quad (3.30)$$

Como $X_l = \alpha^{i_l}$, $1 \leq l \leq \nu$, neste ponto já podemos obter as localizações dos erros i_1, i_2, \dots, i_ν , fazendo o inverso de cada raiz X_l^{-1} de $\Lambda(x)$ e depois aplicando o logaritmo base α :

$$\begin{aligned} i_1 &= \log_\alpha(1/X_1^{-1}) = \log_\alpha(1/26) = \log_\alpha(85) = 150 \\ i_2 &= \log_\alpha(1/X_2^{-1}) = \log_\alpha(1/47) = \log_\alpha(110) = 186 \\ i_3 &= \log_\alpha(1/X_3^{-1}) = \log_\alpha(1/52) = \log_\alpha(164) = 149 \\ i_4 &= \log_\alpha(1/X_4^{-1}) = \log_\alpha(1/60) = \log_\alpha(171) = 178 \\ i_5 &= \log_\alpha(1/X_5^{-1}) = \log_\alpha(1/125) = \log_\alpha(205) = 12 \\ i_6 &= \log_\alpha(1/X_6^{-1}) = \log_\alpha(1/146) = \log_\alpha(68) = 102 \end{aligned} \quad (3.31)$$

Observe que estes valores representam exatamente as posições dos erros inseridos pelo canal, os subscritos i_l de e_{i_l} apresentados na Equação 3.25.

A próxima etapa do decodificador é o cálculo das magnitudes dos erros, $e_{i_l} = Y_l$, $1 \leq l \leq \nu$, através do Algoritmo de Forney, conforme a subseção 3.1.4.3.

Calculando o polinômio avaliador de erros $\Omega(x)$ conforme a Equação 3.16 e os dados deste exemplo, temos:

$$\Omega(x) = \mathbf{161} + \mathbf{161}x + \mathbf{193}x^2 + \mathbf{150}x^3 + \mathbf{141}x^4 + \mathbf{88}x^5 \quad (3.32)$$

A derivada formal $\Lambda'(x)$ é calculada a partir de $\Lambda(x)$ da Equação 3.29, conforme a Equação 3.17, obtendo-se:

$$\Lambda'(x) = \mathbf{189} + \mathbf{49}x^2 + \mathbf{188}x^4 \quad (3.33)$$

A magnitude dos erros $e_{i_l} = Y_l$, $1 \leq l \leq \nu$, pode enfim ser calculada através da aplicação da Equação 3.18, obtendo-se:

$$\begin{aligned} Y_1 &= -\frac{X_1 \cdot \Omega(X_1^{-1})}{\Lambda'(X_1^{-1})} = -\frac{85 \cdot \Omega(26)}{\Lambda'(26)} = -\frac{85 \cdot 26}{57} = 34 \\ Y_2 &= -\frac{X_2 \cdot \Omega(X_2^{-1})}{\Lambda'(X_2^{-1})} = -\frac{110 \cdot \Omega(47)}{\Lambda'(47)} = -\frac{110 \cdot 80}{185} = 150 \\ Y_3 &= -\frac{X_3 \cdot \Omega(X_3^{-1})}{\Lambda'(X_3^{-1})} = -\frac{164 \cdot \Omega(52)}{\Lambda'(52)} = -\frac{164 \cdot 179}{176} = 120 \\ Y_4 &= -\frac{X_4 \cdot \Omega(X_4^{-1})}{\Lambda'(X_4^{-1})} = -\frac{171 \cdot \Omega(60)}{\Lambda'(60)} = -\frac{171 \cdot 91}{215} = 17 \\ Y_5 &= -\frac{X_5 \cdot \Omega(X_5^{-1})}{\Lambda'(X_5^{-1})} = -\frac{205 \cdot \Omega(125)}{\Lambda'(125)} = -\frac{205 \cdot 227}{161} = 51 \\ Y_6 &= -\frac{X_6 \cdot \Omega(X_6^{-1})}{\Lambda'(X_6^{-1})} = -\frac{68 \cdot \Omega(146)}{\Lambda'(146)} = -\frac{68 \cdot 26}{188} = 79 \end{aligned} \quad (3.34)$$

Os valores de Y_l encontrados são exatamente as magnitudes dos erros provocados pelo canal hipotético da Equação 3.25.

Com os resultados das equações 3.31 e 3.34, pode-se montar o conjunto W , apresentado na Equação 3.35, dos pares ordenados que contêm a posição e a magnitude dos erros contidos em $c(x)$: $W = \{(i_l, e_{i_l}) \mid 1 \leq l \leq \nu\}$. A correção dos erros em $c(x)$ é simplesmente a subtração dos erros e_{i_l} nas posições i_l de $c(x)$.

$$W = \{(150, 34), (186, 150), (149, 120), (178, 17), (12, 51), (102, 79)\} \quad (3.35)$$

3.2 Códigos Convolucionais

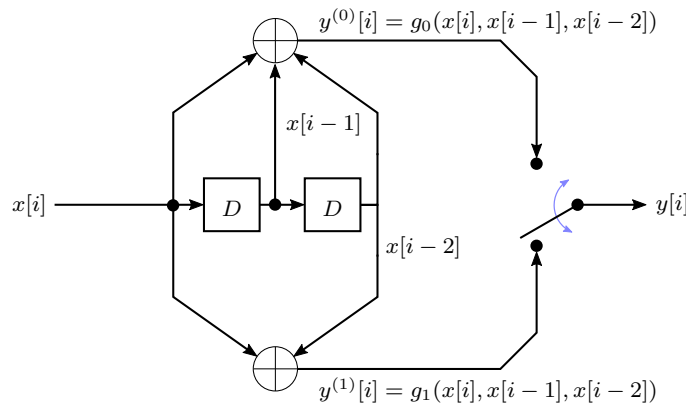
Os códigos convolucionais trabalham com um fluxo contínuo de símbolos, não havendo portanto a segmentação em blocos de informação independentes como ocorre nos códigos de bloco. A notação (n, k) também é utilizada neste tipo de código com o mesmo significado usual: a cada k símbolos de entrada são produzidos n símbolos na saída do codificador, com $n > k$. A taxa do código então, é $R = k/n$. Esta seção abordará somente os códigos convolucionais não recursivos com $k = 1$.

Considere um fluxo contínuo de símbolos $x[i]$ na entrada de um codificador. Na sua concepção mais simples, com $k = 1$, um codificador convolucional trabalha com uma janela deslizante de observação do fluxo de entrada com K símbolos de largura. Esta janela contém o símbolo mais recente, $x[i]$, e os $K - 1$ símbolos anteriores $x[i - 1]$, $x[i - 2]$, \dots , $x[i - K + 1]$ em memória.

Esta memória com capacidade de $M = K - 1$ símbolos pode ser interpretada fisicamente como um registrador de deslocamento (*Shift Register - SR*) de $K - 1$ estágios. A cada novo símbolo disponível na entrada, o conteúdo do SR é deslocado para direita, o novo símbolo da entrada é admitido no estágio mais à esquerda do SR e o conteúdo do estágio mais à direita é descartado.

A Figura 32 mostra um exemplo de um codificador convolucional que trabalha com um alfabeto binário (bits) na entrada e saída, e possui $k = 1$, $n = 2$ e $K = 3$. O comutador na saída deste exemplo de codificador desempenha o papel de um multiplexador, alternando a saída entre $y^{(0)}$ e $y^{(1)}$ com o dobro da taxa de símbolos da entrada.

Figura 32: Exemplo de um codificador convolucional com $K = 3$ e taxa $R = 1/2$



Fonte: o autor

A cada símbolo admitido na entrada são gerados n símbolos na saída: $y^{(0)}$, $y^{(1)}$, \dots , $y^{(n-1)}$. Estas saídas são calculadas através de funções g_0, g_1, \dots, g_{n-1} que associam linearmente os K símbolos dentro da janela de observação atual a uma saída. Portanto,

a saída do codificador depende não somente da entrada atual, mas também das $K - 1$ entradas anteriores.

Cada função linear g_0, g_1, \dots, g_{n-1} é uma soma ponderada das K posições da janela de observação do codificador. Para a j -ésima função g , temos: $g_j = a_{j,0}x[i] + a_{j,1}x[i - 1] + \dots + a_{j,K-1}x[i - K + 1]$. Para o caso do alfabeto binário, $q = 2$, os coeficientes $a_{j,0}, a_{j,1}, \dots, a_{j,K-1}$ podem ser apenas zeros ou uns, representando portanto a presença ou ausência da ligação de cada uma das K posições da janela de observação a um somador módulo 2.

Os coeficientes de uma função g podem ser representados por diferentes formas:

- Através do *polinômio gerador*, que possui a forma geral $g(D) = a_0 + a_1D^1 + \dots + a_{K-1}D^{K-1}$, onde D é o *operador de atraso unitário*. Para o caso do exemplo da Figura 32, $g_0(D) = 1 + D + D^2$ e $g_1(D) = 1 + D^2$;
- Pelo vetor de seus coeficientes ordenados. No exemplo dado $g_0 = [1, 1, 1]$ e $g_1 = [1, 0, 1]$;
- Representação *octal* dos coeficientes, na forma de vetor $g = [d_0, d_1, \dots, d_{u-1}]$, onde d_i é um dígito octal ($0 \leq d_i \leq 7$) calculado através da conversão binário para octal de 3 coeficientes binários a_j, a_{j+1}, a_{j+2} consecutivos. No exemplo dado, $g_0 = [7]$ e $g_1 = [5]$.

É possível observar que cada saída $y^{(0)}, y^{(1)}, \dots, y^{(n-1)}$ é o resultado da convolução da sequência de entrada com o vetor dos coeficientes da função g correspondente (que pode ser interpretado como sua resposta impulsiva). Daí advém o nome *convolucional* destes códigos. Seu funcionamento também pode ser comparado aos filtros de resposta impulsiva finita *Finite Impulse Response - FIR*.

O parâmetro K é conhecido como o *comprimento de restrição* (*constraint length*) do código, e neste trabalho é definido como sendo $K = M + 1$, onde M é o número de elementos de memória (estágios do registrador de deslocamento). Portanto, K representa a quantidade de símbolos da entrada (atual + M anteriores) que influenciam no cálculo da saída do codificador. Esta definição está de acordo com Viterbi e Omura (1979), Clark Jr e Cain (1981), Haykin (2014) e a própria norma do ISDB-T, ARIB (2014, p. 27). Por outro lado, alguns autores como Blahut (2003), Lin e Costello (2004), Johannesson e Zigangirov (1999) utilizam a definição $K = M$.

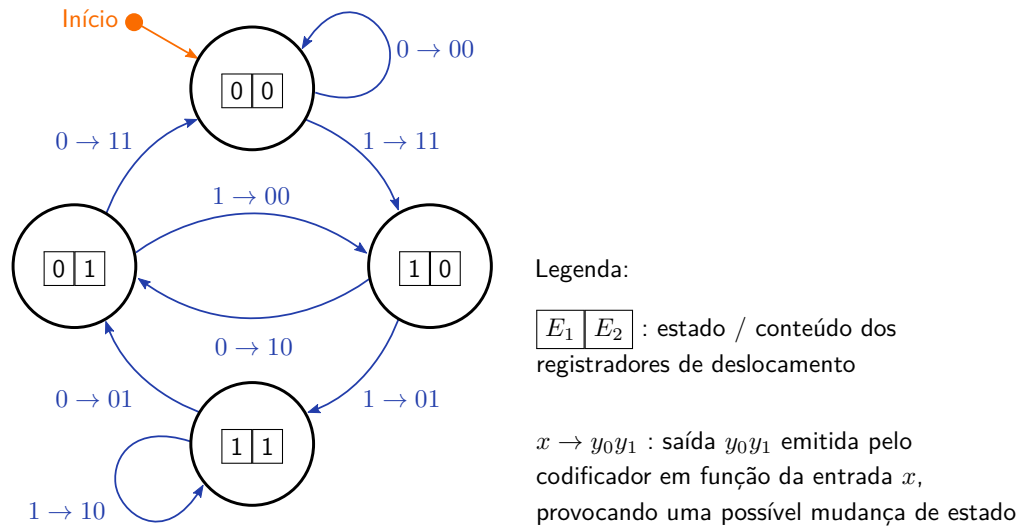
Em síntese, os principais parâmetros de um código convolucional são:

- q : Número de símbolos no alfabeto de entrada e saída;
- n : Quantidade de símbolos produzidos na saída a cada unidade de tempo;

- k : Quantidade de símbolos consumidos da entrada a cada unidade de tempo;
- K : Comprimento de restrição. $K = M + 1$, onde M é a quantidade de elementos de memória (comprimento do registrador de deslocamento).

Uma abordagem alternativa importante para um codificador convolucional é a do *diagrama de transição de estados*, ilustrada na Figura 33. Esta visão apresenta a evolução da codificação como uma alternância entre os $q^{K-1} = 2^{3-1} = 4$ estados. Cada estado é representado por um círculo e é nomeado com o conteúdo dos $K - 1$ estágios do registrador de deslocamento do codificador. Quando há a disponibilidade de um novo bit x na entrada, são gerados dois bits de saída y_0y_1 que dependem do estado atual e do valor de x . Neste instante também pode ocorrer uma mudança de estado, representada por setas.

Figura 33: Codificador convolucional da Figura 32 representado em um diagrama de transição de estados

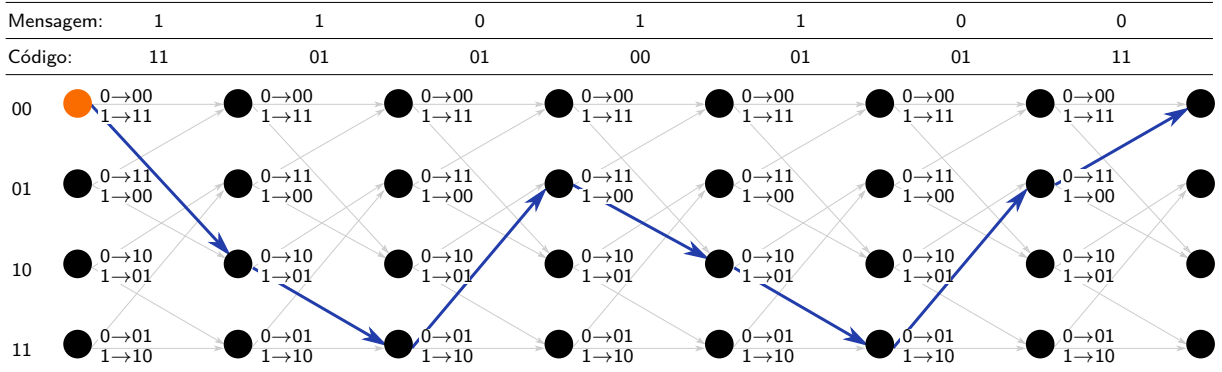


Fonte: o autor

Outra forma de representar um código convolucional é o diagrama de treliça. A Figura 34 apresenta a treliça para o codificador da Figura 32, com um exemplo da codificação da mensagem binária $[1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0]$ considerando o estado inicial 00, que resulta no código $[11 \ 01 \ 01 \ 00 \ 01 \ 01 \ 11]$.

A treliça é importante no estudo da decodificação do código convolucional pelo algoritmo de Viterbi, por registrar a evolução temporal do código em t etapas. Cada etapa é composta pelos 2^{K-1} estados possíveis do codificador, e de cada estado partem 2^k ramos em direção a um estado da etapa seguinte. De forma similar ao do diagrama de transição de estados, em cada ramo da treliça (que indica uma possível mudança de estado) é indicada a saída y_0y_1 emitida pelo codificador em função de uma entrada x , através da representação $x \rightarrow y_0y_1$.

Figura 34: Exemplo de codificação através da treliça



Fonte: o autor

3.2.1 Distância livre

Nos códigos convolucionais, a métrica *distância livre* d_{free} fornece uma medida importante do poder de correção de erros do código. Quando utilizada a decodificação por máxima verossimilhança (*maximum likelihood - ML*), um código convolucionacional pode corrigir t erros *relativamente próximos*, desde que $t \leq \lfloor (d_{free} - 1)/2 \rfloor$. O comprimento do trecho considerado para a ocorrência dos erros que podem ser corrigidos é de 3 a 5 comprimentos de restrição K , dependendo do padrão do erro (SKLAR, 2001, p. 413).

O cálculo de d_{free} é baseado na inspeção do diagrama de transição de estados do código. Partindo do estado inicial zero (E0), São avaliados todos os caminhos possíveis que tenham como destino final o próprio estado zero, exceto o caminho trivial $0 \rightarrow 0$. Na análise de cada caminho, são acumulados os pesos de hamming (números de bits um) das saídas do codificador. A menor soma de pesos obtida é a distância livre d_{free} . Para o caso do codificador de exemplo da Figura 32, $d_{free} = 5$, encontrado através do caminho $00 \rightarrow 10 \rightarrow 01 \rightarrow 00$.

O comprimento de restrição K influencia a distância livre de um código, quanto maior K , maior será d_{free} . A taxa R do código também, quanto menor a taxa, mais robusto é o código. E por fim, a escolha de bons polinômios geradores é imprescindível para a obtenção dos melhores códigos possíveis para um determinado K e R . Proakis e Salehi (2008, p. 517–520) apresentam diversas tabelas com os melhores polinômios geradores para diversas combinações de K e R .

A distância livre não é capaz de caracterizar completamente o poder de correção de erros de um código convolucionacional. Sklar (2001, cap. 7), Proakis e Salehi (2008, cap. 8) apresentam em detalhes o cálculo da *função de transferência* do código e os limites de erros destes códigos.

3.2.2 Puncionamento

Os códigos convolucionais com taxa $R = 1/n$ ($k = 1$) possuem uma grande importância prática, pois seus codificadores e principalmente seus decodificadores são mais simples de se implementar. Porém para muitas aplicações, taxas $R = 1/n$, ou seja, $R \leq 1/2$, são consideradas baixas. É possível obter taxas de codificação R maiores a partir de um *código-mãe* de taxa $R = 1/n$.

A técnica utilizada é o *puncionamento* (ou *perfuração*, *puncturing*) do código, que, em síntese, modifica o código através da supressão periódica de alguns símbolos da saída do codificador-mãe, omitindo-os da transmissão. Isto apresenta vantagens, pois permite a obtenção de taxas maiores e flexíveis a partir de um único codificador/decodificador convolucional de taxa fixa $R = 1/n$, de implementação simples.

Considere a *matriz de puncionamento* $\mathbf{P} = [p_{i,j}]$ de dimensão $n \times v$, com $0 \leq i \leq n - 1$ e $0 \leq j \leq v - 1$, composta por nv elementos binários $p_{i,j}$. Quando um elemento $p_{i,j}$ for igual a 1, significa que as saídas do codificador $y_u^{(i)}$ que satisfizerem a congruência $j \equiv u \pmod{v}$ serão emitidas normalmente em seus tempos; quando igual a 0, significa sua supressão.

Por exemplo, considere a aplicação da matriz de puncionamento \mathbf{P} de dimensão 2×3 apresentada em 3.36 no código gerado pelo codificador $R = 1/2$ da Figura 32. O puncionamento da saída daquele codificador resulta na emissão dos símbolos: $y = [y_0^{(0)}, y_0^{(1)}, -, y_1^{(1)}, y_2^{(0)}, -, \dots]$, onde o hífen indica um símbolo suprimido. Este padrão de supressão se repete indefinidamente. As setas vermelhas desenhadas sobre a matriz 3.36 indicam a sequência de avaliação da supressão para cada símbolo na saída do codificador.

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad (3.36)$$

Seja $s = n \cdot v$ o total de elementos da matriz de puncionamento, $t = \sum_{i=0}^{n-1} \sum_{j=0}^{v-1} p_{i,j}$ o número de *uns* contidos na mesma matriz e R a taxa original do código-mãe. A nova taxa R' do código puncionado será:

$$R' = R \cdot \frac{s}{t}$$

Para o caso do exemplo do codificador da Figura 32 e da matriz 3.36, temos $R' = (1/2) \cdot (6/4) = 3/4$. A matriz de puncionamento também pode ser reescrita na forma de um vetor linha, com a mesma quantidade de elementos, sem causar ambiguidade.

A matriz de puncionamento normalmente é obtida através de uma prospecção por

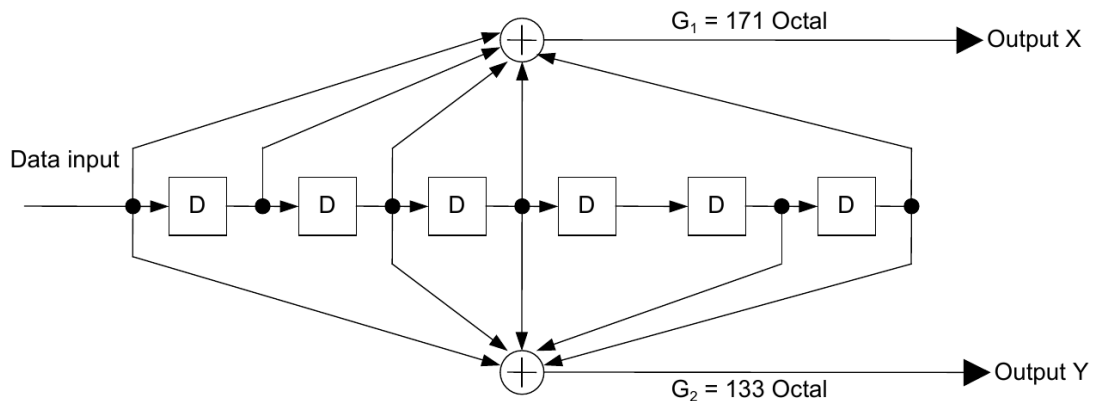
computador, com o objetivo de se obter a melhor matriz para uma taxa de código desejada, no sentido de maximizar o poder de correção de erros. Proakis e Salehi (2008, p. 522) apresentam estas matrizes para diversas taxas.

O aumento da taxa do código puncionado de R para R' causa, naturalmente, uma redução no seu poder de correção de erros. Mas, felizmente, o desempenho do código puncionado é equivalente aos melhores códigos de taxa nativa R' , havendo uma perda no ganho de codificação de 0,1 a 0,2 dB apenas (CAIN; CLARK JR; GEIST, 1979).

3.2.3 Aplicação dos códigos convolucionais no ISDB-T

O ISDB-T utiliza um código-mãe de taxa $R = 1/2$, com comprimento de restrição $K = 7$. Este código utiliza os polinômios geradores $g_0(D) = 1 + D + D^2 + D^3 + D^6$ (octal 171) e $g_1(D) = 1 + D^2 + D^3 + D^5 + D^6$ (133 em octal). Este é um código com a maior distância livre possível para a sua taxa e comprimento de restrição, possuindo $d_{free} = 10$. O codificador é apresentado na Figura 35.

Figura 35: Codificador convolucional utilizado no ISDB-T



Fonte: ARIB (2014, p. 27)

O código convolucional do ISDB-T possui taxa configurável através do puncionamento do código-mãe. As taxas possíveis são: $1/2$, $2/3$, $3/4$, $5/6$ e $7/8$. As matrizes de puncionamento para cada taxa são apresentadas na Tabela 9.

3.2.4 Decodificação pelo Algoritmo de Viterbi

Assim como ocorre com os códigos de bloco, a decodificação dos códigos convolucionais é muito mais complexa do que a sua codificação. Basicamente há dois métodos importantes de decodificação para os códigos convolucionais: O algoritmo de Viterbi (*Viterbi Algorithm* - VA) e a decodificação sequencial. O VA provê a decodificação ótima, ou seja, é garantido que a sua saída é a sequência mais provável transmitida. O VA é

Tabela 9: Padrões de punçãoamento do código convolucional do ISDB-T

Coding rate	Puncturing pattern	Transmission-signal sequence
1/2	X : 1 Y : 1	X1, Y1
2/3	X : 1 0 Y : 1 1	X1, Y1, Y2
3/4	X : 1 0 1 Y : 1 1 0	X1, Y1, Y2, X3
5/6	X : 1 0 1 0 1 Y : 1 1 0 1 0	X1, Y1, Y2, X3 Y4, X5
7/8	X : 1 0 0 0 1 0 1 Y : 1 1 1 1 0 1 0	X1, Y1, Y2, Y3, Y4, X5, Y6, X7

Fonte: ARIB (2014, p. 27)

adequado para códigos com $K \leq 10$, enquanto que a decodificação sequencial é empregada para códigos com $K > 10$. Esta seção abordará a decodificação de códigos convolucionais através do algoritmo de Viterbi.

Em 1967, Andrew James Viterbi publicou o artigo *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm* (VITERBI, 1967). Nascia naquele momento o algoritmo de Viterbi, um marco importante na área das telecomunicações. Curiosamente, na ocasião daquela publicação, o autor não sabia que tinha inventado um decodificador ótimo, e duvidava que ele seria prático (FORNEY, 2005, p. 2).

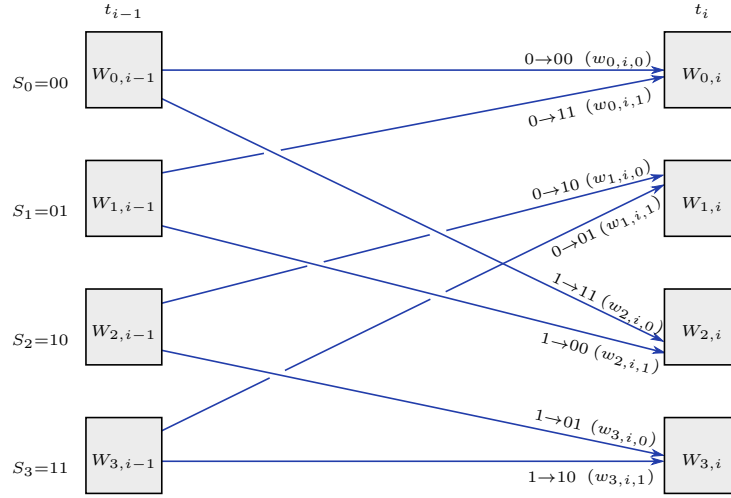
O artigo original do VA se baseia na construção do código em um *diagrama de árvore*. Em 1973, George David Forney Jr. utilizou pela primeira vez o diagrama de treliça para interpretar o VA, e descobriu que se tratava de um decodificador ótimo (FORNEY, 1973). A visão da treliça é útil até hoje para se estudar os códigos convolucionais, principalmente sua decodificação pela VA.

O VA, além de ser utilizado para a decodificação convolucional, se tornou um algoritmo de propósito geral para decodificar modelos escondidos de Markov em diversas aplicações como biologia computacional e reconhecimento de fala.

Considere a Figura 36, que apresenta um trecho da treliça da Figura 34 contendo apenas a *etapa atual* t_i e a *etapa anterior*, t_{i-1} . Em cada etapa, os 2^{K-1} estados $S_0, S_1, \dots, S_{2^{K-1}-1}$ são representados por retângulos na figura. Cada estado de cada etapa possui uma *métrica do caminho* (*path metric*) $W_{j,s}$ associada, onde j é o número que identifica o estado e s é subscrito da etapa t_s que a identifica.

Todas as transições de estado possíveis entre a etapa anterior e a etapa atual são representadas por setas direcionadas, chamadas de *ramos* (*branches*). Os $P = 2^k$ estados de uma etapa anterior que são ligados um determinado estado da etapa atual são chamados de *estados predecessores*.

Figura 36: Um trecho da treliça



Fonte: o autor

Junto com cada ramo há a notação $x \rightarrow y (w_{j,i,u})$, onde y é a saída emitida pelo codificador em função da entrada x e $w_{j,i,u}$ é a *métrica do ramo* (*branch metric - BM*), onde j é o número que identifica o estado de destino na etapa t_i e u identifica cada um dos P ramos que tem o estado $W_{j,i}$ como destino. Há portanto N_{MR} métricas de ramo em cada etapa da treliça:

$$N_{MR} = P \cdot 2^{K-1} \quad (3.37)$$

Após as definições acima, os passos do algoritmo de Viterbi podem ser enunciados:

1. Inicialização do decodificador: Na etapa zero da treliça, t_0 , as métricas de caminho são inicializadas, com $W_{0,0} = 0$ e $W_{j,0} = \infty$, $1 \leq j \leq 2^{K-1} - 1$, ou seja, o estado S_0 possuirá métrica zero e nos demais a métrica infinita (na prática um valor alto, como por exemplo o maior inteiro possível no sistema em que o decodificador for implementado);
2. Cálculo das métricas de ramo (*BM calculation*): Para cada grupo de n bits recebidos pelo decodificador na etapa atual, são calculadas todas as N_{MR} métricas dos ramos que convergem para os estados da etapa atual. Estas métricas são as *distâncias de Hamming* entre os n bits recebidos e as saídas y do codificador associadas a cada ramo;
3. Cálculo das métricas de caminho para a etapa atual da treliça, através dos passos ACS (Add-Compare-Select):

- a) Soma (*add*): As métricas de caminho associadas a cada estado predecessor são somadas às métricas de ramo correspondentes, dando origem à P métricas de caminho candidatas para cada estado da etapa atual;
 - b) Compara (*compare*): Para cada estado da etapa atual, as P métricas de caminho candidatas são comparadas;
 - c) Seleciona (*select*): Para cada estado j da etapa atual, é selecionada a melhor (menor) das P métricas de caminho candidatas que convergem para aquele estado j . No caso de empate, pode-se selecionar aleatoriamente. A métrica selecionada torna-se então a nova métrica de caminho associada ao estado em questão, $W_{j,i}$. O ramo associado à métrica selecionada torna-se parte de um caminho sobrevivente. Todos os ramos sobreviventes são armazenados em uma memória dos estados predecessores.
4. Rastreamento reverso (*traceback*): A partir de uma determinada etapa h da treliça (comprimento do traceback), é possível obter a sequência correta dos bits recebidos (e por consequência, dos bits transmitidos), corrigindo eventuais erros provocados pelo canal ruidoso. O rastreamento começa no estado que possui a menor métrica de caminho acumulada na etapa t_h , e segue em direção às etapas anteriores, t_{h-1}, t_{h-2}, \dots , percorrendo o caminho sobrevivente formado pelos ramos selecionados nos passos ACS, através de consultas na memória de estados predecessores. Durante este percurso, são emitidos os bits de saída do decodificador.

As figuras 37 e 38 mostram os passos de decodificação do VA, considerando como exemplo a mensagem [1 1 0 1 1 0 0] codificada pelo codificador da Figura 32, que gerou o código [11 01 01 00 01 01 11]. Este código é transmitido por um canal ruidoso que provoca dois bits errados no código recebido: o quarto e o sétimo bit, sendo o código recebido portanto [11 00 01 10 01 01 11].

A Figura 37a mostra o decodificador inicializado com as métricas de caminho padrões para a etapa t_0 : $[0, \infty, \infty, \infty]$. O primeiro par de bits [11] é recebido, e com ele todas as 8 métricas de ramo são calculadas através da distância de Hamming. As métricas de ramo são somadas às métricas de caminho dos estados predecessores, e as melhores (menores) são selecionadas. Os ramos selecionados são indicados na figura através de traços vermelhos negritos. Outros dois pares de bits são recebidos: [00] e [01], e o processo descrito se repete nas figuras 37b e 37c, avançando até a etapa t_3 da treliça.

A Figura 38a mostra a treliça após todos os 7 pares de bits serem recebidos pelo decodificador. Observe que há vários caminhos que não chegam até o fim da treliça, estes são os caminhos não-sobreviventes, removidos na Figura 38b para maior clareza.

A Figura 38c ilustra o passo *rastreamento reverso* do VA. Inicia-se na última etapa da treliça, t_7 , pelo estado com menor métrica de caminho acumulada, o estado S_0 que

acumulou $W_{0,7} = 2$. A partir daí, o caminho reverso é percorrido através dos ramos selecionados até então: $S_0 \leftarrow S_2 \leftarrow S_3 \leftarrow S_1 \leftarrow S_2 \leftarrow S_3 \leftarrow S_1 \leftarrow S_0$. Este caminho é ilustrado na figura através da linha vermelha pontilhada, e representa a melhor estimativa da mensagem transmitida.

O VA faz a decodificação ML de uma maneira inteligente, não necessitando considerar todos os 2^L caminhos possíveis para uma mensagem de L bits, mas apenas 2^{K-1} deles, qualquer que seja o comprimento da mensagem. O número de caminhos a avaliar, portanto, depende apenas do comprimento de restrição K do código. O VA consegue esta eficiência computacional basicamente pela rejeição dos caminhos improváveis (não sobreviventes) a cada etapa da treliça, concentrando-se portanto nos caminhos sobreviventes.

Na prática o VA é utilizado para códigos com $K \leq 10$. Acima deste valor, geralmente é utilizado outro método de decodificação mais simples, não ML, como por exemplo a *decodificação sequencial*.

Quando o codificador utiliza um código puncionado, o VA pode ser utilizado com pequenas modificações: 1. Há a necessidade da obtenção, pelo receptor, do sincronismo da matriz de puncionamento \mathbf{P} (Equação 3.36); 2. Nos instantes equivalentes em que bits foram suprimidos pelo puncionador no transmissor, o receptor deve inserir símbolos de rasura (*erasures*) no fluxo de bits na entrada do VA, normalmente representados pela letra X na literatura. Estas rasuras são processadas pelo VA normalmente, porém não influem no cálculo das métricas. Por exemplo, a distância de Hamming entre [1X] e [10] ou entre [1X] e [11] é zero em ambos os casos..

Quando o VA processa uma sequência finita e curta, o comprimento do traceback h pode ser igual ao tamanho da sequência recebida. Porém quando esta sequência é longa ou infinita (que é o caso do fluxos utilizados pela TV digital), faz-se necessário usar um h pequeno o suficiente para não causar atrasos significativos na decodificação e para reduzir o tamanho necessário da memória dos estados predecessores do VA. Isto é chamado de *truncamento* do código. O truncamento faz com que o VA não seja mais um método de decodificação ML, porém, escolhendo-se um valor de h grande o suficiente, é possível obter um desempenho muito próximo ao do ML.

Para não haver perda de desempenho significativa pelo truncamento do código, de acordo com Lin e Costello (2004, p. 548), para um código não puncionado de taxa $1/2$, h deve ser no mínimo igual a 4 ou 5 vezes o comprimento de restrição K . Já para códigos de taxa $3/4$, h deve ser maior ou igual a $10K$ (CLARK JR; CAIN, 1981, p. 263).

Para o caso do truncamento do código, mesmo escolhendo adequadamente o comprimento do traceback, há uma nuance que afeta de maneira significativa o desempenho do VA. A Figura 39 apresenta dois métodos de traceback: o simples e o duplo. Para o método simples, a cada h etapas da treliça é executado um traceback a partir do estado

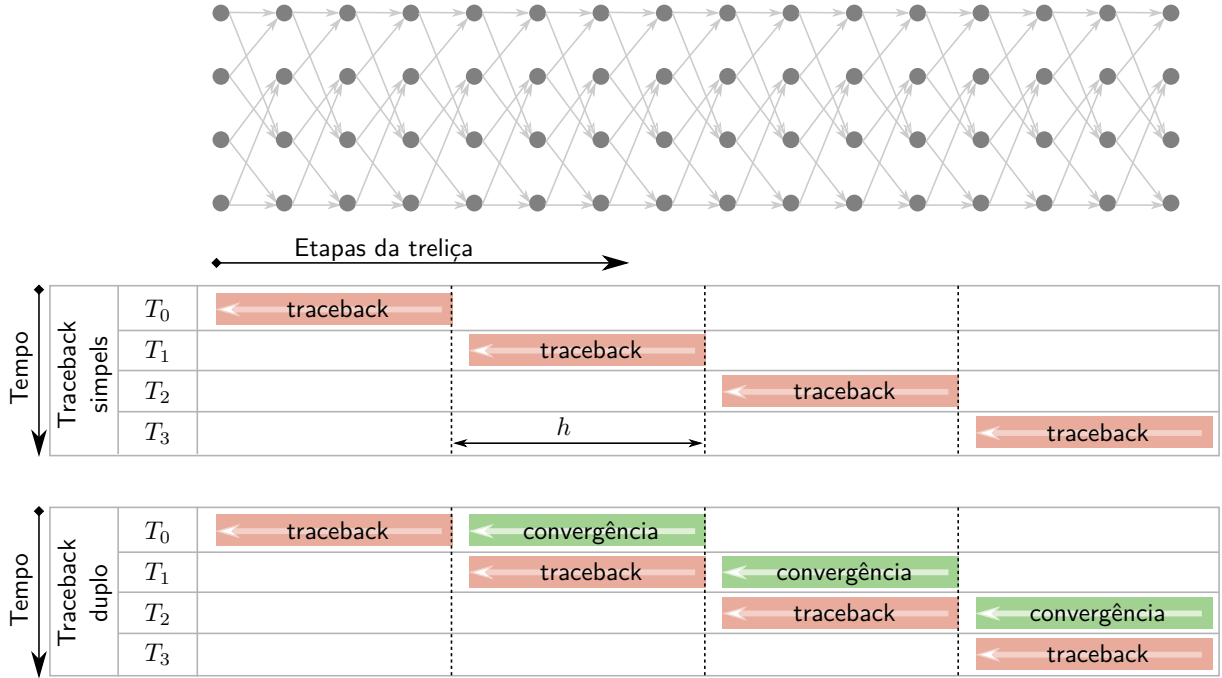
Figura 38: Exemplo de decodificação através do algoritmo de Viterbi - parte 2



Fonte: o autor

que possuir a melhor métrica de caminho acumulada naquele instante, sendo emitidos como saída os bits decodificados.

Figura 39: Traceback simples e duplo



Fonte: o autor

No método duplo, a cada h etapas da treliça¹ são executados dois passes: a convergência e o traceback. O passe da convergência é idêntico ao traceback do método simples, mas não emite saída. Seu único propósito é a obtenção do melhor estado para se iniciar o próximo passe, o traceback propriamente dito, com emissão de saída. Observe que há uma sobreposição entre as janelas de execução deste método.

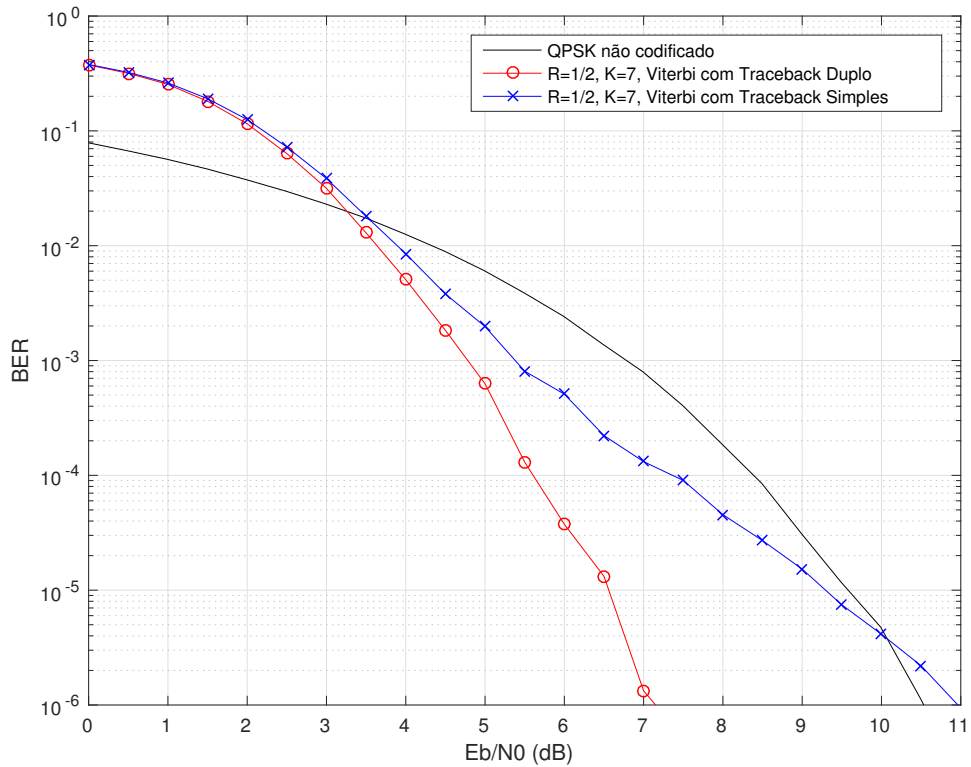
O VA que utiliza o traceback duplo apresenta desempenho muito superior se comparado ao que utiliza o método simples, pois o passe de convergência elimina possíveis opções de ramos ruins, que ainda não convergiram para um caminho comum. É possível compreender melhor este fenômeno observando o exemplo da Figura 38b, onde as últimas 4 etapas da treliça (t_7, t_6, t_5, t_4) apresentam várias opções de caminhos, mas todos eles convergem para um único caminho em comum a partir da etapa t_3 em direção às etapas anteriores, independentemente da escolha do estado com menor métrica de caminho em t_7 para se iniciar o traceback.

O passe de convergência do método duplo evita estes caminhos ainda não consolidados, reduzindo a taxa de erros na saída do decodificador, conforme pode ser verificado na Figura 40, que mostra a simulação do VA para os dois casos, considerando o código de taxa 1/2 com comprimento de restrição $K = 7$ da Figura 35 em um canal AWGN. O

¹ Exceto para a primeira execução, T_0 , que é iniciada apenas após $2h$ etapas da treliça

método duplo apresenta uma vantagem de cerca de 4 dB para a taxa de erro de bit de 10^{-6} .

Figura 40: Comparação do desempenho entre o método do traceback simples e do duplo



Fonte: o autor

A decodificação apresentada até aqui se baseou em entradas binárias, normalmente obtidas de um demodulador que aplica *decisões rígidas* (*hard decisions*), geralmente através da comparação do nível de sinal na entrada com um limiar, resultando no bit que pode assumir o valor zero ou um. Neste caso, o conjunto [modulador \rightarrow canal AWGN \rightarrow demodulador] pode ser visto como um canal binário simétrico (*Binary Symmetric Channel* - *BSC*) com probabilidade de erro p .

Há outra construção possível para o demodulador (e consequentemente para o decodificador Viterbi), que aumenta de forma significativa a performance de decodificação: o uso de *decisões brandas* (*soft decision*). O demodulador entrega para o decodificador não mais um feixe binário, mas sim um feixe de *soft bits* que carregam consigo a probabilidade daquele símbolo ser zero ou um.

A decisão branda pode ser de dois tipos: não quantizada e quantizada. No primeiro tipo, o soft bit na saída do demodulador é um número real v , por exemplo na faixa $-1 \leq v \leq +1$. Quanto mais próximo de $+1$, maior a chance de que o bit 1 foi transmitido, quanto mais próximo de -1 , maior a probabilidade de que o bit 0 foi transmitido. Valores intermediários de v significam um grau maior de dúvida sobre o bit transmitido.

Na decisão branda quantizada, o soft bit v na saída do demodulador é um número inteiro que pode ser interpretado como sendo uma versão discreta, com l níveis, da decisão branda não quantizada. Normalmente l é uma potência de 2 para permitir a representação de v por um conjunto de q bits, sendo portanto $l = 2^q$. Por exemplo, no caso de $l = 2^3 = 8$, a saída do demodulador pode produzir $0 \leq v \leq 7, v \in \mathbb{Z}$, onde 0 significa muita certeza que o bit 0 foi transmitido, 7 significa muita certeza que o bit 1 foi transmitido, e valores intermediários indicam diferentes graus de confiança que um determinado bit foi transmitido. O caso $q = 1$ é equivalente à decisão rígida, já $q \rightarrow \infty$ é o mesmo que a decisão branda não quantizada, pois haveriam infinitos níveis de quantização.

A decisão branda é vantajosa pois entrega a informação da confiabilidade de cada bit, deixando para o decodificador escolher a sua saída baseada na distância Euclidiana e não na distância de Hamming. Isto traz ganhos de performance importantes para o decodificador.

A Figura 41 apresenta curvas BER x E_b/N_0 considerando o código de taxa 1/2 com comprimento de restrição $K = 7$ da Figura 35 operando sob um canal AWGN. Há uma curva para o caso da decisão rígida ($q = 1$) e 2 curvas para decisões brandas quantizadas ($q = 2$ e $q = 3$). É possível observar que na taxa de erro de bit de 10^{-6} , há um ganho de cerca de 2 dB a favor da decisão branda quantizada de 8 níveis ($q = 3$) quando comparada à decisão rígida ($q = 1$), valor compatível com as simulações de Lin e Costello (2004, p. 554) e de (SKLAR, 2001, p. 426).

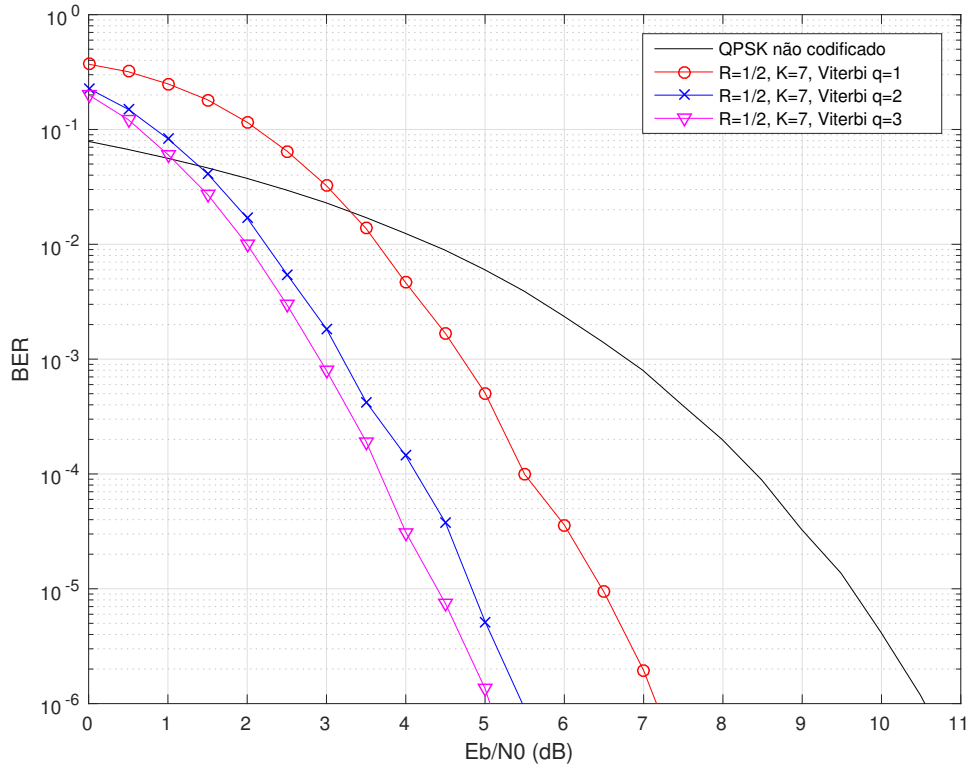
Na decisão branda quantizada, geralmente são utilizados 8 níveis de quantização ($q = 3$). Acima disto, o ganho de desempenho é desprezível, sendo a performance para $q = 3$ apenas 0,25 dB inferior à decisão branda não quantizada (CLARK JR; CAIN, 1981, p. 247).

O VA com entradas brandas (*soft input*) é similar ao VA com entradas binárias. A diferença está no cálculo das métricas de ramo, que agora utiliza a distância Euclidiana quadrática entre um soft bit recebido v_i no instante i , e uma saída y_i de um codificador hipotético que gera a transição de estado na treliça, onde y_i pode ser igual a um dos dois valores extremos que v_i pode assumir: z_0 (bit 0 muito provável) ou z_1 (bit 1 muito provável). A métrica de ramo w baseada na distância Euclidiana quadrática d^2 , para um conjunto de n soft bits recebidos, é calculada por:

$$w = d^2 = \sum_{i=0}^{n-1} (v_i - y_i)^2$$

Por exemplo, considere um demodulador que entrega no instante i um soft bit $v_i \in \mathbb{Z}$ quantizado em $l = 8$ níveis na entrada de um decodificador convolucional (2, 1). Considere que $v_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ e que o menor inteiro deste conjunto, $v_i = 0$, representa a máxima certeza que o bit 0 foi transmitido, e que o maior inteiro, $v_i = 7$,

Figura 41: Comparação do desempenho do VA para diferentes níveis de quantização da entrada



Fonte: o autor

representa a máxima certeza que o bit 1 foi transmitido. Portanto, $z_0 = 0$ e $z_1 = 7$.

Considere a sequência \mathbf{v} de $n = 2$ soft bits consecutivos recebidos: $\mathbf{v} = [v_{i-1}, v_i] = [6, 2]$. No VA por decisões rígidas, uma transição de estados na treliça é representada por 2 bits tais como $[1, 0]$, que são equivalentes a $\mathbf{y} = [z_1, z_0] = [7, 0]$ utilizando decisões brandas. Com os dados deste exemplo, a métrica de ramo w é calculada da seguinte forma:

$$w = \sum_{j=0}^{n-1} (v_j - y_j)^2 = (6 - 7)^2 + (2 - 0)^2 = 1 + 4 = 5$$

O VA provê a decodificação ótima (ML), mas o seu uso é normalmente restrito a códigos com comprimento de restrição $K \leq 10$, pois sua complexidade é proporcional a 2^K . Um exceção notável foi o *Grande Decodificador Viterbi* (*Big Viterbi Decoder - BVD*) construído para ser usado na missão Galileo da NASA ao planeta Júpiter em 1989 (ABRANTES, 2010, p. 312), (ONYSZCHUK, 1991), (COSTELLO et al., 1998, p. 2537). O BVD foi construído para decodificar um código com $K = 15$, representável por uma treliça com 16384 estados.

A *decodificação sequencial* é outro algoritmo para a decodificação de códigos convolucionais, sendo mais adequado do que o VA para o caso $K > 10$, pois sua complexidade

é essencialmente independente de K (JOHANNESSON; ZIGANGIROV, 1999, p. 22). A decodificação sequencial não é ML, mas seu emprego associado a códigos com maior K (e conseqüentemente maior distância livre) pode ser interessante em algumas aplicações devido ao ganho de codificação, conforme as curvas de desempenho apresentadas por Proakis e Salehi (2008, p. 534).

Os códigos convolucionais, apesar de não gozarem de uma rica estrutura algébrica como a dos códigos de bloco cíclicos, são atrativos principalmente pela disponibilidade de um algoritmo de decodificação ótimo e pela relativa facilidade de implementação da decodificação com entradas brandas.

4 Sistema ISDB-T

4.1 Histórico

O sistema de televisão digital ISDB-T foi desenvolvido no Japão e suas primeiras transmissões comerciais ocorreram em 2003 naquele país. O Brasil, após extensiva comparação dos padrões vigentes ATSC, DVB-T e ISDB-T, optou por desenvolver o padrão chamado *ISDB-Tb* utilizando como base o padrão japonês (PRESIDÊNCIA DA REPÚBLICA, 2006). Hoje este padrão também é conhecido como ISDB-T International ou SBTVD-T, e foi adotado por quase toda a América do Sul e por alguns países da América Central e da Ásia.

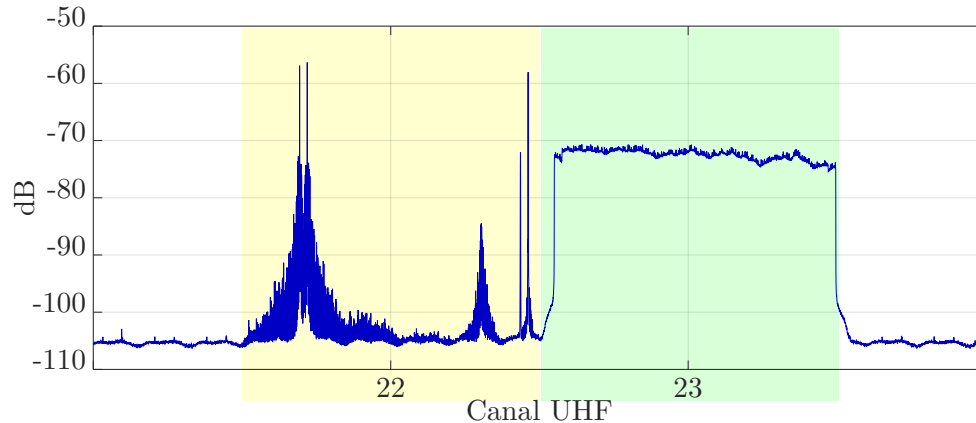
O Brasil iniciou suas primeiras transmissões de TV digital em 2007 na cidade de São Paulo. Começava ali uma fase de transição, pois mesmo onde o sinal digital chegava, o sinal analógico antigo era mantido. Em 15/02/2016, o município de Rio Verde-GO foi o primeiro escolhido para desligar seus transmissores analógicos, encerrando a fase de transição naquele local. O cronograma criado pelo governo (MINISTÉRIO DAS COMUNICAÇÕES, 2016) estabelece que em 05/12/2018 as últimas cidades do Brasil encerrarão suas transmissões analógicas, pondo um fim definitivo na TV analógica terrestre no Brasil.

As principais diferenças entre o ISDB-T e o ISDB-Tb estão no uso de CODECs mais avançados pelo último, na máscara de transmissão de RF, e no protocolo de interatividade. Como estas diferenças não afetam o receptor descrito neste trabalho, utilizaremos unicamente o termo ISDB-T para se referir ao padrão de TV digital nipônico-brasileiro.

Para efeito de comparação entre o ISDB-T e o sinal analógico PAL-M, a Figura 42 apresenta o espectro de dois canais de TV adjacentes¹, o da esquerda (canal 22) é um canal analógico, onde é fácil observar os picos referentes à luminância, crominância e o áudio. O canal da direita (canal 23) é um canal digital ISDB-T. Seu espectro é plano devido ao uso do OFDM.

¹ Canais de Uberlândia. A captura do espectro foi feita em novembro de 2017.

Figura 42: Espectro de dois canais adjacentes, sendo um analógico e o outro digital



Fonte: o autor

4.2 Segmentação do canal

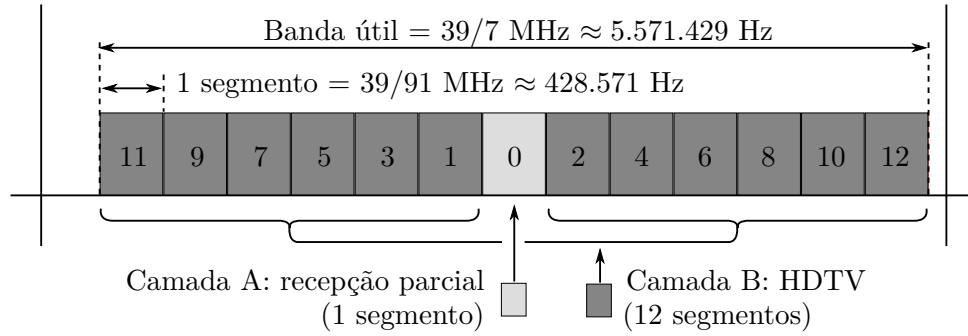
O ISDB-T é baseado no BST-OFDM (*Band Segmented Transmission Orthogonal Frequency Division Multiplexing*). Esta tecnologia divide o banda útil de TV (5,57 MHz no Brasil) em 13 segmentos de largura de faixa $BW_s \approx 428,571$ KHz.

Cada segmento comporta s_c subportadoras multiplexadas através da *Orthogonal frequency-division multiplexing* – OFDM. O número de subportadoras por segmento depende do modo de transmissão escolhido, e pode ser $s_c = 108$ para o modo 1, $s_c = 216$ para o modo 2 e $s_c = 432$ para o modo 3. O número total de subportadoras OFDM é $c = 13s_c + 1$, pois há uma portadora piloto extra do lado direito do segmento 12, tornando c um número ímpar e permitindo que o centro do espectro coincida com a subportadora OFDM central.

Os segmentos podem ser reunidos em até 3 grupos, conhecidos como *camadas hierárquicas* A, B e C. Cada camada hierárquica pode ser configurada com parâmetros diferentes de modulação e codificação de canal. Isto permite otimizar cada camada para um determinado tipo de serviço.

A configuração mais utilizada pelas emissoras é a de duas camadas hierárquicas: Na primeira camada (A), composta somente pelo segmento central de número zero, é transmitido um sinal com taxa de bits mais baixa, utilizando modulação e codificação robustas, destinado aos receptores portáteis (como telefones celulares); Na segunda camada (B), são utilizados os 12 segmentos restantes (numerados de 1 a 12), onde é transmitido um outro sinal de alta definição (com taxa de bits muito mais elevada) destinado aos receptores fixos de TV. Este exemplo de arranjo é ilustrado na Figura 43.

Figura 43: Arranjo dos segmentos e exemplo de configuração para duas camadas hierárquicas



Fonte: o autor

4.3 Recepção parcial

O segmento central de número zero pode ser usado pelas emissoras para transmitir um sinal dedicado para dispositivos portáteis. Assim o receptor de um smartphone, por exemplo, precisa amostrar e processar apenas um segmento para ter o sinal da emissora, simplificando o hardware e reduzindo o consumo de bateria. Este tipo de recepção é denominada recepção parcial, e o receptor é conhecido como *1-seg* ou *one-seg*. Já os receptores fixos, capazes de demodular todos os 13 segmentos são conhecidos como *full-seg*.

O sinal de vídeo na recepção parcial possui geralmente a resolução de 320x240 pixels e taxa de quadros 29,97 Hz, com compressão H.264. O áudio é comprimido através do AAC (*Advanced Audio Coding*). A taxa de bits típica na saída do receptor 1-seg é de cerca de 440 kbit/s.

4.4 Canalização

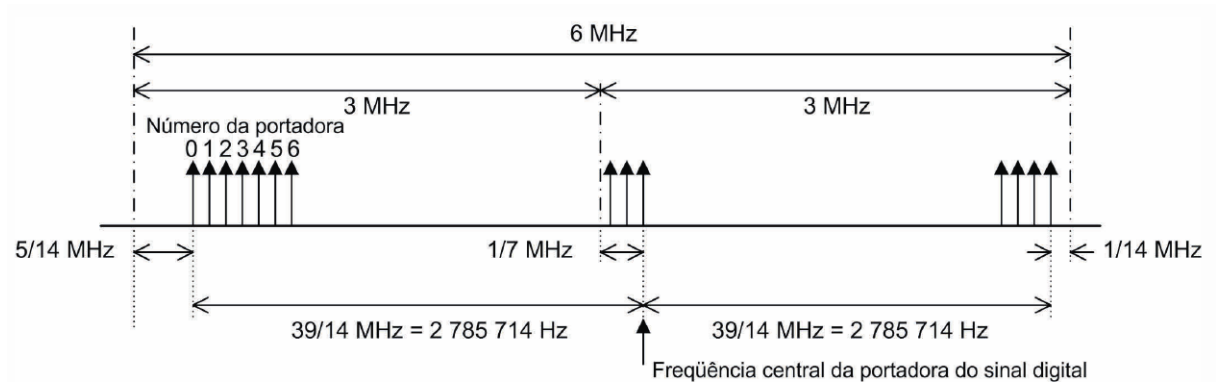
No Brasil foram designados os canais da faixa de VHF alto e UHF para o ISDB-T. A numeração e a largura de faixa de cada canal – 6 MHz – é a mesma da TV analógica. A Figura 44 apresenta o posicionamento das portadoras OFDM do sinal digital dentro do canal de televisão de 6 MHz. No extremo inferior do canal há uma faixa de guarda de $5/14$ MHz e no extremo superior esta faixa é de $1/14$ MHz. A frequência da portadora central, f_c , está deslocada em $1/7$ MHz do centro do canal.

Na faixa VHF alto, os canais são numerados de 7 a 13, e a frequência da portadora central, f_c , do sinal ISDB-T do canal de número C é dada por:

$$f_c = (C - 7) \cdot 6 \times 10^6 + 174 \times 10^6 + 22/7 \times 10^6$$

Já na faixa UHF, os canais são numerados de 14 a 69, e a f_c é calculada por:

Figura 44: Posicionamento da portadoras OFDM do ISDB-T dentro de um canal de televisão



Fonte: ABNT (2007, p. 52)

$$f_c = (C - 14) \cdot 6 \times 10^6 + 470 \times 10^6 + 22/7 \times 10^6$$

A faixa UHF possui algumas restrições. O canal 37 (608 a 614 MHz) não é utilizado para televisão, pois é atribuído internacionalmente ao serviço de radioastronomia, em caráter primário. Além disso, atualmente a faixa de UHF para a televisão foi reduzida, se restringindo aos canais de 14 a 51, pois a faixa de 698 a 806 MHz foi atribuída ao serviço móvel (para o 4G, chamada de *faixa de 700 MHz*), conforme resolução ANATEL nº 625, de 11 de novembro de 2013 (MINISTÉRIO DAS COMUNICAÇÕES, 2013). Portanto há no total 44 canais possíveis para a TV digital no Brasil.

4.5 Parâmetros de transmissão

O ISDB-T é um padrão muito flexível, permitindo que a emissora configure diversos parâmetros de transmissão para se adequar às características do canal e aos parâmetros das fontes de informação (áudio e vídeo) a serem transmitidos. A Tabela 11 apresenta todos os parâmetros configuráveis do ISDB-T, as constantes do sistema, as equações para o cálculo de parâmetros derivados, e a simbologia utilizada neste capítulo. A Tabela 12 apresenta os parâmetros já calculados para todos os casos. Conforme os parâmetros escolhidos, a taxa de bits por segmento pode variar de 280,85 a 1787,28 kbit/s (ou de 3651 a 23234 kbit/s, considerando todos os segmentos configurados em uma única camada).

O modo de transmissão e o intervalo de guarda são parâmetros globais, que se aplicam a todas as camadas, não sendo portanto passíveis de configuração independente por camada.

Conforme a Tabela 10, a escolha do modo de transmissão afeta, principalmente, dois parâmetros inversamente proporcionais: o espaçamento entre subportadoras (c_s) e

Tabela 10: Parâmetros OFDM para diferentes modos de transmissão

	# de portadoras por segmento (s_c)	# total de portadoras (c)	Tamanho da FFT (N)	Espaçamento entre subportadoras (c_s)	Duração útil do símbolo (T_u)
Modo 1	108	1405	2048	3968 Hz	252 μs
Modo 2	216	2809	4096	1984 Hz	504 μs
Modo 3	432	5617	8192	992 Hz	1008 μs

Fonte: o autor

a duração útil do símbolo (T_u). Em regiões onde a propagação é caracterizada por um grande espalhamento do atraso (σ_τ) devido ao efeito do multipercurso, é interessante que a duração dos símbolos, T_u , seja a maior possível para evitar a interferência intersimbólica (modo 3). Nos casos onde deve haver maior tolerância à desvios (erros) de frequência no receptor (por exemplo, na recepção em veículos de alta velocidade, devido ao efeito Doppler), é recomendado um maior espaçamento entre subportadoras c_s (modo 1) para evitar a interferência entre subportadoras.

O tamanho da FFT² é uma potência de 2 para permitir o uso de algoritmos eficientes no seu cálculo. Assim, a menor FFT que comporta as c portadoras utilizadas pode ser calculado por³:

$$N = 2^{\lceil \log_2 c \rceil}$$

As $N - c$ portadoras restantes desocupadas são chamadas de *portadoras virtuais* ou *portadoras nulas*. Elas geram faixas de guarda na extremidade do espectro transmitido, importantes para reduzir a interferência entre canais adjacentes.

Como a duração útil de símbolo, T_u , varia proporcionalmente com N conforme o modo escolhido, a taxa de amostragem f_s da IFFT do transmissor (ou da FFT do receptor *full-seg*) é constante, qualquer que seja o modo de transmissão escolhido:

$$f_s = \frac{N}{T_u} = \frac{2048}{252} = \frac{4096}{504} = \frac{8192}{1008} \approx 8,126984 \text{ MHz}$$

A duração total do símbolo é a soma da duração útil (que carrega informação) com a duração do intervalo de guarda $T_s = T_u + T_g$. O duração do intervalo de guarda, T_g é uma fração configurável da duração útil do símbolo, e pode ser $T_u/4$, $T_u/8$, $T_u/16$ ou $T_u/32$.

² O termo *tamanho da FFT* refere-se ao tamanho da FFT utilizada no receptor *full-seg*, e também ao tamanho da IFFT utilizada no transmissor

³ O operador matemático $\lceil x \rceil$ denota a operação *teto*, o menor inteiro maior ou igual a x

Tabela 11: Cálculos dos parâmetros do ISDB-T

Item	Símbolo	Valor
<i>Parâmetros configuráveis:</i>		
Modo	M	1, 2 ou 3
Intervalo de guarda	IG	1/4, 1/8, 1/16 ou 1/32
Profundidade do intercalador de tempo (ver Tabela 14)	—	0; 95,76 ms; 191,52 ms ou 383,04 ms
Modulação	—	DQPSK, QPSK, 16-QAM ou 64-QAM
Núm. de segmentos da camada A	N_{sa}	1 até 13
Núm. de segmentos da camada B	N_{sb}	0 até 13
Núm. de segmentos da camada C	N_{sc}	0 até 13
Taxa do cod. convolucional	R	1/2, 2/3, 3/4, 5/6 ou 7/8
<i>Constantes:</i>		
Núm. de segmentos	N_s	$N_{sa} + N_{sb} + N_{sc} = 13$
Símbolos por quadro	N_q	204
Frequência de amostragem da FFT	f_s	$2048/252 \approx 8,126984$ MHz
Largura de faixa por segmento	BW_s	$f_s \cdot (108/2048) = 6 \times 10^6/14 \approx 428,571$ kHz
<i>Parâmetros calculados:</i>		
Tamanho da FFT (13 segmentos)	N	$2048 \cdot 2^{M-1}$
Núm. de portadoras (13 segmentos)	c	$13s_c + 1 = (1404 \cdot 2^{M-1}) + 1$
Núm. de portadoras por segmento	s_c	$108 \cdot 2^{M-1}$
Núm. de portadoras de dados por segmento	n_c	$(8/9) \cdot s_c$
Espaçamento entre subportadoras	c_s	$BW_s/s_c = f_s/N$
Largura de faixa total	BW	$BW_s \cdot N_s + c_s$
Duração útil de símbolo	T_u	N/f_s
Duração do intervalo de guarda	T_g	$T_u \cdot IG$
Duração total do símbolo	T_s	$T_u + T_g$
Duração do quadro	T_q	$N_q \cdot T_s$
Núm. de bits por símbolo modulado	b	2 para DQPSK e QPSK; 4 para 16QAM; 6 para 64QAM
Taxa líquida de transmissão por segmento (bits/s)	B	$(b \cdot R \cdot n_c \cdot (188/204))/T_s$

Fonte: o autor

Tabela 12: Parâmetros de transmissão do ISDB-T

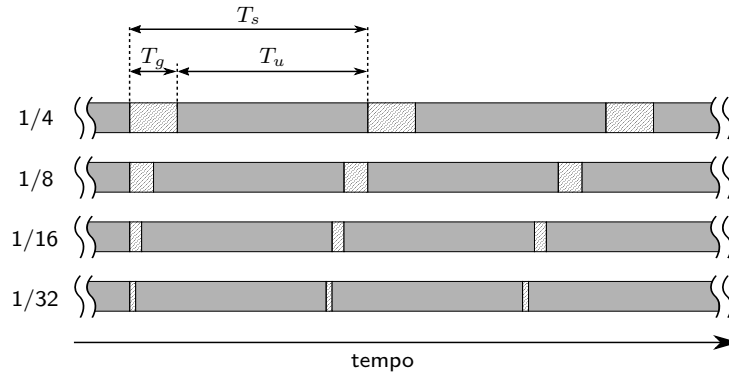
		Modo 1		Modo 2		Modo 3	
Largura de faixa por segmento (BW_s)		6000/14 \approx 428,571 kHz					
Espaçamento entre sub-portadoras (c_s)		$BW_s/108$ \approx 3,968 kHz		$BW_s/216$ \approx 1,984 kHz		$BW_s/432$ \approx 0,992 kHz	
# de portadoras por segmento	Total (s_c)	108	108	216	216	432	432
	Dados (n_c)	96	96	192	192	384	384
	SP ^a	9	0	18	0	36	0
	CP ^b	0	1	0	1	0	1
	TMCC ^c	1	5	2	10	4	20
	AC1 ^d	2	2	4	4	8	8
	AC2 ^d	0	4	0	9	0	19
Modulação		QPSK 16QAM 64QAM	DQPSK	QPSK 16QAM 64QAM	DQPSK	QPSK 16QAM 64QAM	DQPSK
Símbolos por quadro (N_q)		204					
Duração útil de símbolo (T_u)		252 μ s		504 μ s		1008 μ s	
Duração do intervalo de guarda (T_g)		63 μ s (1/4) 31,5 μ s (1/8) 15,75 μ s (1/16) 7,875 μ s (1/32)		126 μ s (1/4) 63 μ s (1/8) 31,5 μ s (1/16) 15,75 μ s (1/32)		252 μ s (1/4) 126 μ s (1/8) 63 μ s (1/16) 31,5 μ s (1/32)	
Duração total do símbolo ($T_s = T_u + T_g$)		315 μ s (1/4) 283,5 μ s (1/8) 267,75 μ s (1/16) 259,875 μ s (1/32)		630 μ s (1/4) 567 μ s (1/8) 535,5 μ s (1/16) 519,75 μ s (1/32)		1260 μ s (1/4) 1134 μ s (1/8) 1071 μ s (1/16) 1039,5 μ s (1/32)	
Duração do quadro ($T_q = N_q \cdot T_s$)		64,26 ms (1/4) 57,834 ms (1/8) 54,621 ms (1/16) 53,0145 ms (1/32)		128,52 ms (1/4) 115,668 ms (1/8) 109,242 ms (1/16) 106,029 ms (1/32)		257,04 ms (1/4) 231,336 ms (1/8) 218,484 ms (1/16) 212,058 ms (1/32)	
Frequência de amostragem da FFT (f_s)		2048/252 \approx 8,126984 MHz					
Código interno		Código convolucional, taxas 1/2, 2/3, 3/4, 5/6 ou 7/8					
Código externo		Código Reed-Solomon RS(204,188)					

Fonte: ARIB (2014, p. 13)

^a SP: *Scattered Pilot* - pilotos espalhados - utilizado para equalização no receptor^b CP: *Continual Pilot* - pilotos contínuo - utilizado para equalização no receptor^c TMCC: *Transmission and Multiplexing Configuration Control* - contém os parâmetros de transmissão^d AC1 e AC2: *Auxiliary Channel* - canal auxiliar, utilizado para informação adicional para controle do sinal de transmissão

O intervalo de guarda é descartado no receptor, evitando a ocorrência de interferência intersimbólica causada pela propagação multipercurso desde que o espalhamento de atraso seja inferior à sua duração. Quanto maior T_g , menor a quantidade de informação transmitida por unidade de tempo. A Figura 45 ilustra as diferentes durações do intervalo de guarda.

Figura 45: Opções de intervalo de guarda



Fonte: o autor

Os parâmetros *profundidade do intercalador de tempo*, *modulação* e *taxa do codificador convolucional* podem ser configurados individualmente para cada camada.

A escolha da modulação e da taxa do codificador convolucional ditam a robustez do sinal frente ao ruído em troca de menores taxas de transmissão. Já a opção de uma maior profundidade do intercalador de tempo deixa o sistema menos susceptível aos ruídos do tipo impulsivo, em troca de uma maior latência na comunicação.

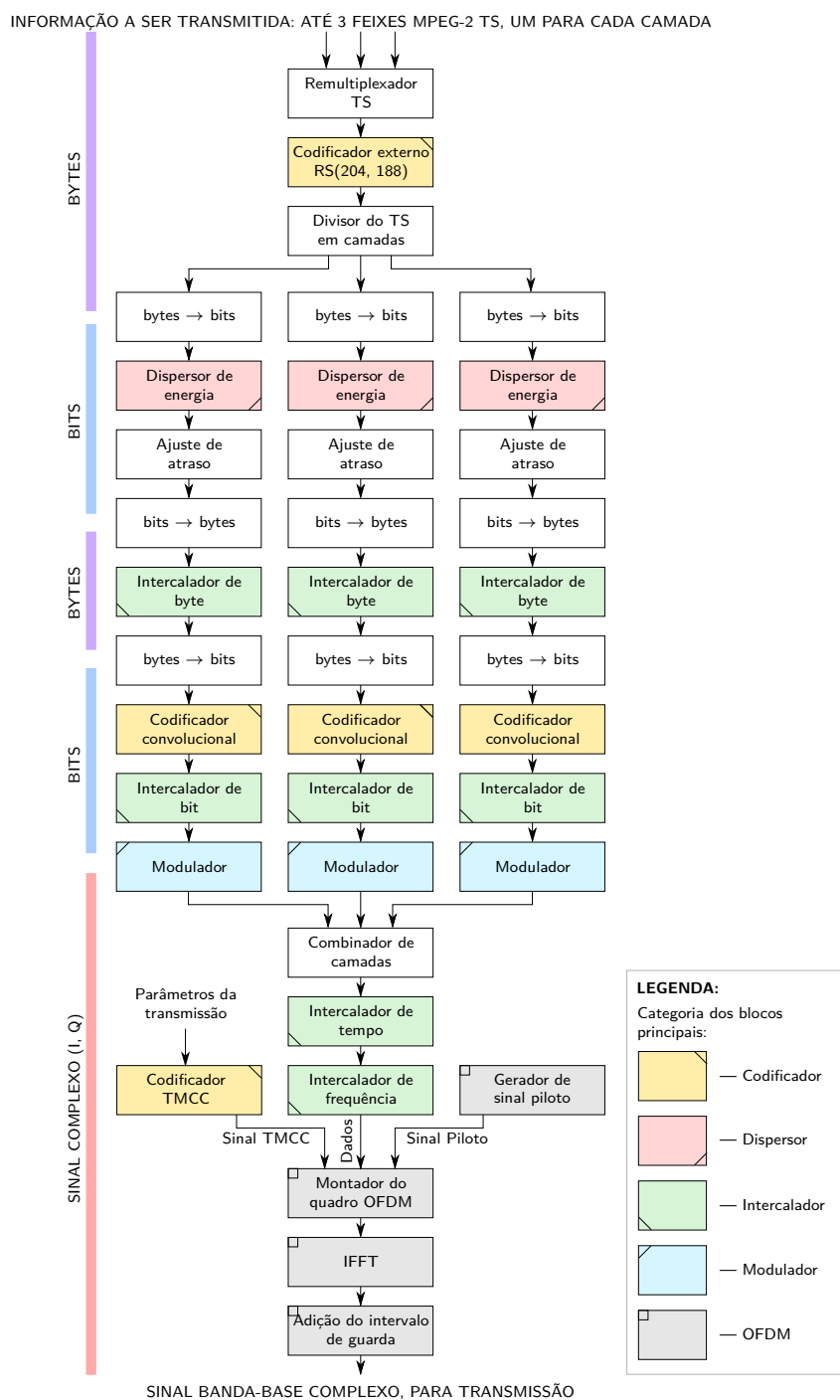
4.6 Transmissão ISDB-T

O sistema de transmissão ISDB-T é padronizado pelas normas ARIB (2014) e ABNT (2007) que estabelecem os requisitos do sistema e apresentam o processo de codificação de canal e modulação. A Figura 46 apresenta estes processos através de um diagrama de blocos.

O transmissor pode ser configurado para até 3 camadas, e cada uma delas requer um feixe MPEG-2 TS distinto. O MPEG-2 TS é padronizado pela norma ISO/IEC-13818-1:2000 (ISO/IEC, 2000), e transporta sinais de áudio e vídeo comprimidos, legendas, guia eletrônico de programação, dentre outras informações divididas em pacotes de 188 bytes.

O sinal de radiofrequência é gerado a partir de diversas etapas (blocos) de processamento de sinal. Os principais blocos serão apresentados nas seções seguintes.

Figura 46: Diagrama de blocos do transmissor ISDB-T



Fonte: ARIB (2014, p. 18)

4.6.1 Codificador externo RS(204, 188)

O codificador externo é do tipo Reed-Solomon, onde são gerados 16 bytes de paridade para cada pacote MPEG-2 TS de 188 bytes, compondo um pacote *TSP* de 204 bytes de comprimento. No receptor, este código é capaz de corrigir até 8 bytes corrompidos dentro de um pacote TSP. Os detalhes do código Reed-Solomon utilizados no ISDB-T são descritos na subseção 3.1.5.

4.6.2 Dispersor de energia

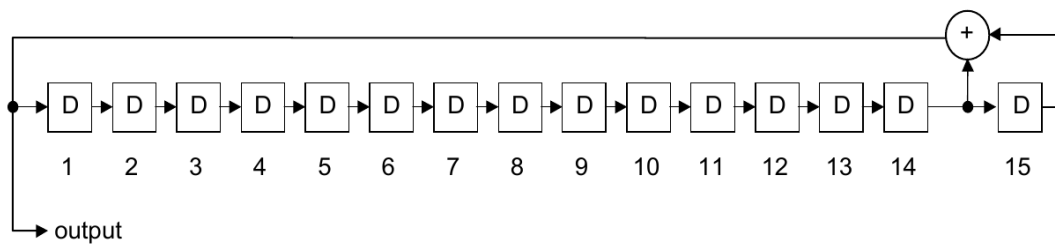
Este bloco faz a operação lógica *OU exclusivo* bit a bit do sinal de entrada com os bits emitidos por um gerador de sequência binária pseudo-aleatória (Pseudo-Random Binary Sequence — PRBS). O objetivo deste bloco é tornar os bits 0 e 1 equiprováveis em sua saída, forma adequada para processamento posterior pelo codificador convolucional.

O PRBS é caracterizado pela Equação 4.1. A Figura 47 apresenta o gerador PRBS implementado através de registradores de deslocamento.

$$g(x) = X^{15} + X^{14} + 1 \quad (4.1)$$

Os registradores de deslocamento do gerador PRBS são reiniciados a cada quadro OFDM com o conteúdo 100101010000000. Os bytes de sincronismo dos pacotes MPEG-2 TS (hexadecimal: 0x47h / binário: 0100 0111 / decimal: 71) não são submetidos a esta operação.

Figura 47: Gerador PRBS utilizado no dispersor de energia

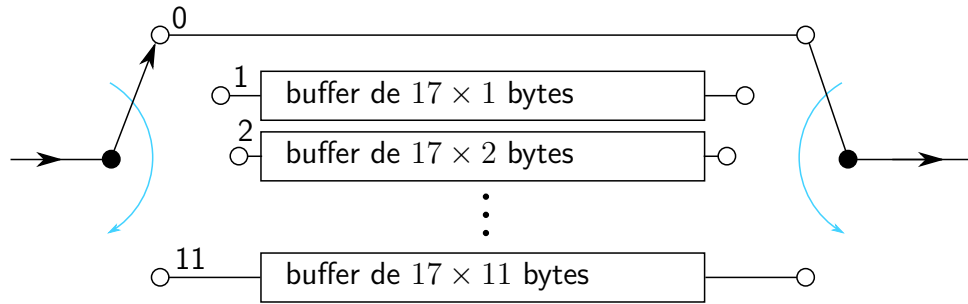


Fonte: ARIB (2014, p. 24)

4.6.3 Intercalador de byte

A Figura 48 apresenta o circuito do intercalador de byte, composto por 12 buffers FIFO de comprimentos crescentes (0 a 187) e de chaveadores sincronizados na entrada e na saída que comutam a cada byte na entrada de forma cíclica, iniciando pela posição 0 no processamento do byte de sincronismo (primeiro byte) de cada pacote MPEG-2 TS.

Figura 48: Funcionamento do intercalador de byte



Fonte: ARIB (2014, p. 26)

4.6.4 Codificador convolucional

O codificador interno é um codificador convolucional com comprimento de restrição $K = 7$ e taxa básica de $1/2$. Sua taxa pode ser configurada para outros valores: $2/3$, $3/4$, $5/6$ ou $7/8$, através do punctionamento. Os detalhes do código convolucional utilizado no ISDB-T são descritos na subseção 3.2.3.

4.6.5 Modulador e intercalador de bit

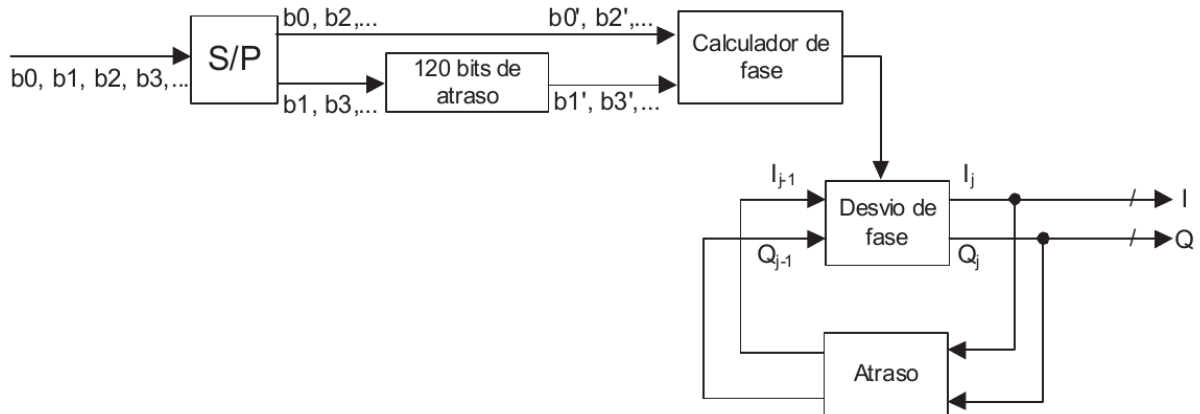
O ISDB-T permite, para cada camada hierárquica, a escolha entre 4 tipos de modulações disponíveis: DQPSK (modulação diferencial); QPSK, 16QAM ou 64QAM (modulações síncronas). As próximas seções explicam cada um destes tipos de modulação empregados no ISDB-T e seus intercaladores de bit associados.

4.6.5.1 DQPSK

A modulação DQPSK é uma modulação diferencial que não requer do receptor a recuperação da fase absoluta para a demodulação. O DQPSK mapeia dois bits em cada transição da constelação, e consiste dos blocos apresentados na Figura 49. O bloco *calculador de fase* gera em sua saída as fases estipuladas pela Tabela 13. Como o mapeamento é deslocado em $\pi/4$, a diferença de fase na saída do modulador é de no máximo $3\pi/4 = 135^\circ$, evitando o cruzamento na origem da constelação e reduzindo as variações de amplitude no envelope do sinal gerado.

Os bits de um dos ramos da saída do conversor série-paralelo (S/P) passam através de uma linha de atraso de 120 bits, implementada através de um *buffer* do tipo FIFO. Esta construção provoca o intercalamento entre os bits modulados. A constelação do sinal complexo (I/Q) gerado é mostrada na Figura 50.

Figura 49: Diagrama do modulador DQPSK



Fonte: ABNT (2007, p. 22)

Tabela 13: Calculador de fase DQPSK

Entrada $b_0' b_1'$	Saída θ_j
0 0	$\pi/4$
0 1	$-\pi/4$
1 0	$3\pi/4$
1 1	$-3\pi/4$

Fonte: ABNT (2007, p. 23)

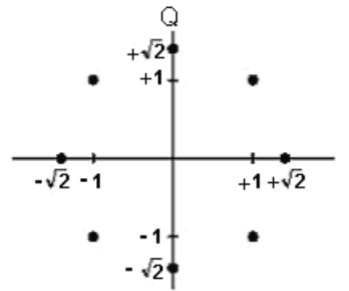
4.6.5.2 QPSK

A modulação QPSK faz o mapeamento de 2 bits por vez na constelação, e consiste dos blocos apresentados na Figura 51. Assim como na modulação DQPSK, os bits de um dos ramos da saída do conversor série-paralelo (S/P) passam através de uma linha de atraso de 120 bits, que provoca o intercalamento entre os bits modulados. O mapeamento dos bits na constelação do sinal complexo (I/Q) é mostrada na Figura 52.

4.6.5.3 16QAM

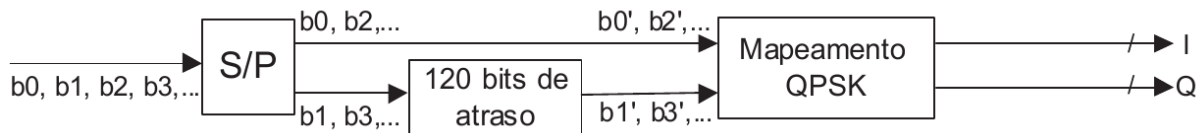
A modulação 16QAM é executada pelos blocos apresentados na Figura 53 e mapeia 4 bits por vez na constelação. Os bits de três dos quatro ramos de saída do conversor série-paralelo (S/P) passam através de três linhas de atraso de 40, 80 e 120 bits, que provocam o intercalamento entre os bits modulados. O mapeamento dos bits na constelação do sinal complexo (I/Q) é mostrada na Figura 54.

Figura 50: Constelação DQPSK



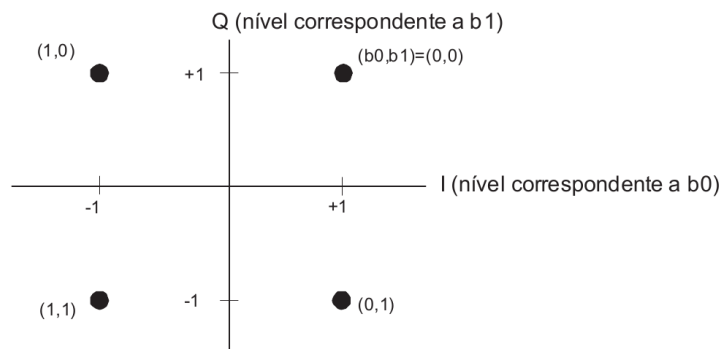
Fonte: ABNT (2007, p. 22)

Figura 51: Diagrama do modulador QPSK



Fonte: ABNT (2007, p. 23)

Figura 52: Mapeamento dos bits na constelação QPSK

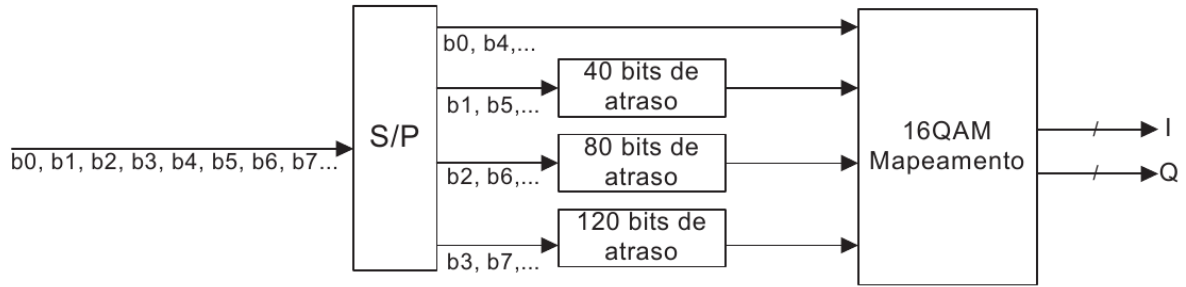


Fonte: ABNT (2007, p. 23)

4.6.5.4 64QAM

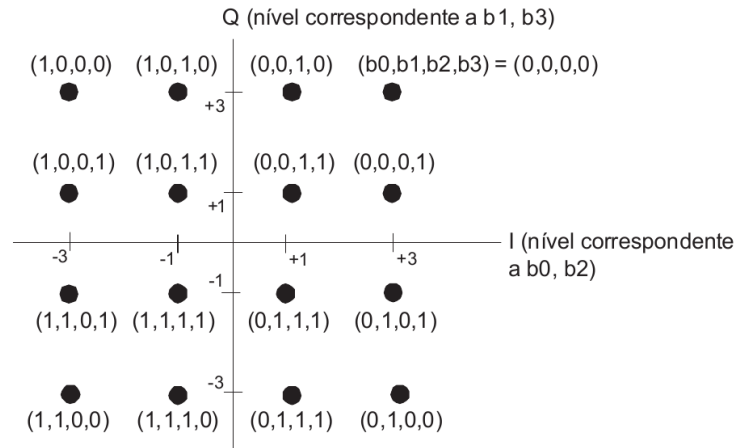
A modulação 64QAM mapeia 6 bits por vez na constelação I/Q. Ela é executada pelos blocos apresentados na Figura 55. Os bits de cinco dos seis ramos de saída do conversor série-paralelo (S/P) passam através das cinco linhas de atraso de 24, 48, 72, 96 e 120 bits, que provocam o intercalamento entre os bits modulados. O mapeamento dos bits na constelação do sinal complexo (I/Q) é mostrada na Figura 56.

Figura 53: Diagrama do modulador 16QAM



Fonte: ABNT (2007, p. 24)

Figura 54: Mapeamento dos bits na constelação 16QAM



Fonte: ABNT (2007, p. 24)

4.6.6 Intercalador de tempo

A Figura 57 apresenta o circuito do intercalador de tempo, composto por n_c buffers FIFO de comprimentos variáveis e de chaveadores sincronizados na entrada e na saída. O comprimento L_i do i -ésimo buffer é dado pela expressão:

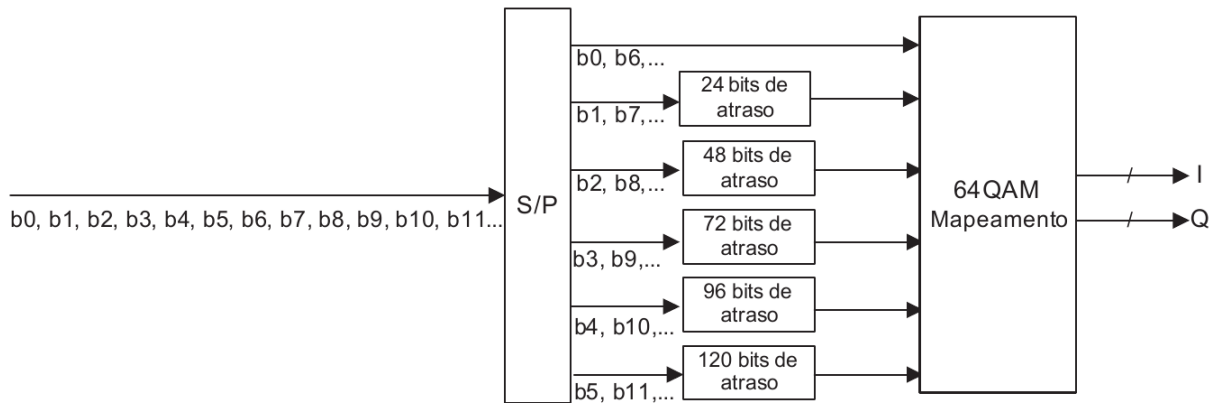
$$L_i = I \cdot m_i, \quad 0 \leq i \leq n_c - 1$$

onde

$$m_i = i \cdot 5 \mod 96, \quad 0 \leq i \leq n_c - 1$$

A constante I é um parâmetro de transmissão configurável, que pode assumir um dos quatro valores da Tabela 14 para um determinado modo de transmissão. O valor $\max(L_i)$ da tabela é o comprimento do maior buffer para um determinado I . A Figura 58

Figura 55: Diagrama do modulador 64QAM



Fonte: ABNT (2007, p. 24)

apresenta o atraso dos símbolos ao longo do tempo para um sistema no modo 3 ($n_c = 384$) e $I = 4$.

Tabela 14: Parâmetros para o intercalador de tempo

Opção	Modo 1		Modo 2		Modo 3	
	I	$\max(L_i)$	I	$\max(L_i)$	I	$\max(L_i)$
A: Desabilitado (0 ms)	0	0	0	0	0	0
B: 95,76 ms	4	380	2	190	1	95
C: 191,52 ms	8	760	4	380	2	190
D: 383,04 ms	16	1520	8	760	4	380

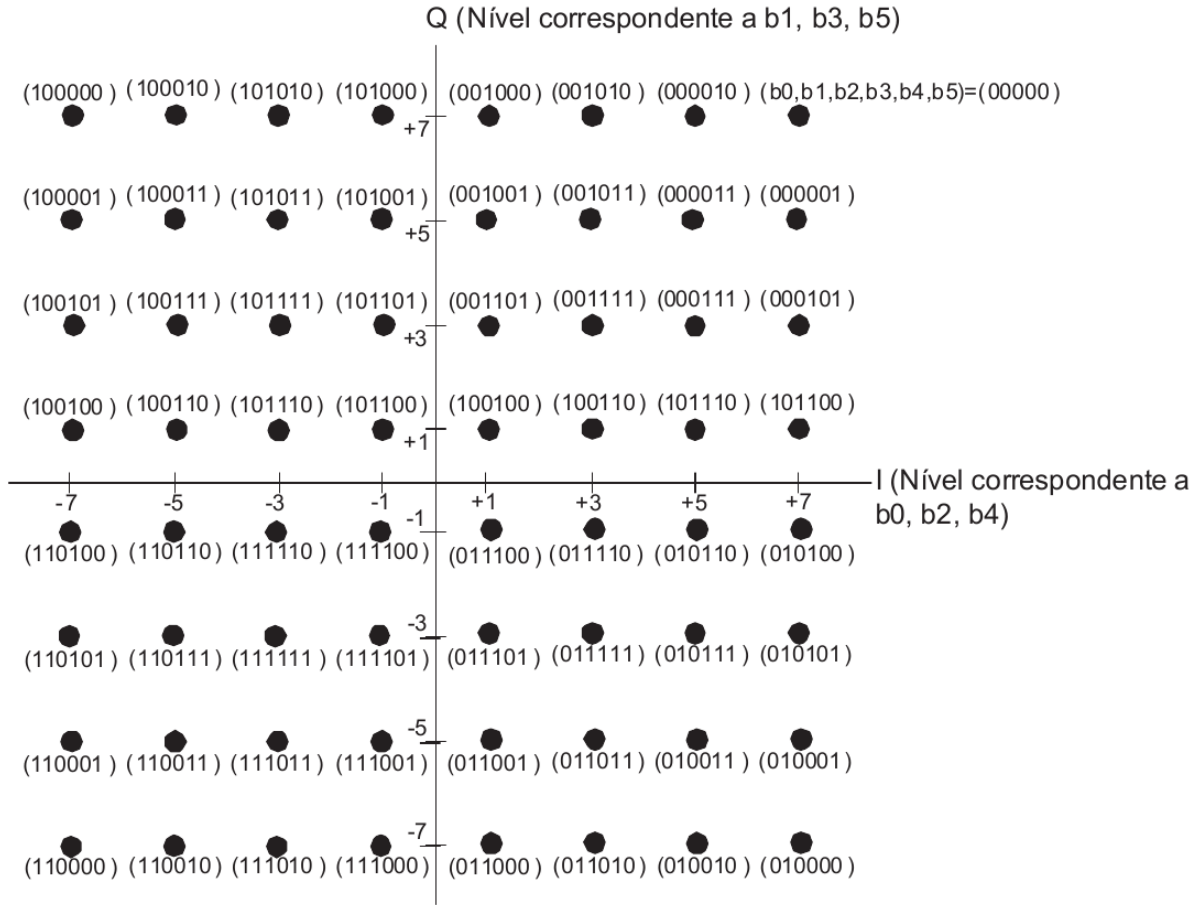
Fonte: o autor

4.6.7 Intercalador de frequência

O intercalamento de frequência pode ocorrer em até 3 etapas, dependendo do tipo da camada hierárquica, conforme a Figura 59. As etapas são: intercalamento entre diferentes segmentos (não executada na camada de recepção parcial); rotação de portadoras dentro do segmento; e aleatorização de portadoras dentro do segmento.

Caso haja camadas com diferentes categorias de modulação (diferencial e síncrona), o intercalamento de frequência deve ser executado separadamente entre grupos de segmentos das camadas que possuam a mesma categoria de modulação.

Figura 56: Mapeamento dos bits na constelação 64QAM



Fonte: ABNT (2007, p. 25)

4.6.7.1 Intercalamento entre diferentes segmentos

As portadoras de diferentes segmentos⁴ que possuam a mesma categoria de modulação são intercaladas entre si, através do seguinte mapeamento:

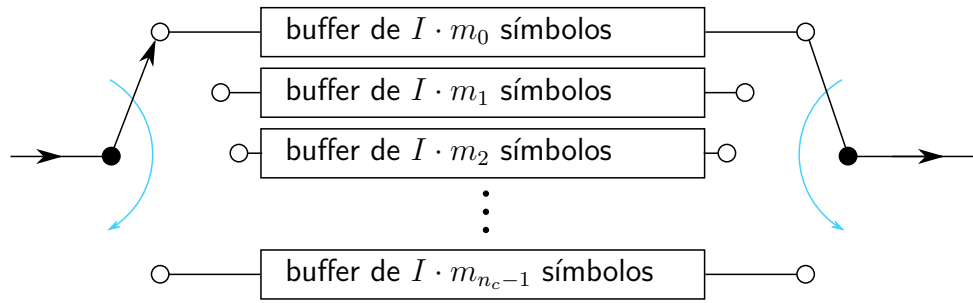
$$S_x \rightarrow S_y \quad \text{onde} \quad 0 \leq x \leq n \cdot n_c - 1 \quad \text{e} \quad y = x \cdot n \bmod (n \cdot n_c) + \left\lfloor \frac{x}{n_c} \right\rfloor$$

Onde n_c é o número de portadoras de dados por segmento (ver Tabela 12) e n é o número de segmentos da mesma categoria de modulação.

Os subscritos de S , x e y , são os índices dos símbolos quando ordenados pelo número da portadora e depois pelo número do segmento, ou seja $x = i_1 + k_1 \cdot n_c$ e $y = i_2 + k_2 \cdot n_c$, onde i_1 e i_2 são os índices das portadoras antes e depois do mapeamento, respectivamente; e k_1 e k_2 são os índices dos segmentos antes e depois do mapeamento, respectivamente.

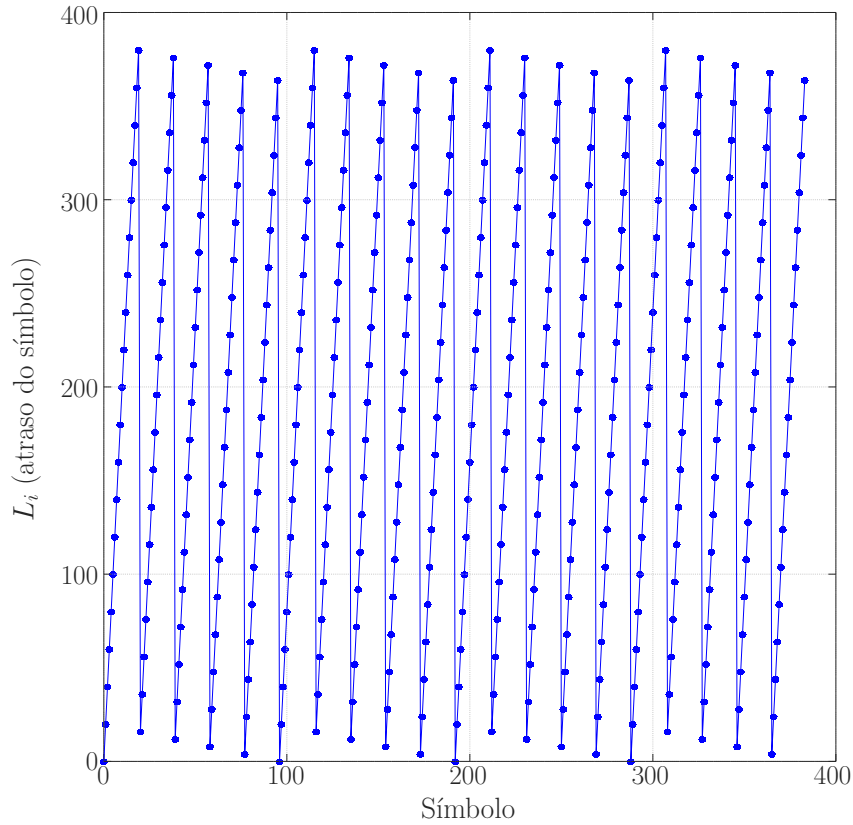
⁴ Estas portadoras podem pertencer a diferentes camadas, desconsiderando a de recepção parcial

Figura 57: Funcionamento do intercalador de tempo



Fonte: ARIB (2014, p. 36)

Figura 58: Espalhamento do atraso de símbolo provocado pelo intercalador de tempo



Fonte: o autor

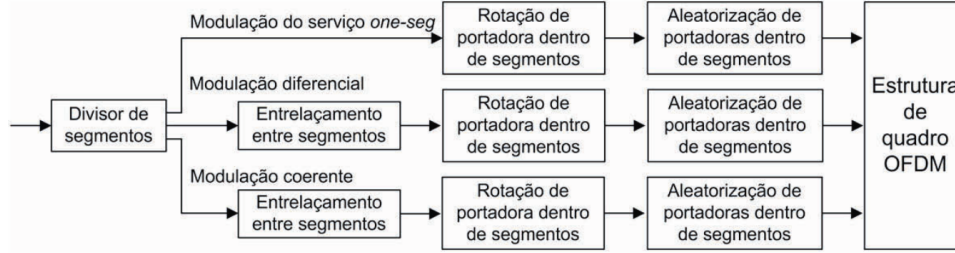
4.6.7.2 Rotação de portadoras dentro do segmento

A rotação entre as portadoras de um segmento é feita através do mapeamento do símbolo de dados S_i da portadora de número i para a portadora de número j , S_j :

$$S_i \rightarrow S_j \quad \text{onde} \quad 0 \leq i \leq n_c - 1 \quad \text{e} \quad j = (k + i) \mod n_c$$

Onde k é o número do segmento e n_c é o número de portadoras de dados por

Figura 59: Etapas do intercalador de frequência



Fonte: ABNT (2007, p. 30)

segmento (ver Tabela 12). É possível observar que, para o segmento zero, $k = 0$ (que pode ser utilizado para a recepção parcial, quando configurado), esta operação não tem efeito, pois $i = j = i \bmod n_c$ quando $0 \leq i \leq n_c - 1$.

4.6.7.3 Aleatorização de portadoras dentro do segmento

Para cada segmento, é executada uma aleatorização das portadoras conforme as extensas tabelas apresentadas por ARIB (2014, p. 41) ou ABNT (2007, p. 33).

4.6.8 Sinal piloto

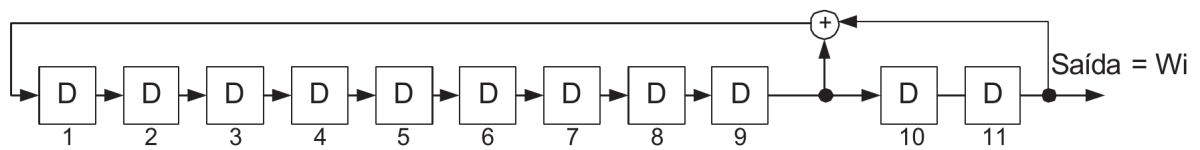
Para auxiliar o receptor na equalização, na transmissão são gerados diversos sinais pilotos modulados em BPSK que dependem do tipo da modulação escolhida para os dados. Na modulação diferencial (DQPSK) é utilizado o piloto contínuo (*Continual Pilot* - *CP*), onde a portadora que o transmite é fixa (portadora 0). Já para as modulações síncronas (demais modulações), é utilizado o piloto espalhado (*Scattered Pilot* - *SP*), que é transmitido a cada 12 portadoras em posições variáveis ao longo do tempo (mais detalhes na subseção 4.6.10).

O sinal piloto é sintetizado através do gerador de sequência binária pseudo-aleatória (Pseudo-Random Binary Sequence — PRBS) caracterizado pela Equação 4.2. A Figura 60 mostra o gerador PRBS implementado através de registradores de deslocamento, cuja saída W_i é modulada em BPSK. O índice i é o número da portadora do segmento OFDM.

$$g(x) = X^{11} + X^9 + 1 \quad (4.2)$$

Os registradores de deslocamento deste gerador PRBS são reiniciados a cada quadro OFDM com um valor que depende do número do segmento e do modo de transmissão (ABNT, 2007, p. 41). Para o segmento 0, por exemplo, temos os seguintes valores de inicialização: modo 1: 11001000010; modo 2: 01110001001; modo 3: 00100001011.

Figura 60: Gerador PRBS utilizado para a síntese do sinal piloto



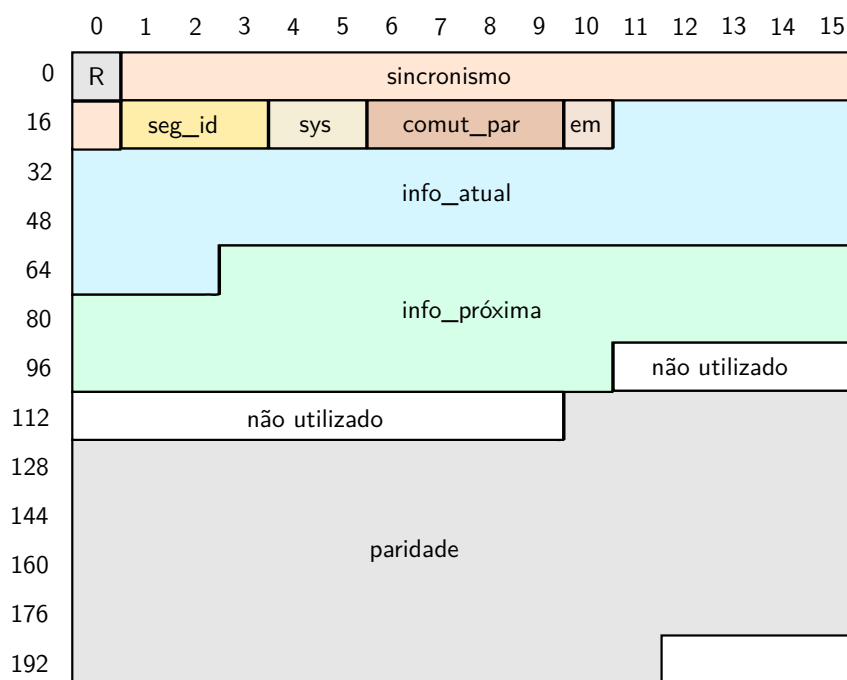
Fonte: ABNT (2007, p. 41)

O mapeamento BPSK de W_i para a constelação I/Q é: $W_i = 1 \rightarrow (-4/3, 0)$ e $W_i = 0 \rightarrow (+4/3, 0)$.

4.6.9 TMCC

Todos os parâmetros da transmissão, exceto o modo e o intervalo de guarda⁵ são transmitidos continuamente em cada quadro OFDM através do sinal TMCC (*Transmission and Multiplexing Configuration Control*) modulado em DBPSK.

Figura 61: Informações contidas no quadro TMCC



Fonte: o autor

O TMCC possui 204 bits, e comporta as informações representadas na Figura 61, que são:

- R: Referência para demodulação diferencial (DBPSK) do TMCC;

⁵ O receptor deve obter estes parâmetros não contidos no TMCC às cegas

- **sincronismo:** Alterna entre 0011010111101110 e seu inverso a cada quadro OFDM. Utilizado para que o receptor possa sincronizar o início do quadro OFDM;
- **seg_id:** Identificação do tipo de segmento (diferencial: 111; síncrono: 000);
- **sys:** Tipo de sistema, sempre 00 para o sistema de televisão terrestre ISDB-T atual;
- **comut_par:** Normalmente 1111. Quando diferente, indica um contador regressivo para o chaveamento de parâmetros, onde **info_atual** será igual a **info_próxima**;
- **em:** Se igual a 1 indica que a emissora pode controlar o receptor para que ele ligue ao receber um sinal de emergência;
- **info_atual** e **info_próxima:** parâmetros de transmissão atuais e próximos (para o caso de um chaveamento). Ver detalhamento destes bits a seguir;
- **não utilizado:** 0 para todos os bits;
- **paridade:** Paridade calculada para os bits 20 até 121 do TMCC, pelo codificador DSC(184, 102) do tipo *difference-set cyclic code* - DSC. O código é uma versão encurtada do DSC(273,191).

Os blocos de bits **info_atual** e **info_próxima** tem a mesma estrutura, e o conteúdo de cada um deles dentro do quadro TMCC é ilustrado na Figura 62, onde:

- **p:** 1 indica que o segmento de número zero é utilizado para a recepção parcial e que a camada A é destinada para esta finalidade;
- **A_mod, B_mod e C_mod:** Indica a modulação da respectiva camada — 000: DQPSK; 001: QPSK; 010: 16QAM; 011: 64QAM; 111: camada não utilizada; demais valores são reservados;
- **A_taxa_cod, B_taxa_cod e C_taxa_cod:** Indica a taxa utilizada pelo codificador convolucional na respectiva camada — 000: 1/2; 001: 2/3; 010: 3/4; 011: 5/6; 100: 7/8; 111: camada não utilizada; demais valores são reservados;
- **A_interc, B_interc e C_interc:** Profundidade do intercalador de tempo (ver Tabela 14) — 000: 0 (opção A); 001: 95,76 ms (opção B); 010: 191,52 ms (opção C); 011: 383,04 ms (opção D); 111: camada não utilizada; demais valores são reservados;
- **A_num_seg, B_num_seg e C_num_seg:** Número de segmentos utilizados na respectiva camada, representado através da conversão direta binário-decimal, onde 1100 indica, por exemplo, que a camada ocupa 12 segmentos; 1111: indica camada não utilizada; demais valores (incluindo o 0000) são reservados;

Figura 62: Informações contidas nos blocos `info_atual` e `info_proxima`

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	p	A_mod			A_taxa_cod			A_interc			A_num_seg			B_mod		
16	B_taxa_cod		B_interc		B_num_seg			C_mod								
32	C_interc		C_num_seg			C_taxa_cod										

Fonte: o autor

4.6.10 Quadro OFDM

O quadro OFDM possui duração de 204 símbolos OFDM. Sua função é a de multiplexar os símbolos de dados, pilotos, AC e TMCC e distribuí-los ao longo das portadoras do segmento. Há 6 tipos de quadros diferentes no ISDB-T, dependendo do modo escolhido e da categoria da modulação (diferencial: DQPSK; ou síncrona: QPSK, 16QAM, 64QAM). Além disso, a distribuição dos sinais AC e TMCC dependem do número do segmento.

As tabelas 15 e 16 apresentam a distribuição dos sinais de dados, pilotos, AC e TMCC ao longo das portadoras dentro de um quadro do segmento zero. As figuras 63 e 64 ilustram a composição do quadro no modo 1 para o caso da modulação diferencial e síncrona, respectivamente.

Na composição do quadro, qualquer que seja o modo de transmissão, os dados ocupam 8/9 dos símbolos. Portanto o *overhead* é de $1/9 \approx 11,1\%$.

4.6.11 IFFT

A IFFT (*Inverse Fast Fourier Transform* ou Transformada Rápida de Fourier Inversa) de tamanho N , converte a cada execução, um conjunto de c símbolos S_0, S_1, \dots, S_{c-1} do domínio da frequência para o domínio do tempo, obtendo assim um símbolo OFDM em banda base — o vetor $s_n = [s_0, s_1, \dots, s_{N-1}]$ — conforme a equação:

$$s_n = \mathcal{F}^{-1}(S_k) = \frac{1}{N} \sum_{k=0}^{N-1} S_k \cdot e^{j2\pi kn/N}$$

Os $N - c$ símbolos na entrada da IFFT, S_c até S_{N-1} , são *portadoras virtuais*, com valor igual a zero. A subseção 2.2.3 apresenta com maiores detalhes a execução da IFFT em um transmissor OFDM.

Tabela 15: Configuração das portadoras OFDM para o segmento zero na modulação diferencial

Sinal	Índice das portadoras
<i>Modo 1 — 108 portadoras:</i>	
CP	0
SP	Ausente
AC1	35; 79
AC2	3; 72; 85; 89
TMCC	49; 61; 96; 99; 104
DADOS	Portadoras restantes (96)
<i>Modo 2 — 216 portadoras:</i>	
CP	0
SP	Ausente
AC1	98; 101; 118; 136
AC2	10; 30; 55; 81; 108; 111; 153; 167; 185
TMCC	23; 37; 51; 68; 105; 121; 158; 178; 191; 195
DADOS	Portadoras restantes (192)
<i>Modo 3 — 432 portadoras:</i>	
CP	0
SP	Ausente
AC1	7; 89; 206; 209; 226; 244; 377; 407
AC2	25; 30; 42; 104; 108; 118; 138; 163; 189; 216; 219; 261; 275; 293; 324; 327; 339; 364; 382
TMCC	34; 48; 54; 70; 101; 131; 145; 159; 176; 213; 229; 266; 286; 299; 303; 349; 387; 397; 404; 417
DADOS	Portadoras restantes (384)

Fonte: o autor

4.6.12 Intervalo de guarda

Na saída da IFFT é inserido um intervalo de guarda, na forma de *prefixo cíclico*, de duração configurável, sendo 1/32, 1/16, 1/8 ou 1/4 da duração útil de um símbolo.

O intervalo de guarda torna o sinal mais robusto aos efeitos da propagação multi-percurso, absorvendo os *ecos* do símbolo anterior e evitando a interferência intersimbólica. A subseção 2.2.4 aborda com maiores detalhes como o prefixo cíclico é gerado e sua função.

Tabela 16: Configuração das portadoras OFDM para o segmento zero nas modulações síncronas

Sinal	Índice das portadoras
<i>Modo 1 — 108 portadoras:</i>	
CP	Ausente
SP	$\{12k + 3(j \bmod 4) \mid 0 \leq k \leq 8\}$
AC1	35; 79
AC2	Ausente
TMCC	49
DADOS	Portadoras restantes (96)
<i>Modo 2 — 216 portadoras:</i>	
CP	Ausente
SP	$\{12k + 3(j \bmod 4) \mid 0 \leq k \leq 17\}$
AC1	98; 101; 118; 136
AC2	Ausente
TMCC	23; 178
DADOS	Portadoras restantes (192)
<i>Modo 3 — 432 portadoras:</i>	
CP	Ausente
SP	$\{12k + 3(j \bmod 4) \mid 0 \leq k \leq 35\}$
AC1	7; 89; 206; 209; 226; 244; 377; 407
AC2	Ausente
TMCC	101; 131; 286; 349
DADOS	Portadoras restantes (384)

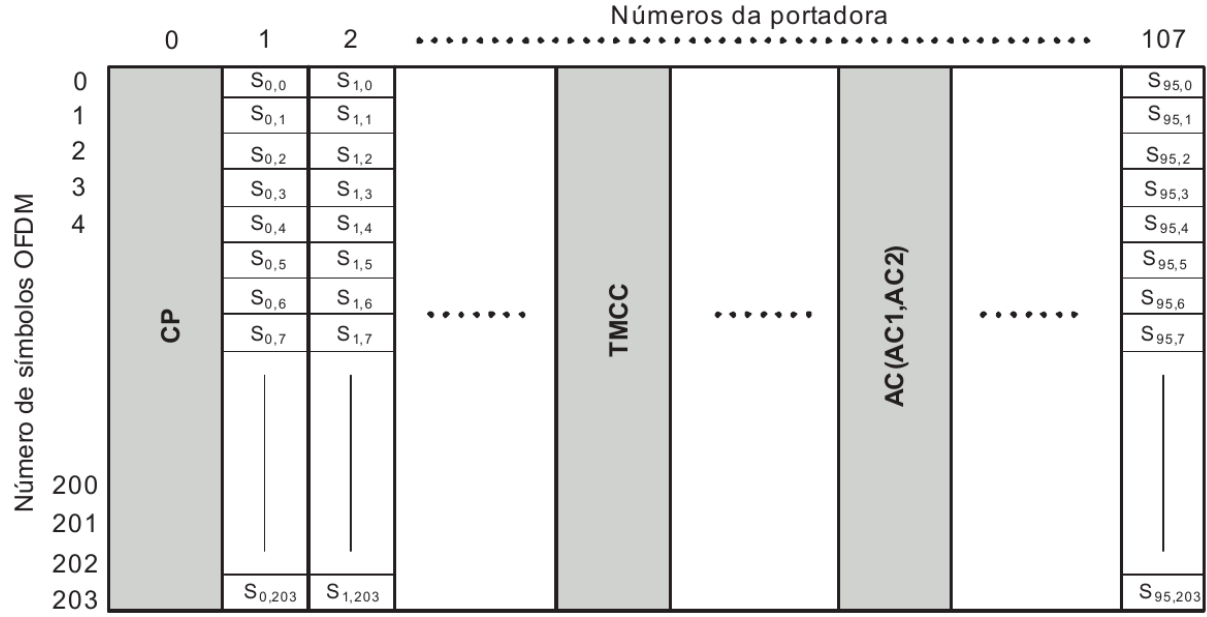
j é o índice do símbolo OFDM dentro do quadro ($0 \leq j \leq 203$)

Fonte: o autor

4.6.13 Forma de onda na saída do transmissor

A forma de onda $s(t)$ da portadora modulada pelo sinal ISDB-T é dada pelas equações 4.3 e 4.4, conforme ARIB (2014) e ABNT (2007). k é o número da subportadora OFDM; n é o número sequencial do símbolo; K é o numero total de subportadoras OFDM; T_s é a duração temporal do símbolo; T_g é a duração do intervalo de guarda; T_u é a duração da parte útil do símbolo; f_c é a frequência central do sinal de RF; K_c é o número da

Figura 63: Exemplo de quadro utilizado na modulação diferencial (DQPSK) no modo 1



$S_{i,j}$ é o i -ésimo símbolo de dados dentro do j -ésimo símbolo OFDM

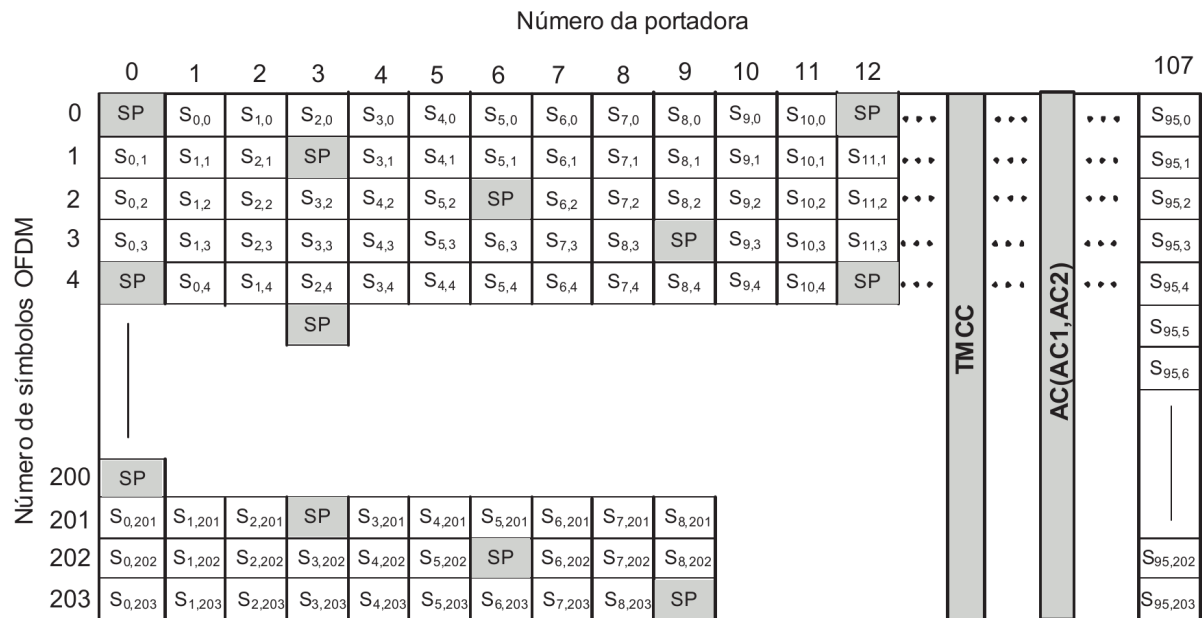
Fonte: ABNT (2007, p. 36)

portadora central; e $c(n,k)$ é o vetor complexo do símbolo n e subportadora k .

$$s(t) = \text{Re} \left\{ e^{j2\pi f_c t} \sum_{n=0}^{\infty} \sum_{k=0}^{K-1} c(n,k) \Psi(n,k,t) \right\} \quad (4.3)$$

$$\Psi(n,k,t) = \begin{cases} e^{j2\pi \frac{k-K_c}{T_u} (t-T_g-nT_s)}, & nT_s \leq t < (n+1)T_s \\ 0, & t < nT_s, t \geq (n+1)T_s \end{cases} \quad (4.4)$$

Figura 64: Exemplo de quadro utilizado nas modulações síncronas (QPSK, 16QAM e 64QAM) no modo 1



$S_{i,j}$ é o i -ésimo símbolo de dados dentro do j -ésimo símbolo OFDM

Fonte: ABNT (2007, p. 39)

5 Receptor desenvolvido

Este capítulo apresenta o desenvolvimento de um receptor de TV digital ISDB-T 1-seg de baixo custo, utilizando a tecnologia de Rádio Definido por Software com processamento em tempo real executado por um computador de mesa convencional. O receptor pode ser dividido pela sua parte de hardware e de software:

- Hardware: constitui-se do RTL-SDR - uma plataforma de hardware RDS de baixo custo - e de um computador. O RTL-SDR é responsável por entregar as amostras, em banda base, do sinal sintonizado ao computador através do barramento *Universal Serial Bus* - USB, conforme ilustrado na Figura 65;
- Software: desenvolvido na linguagem de programação C, utiliza a API provida pela biblioteca *librtlsdr* e contém todas as funções de rádio necessárias para a decodificação do sinal ISDB-T 1seg a partir das amostras em banda base do sinal sintonizado. O software desenvolvido faz a decodificação em tempo real utilizando o processador do computador (GPP).

Figura 65: Visão geral do hardware do receptor desenvolvido



Fonte: o autor

A Figura 66 apresenta toda a cadeia de processamento do transmissor (lado esquerdo) e do receptor 1-seg (lado direito) na forma de um diagrama de blocos, iniciando pelos feixes MPEG-2 TS na entrada do transmissor e encerrando na recuperação, pelo receptor, do feixe MPEG-2 TS correspondente à camada de recepção parcial.

O processamento do sinal pelo receptor segue basicamente o inverso do processamento do transmissor, ou seja, cada bloco do receptor desfaz o que o bloco equivalente do transmissor (na mesma linha da Figura 66) fez.

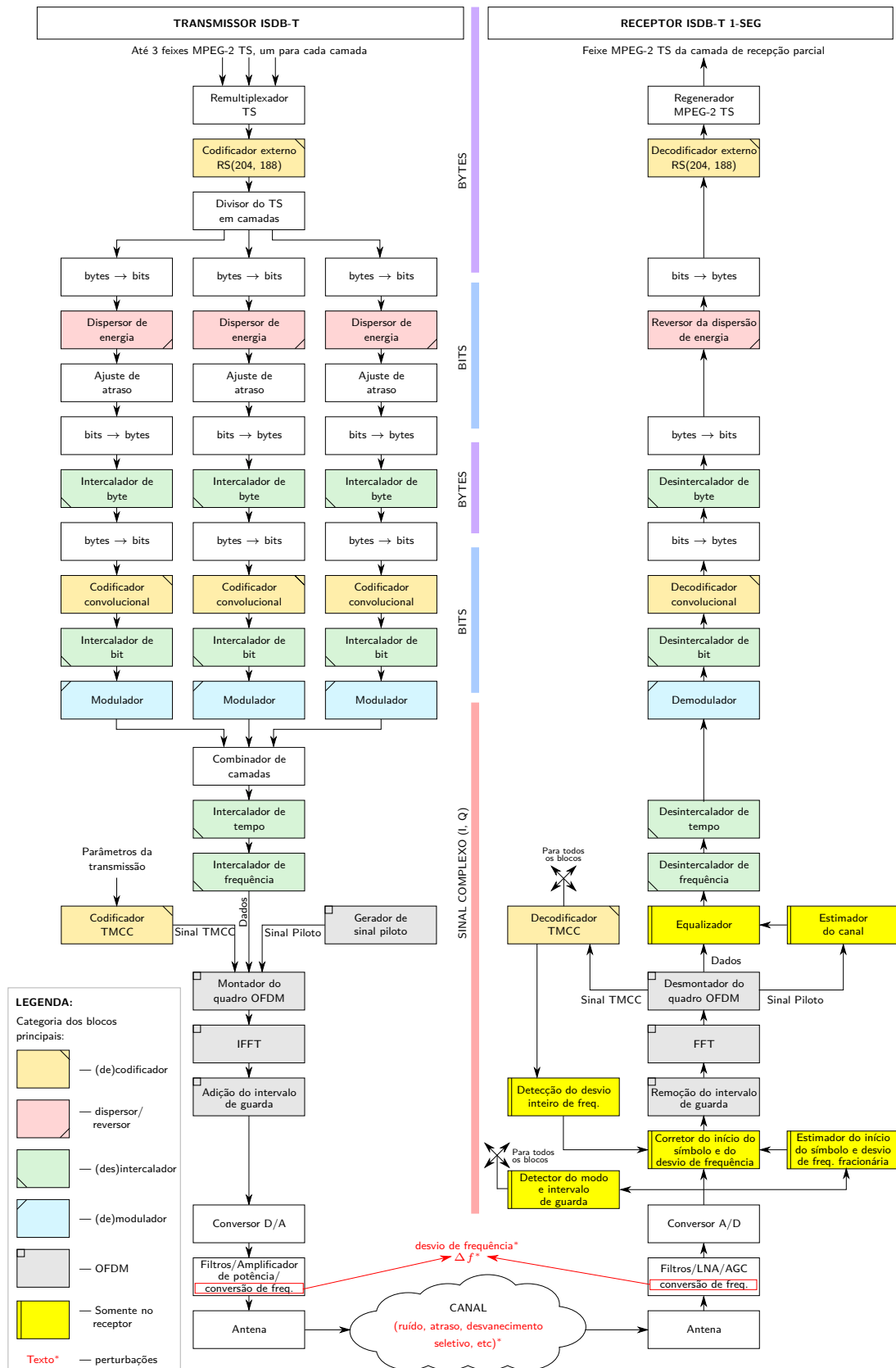
Alguns blocos do receptor não tem equivalentes no transmissor, pois no receptor há a necessidade de estimar/detectar e corrigir alguns parâmetros que não são transmitidos

ou que são desconhecidos do receptor devido à perturbações provocadas pelo canal ou por imperfeição nos componentes do transmissor e do receptor. São funções exclusivas do receptor:

- detecção do modo de transmissão e da duração do intervalo de guarda utilizados pela emissora (parâmetros não transmitidos explicitamente no ISDB-T);
- estimação e correção do desvio de frequência da portadora, causada pela diferença entre os osciladores do transmissor e do receptor (imperfeição de componentes);
- estimação e correção da posição do início do símbolo e quadro (atraso provocado pelo processamento e pelo canal, que depende da distância entre o transmissor e o receptor);
- estimação e correção da resposta do canal (equalização, para corrigir as distorções seletivas em frequência provocadas pelo canal).

O RTL-SDR é responsável por entregar ao computador um feixe contínuo de amostras em quadratura do sinal da emissora sintonizada, em banda base. O software decodificador consome estas amostras executando cada uma das etapas de recepção especificadas no lado direito da Figura 66 em tempo real, entregando em sua saída um feixe MPEG-2 TS pronto para ser reproduzido por um reproduutor de mídia (*player*) convencional, como o *VLC*, *mplayer* e *ffplay*.

Figura 66: Cadeia de processamento do transmissor ISDB-T e do receptor ISDB-T 1-seg



Fonte: o autor, baseado na ABNT (2007, p. 12) e ARIB (2014, p. 18)

5.1 Hardware

A plataforma de hardware RDS escolhida para este trabalho é o RTL-SDR, um dispositivo de baixo custo que pode ser encontrado por até US\$ 7,00, baseado no chip *Realtek RTL2832U*. A subseção 1.4.1 apresenta esta plataforma com maiores detalhes.

Sua escolha se deu por ser a plataforma de hardware RDS de menor custo disponível atualmente, capaz de sintonizar frequências de TV UHF (470–806 MHz) utilizadas na TV digital brasileira com taxa de amostragem suficiente para capturar um segmento completo do sinal ISDB-T (sinal 1-seg).

5.1.1 Taxa de amostragem

A taxa de amostragem, f_s , para o receptor desenvolvido é determinada por:

$$f_s = \frac{N}{T_u}$$

Onde N é o tamanho da FFT escolhida e T_u é a duração útil de símbolo (ver Tabela 12). O tamanho da FFT deve ser maior ou igual ao número de portadoras OFDM do segmento central ($N \geq s_c$), permitindo assim a recepção de todas as portadoras. No ISDB-T, tanto s_c quanto T_u dependem do modo de operação, mas como são proporcionais entre si, resultam em uma única taxa de amostragem qualquer que seja o modo, conforme apresentado pela Tabela 17.

Outra restrição para o tamanho da FFT, é que ela deve ser uma potência de 2 para permitir o uso de algoritmos eficientes no seu cálculo. Assim, a menor FFT que comporta s_c portadoras OFDM do segmento central pode ser calculado¹ por:

$$N_{\min} = 2^{\lceil \log_2 s_c \rceil}$$

Tabela 17: Processo de escolha da frequência de amostragem f_s do receptor desenvolvido

Modo	T_u	s_c	N_{\min}	$f_{s \min}$	$N_{2\times}$	$f_{s 2\times}$
1	252 μs	108	128	$\approx 507.936,5$ Hz	256	$\approx 1.015.873$ Hz
2	504 μs	216	256	$\approx 507.936,5$ Hz	512	$\approx 1.015.873$ Hz
3	1008 μs	432	512	$\approx 507.936,5$ Hz	1024	$\approx 1.015.873$ Hz

Fonte: o autor

¹ O operador matemático $\lceil x \rceil$ denota a operação *teto*, o menor inteiro maior ou igual a x

A Tabela 17 apresenta o N_{\min} calculado para cada modo e também a frequência de amostragem para o caso $N = N_{\min}$, utilizando a FFT com comprimento menor possível e potência de 2. Neste caso, a frequência de amostragem $f_s = f_{s \min}$ é de $\approx 507.936,5$ Hz.

Porém, conforme subseção 1.4.1.4, o chip RTL2832U não suporta esta taxa de amostragem mínima. A solução foi escolher uma taxa de amostragem maior, que resultasse em uma FFT de tamanho também potência de 2. Assim, fazendo $N = 2N_{\min} = N_{2\times}$, obtém-se $f_s = 2f_{s \min} = f_{s 2\times} \approx 1.015.873$ Hz (ver as últimas 2 colunas da Tabela 17). Esta taxa é suportada pelo RTL-SDR, e foi a escolhida para o desenvolvimento do receptor ISDB-T 1 seg.

5.2 Software

5.2.1 Arquitetura

A implementação dos blocos funcionais do receptor da Figura 66 é feita através de diversos *componentes* de software, desenvolvidos em C sob o sistema operacional Linux. Estes componentes e suas relações são apresentados na Figura 67.

5.2.2 Blocos funcionais

Nesta seção serão detalhados cada um dos blocos funcionais do receptor da Figura 66.

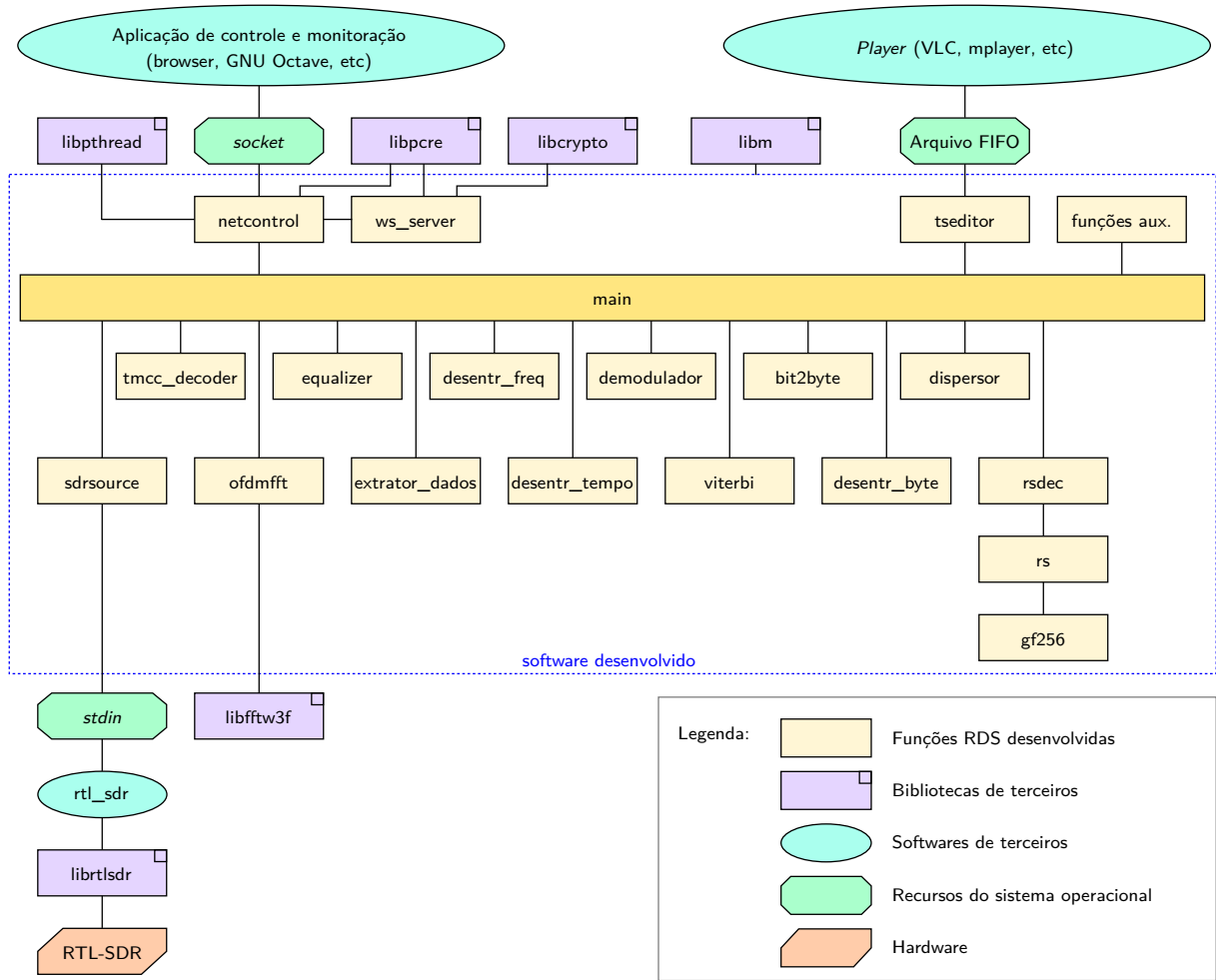
5.2.2.1 Detector do modo de transmissão e do intervalo de guarda

Bloco:	Detector do modo de transmissão e do intervalo de guarda
Implementado pelo componente:	funções aux. \rightarrow <code>isdbt_busca_modos_ig()</code>
Entrada(s):	$r[k]$: sinal amostrado da entrada do receptor (<code>complex float</code>)
Saída(s):	modo (<code>int</code>) div_ig : divisor do intervalo de guarda (<code>int</code>)
Função equivalente no transmissor:	—

O modo e o intervalo de guarda utilizados na transmissão não estão contidos no TMCC ou em qualquer outra informação transmitida, e devem ser obtidos pelo receptor às cegas. A detecção destes parâmetros ocorre somente no primeiro momento de operação do receptor.

Estes parâmetros são obtidos através de testes com cada uma das 12 combinações possíveis entre `{modo, intervalo de guarda}` utilizando a função `sync_ofdm()` descrita

Figura 67: Diagrama dos componentes do receptor ISDB-T 1-seg desenvolvido



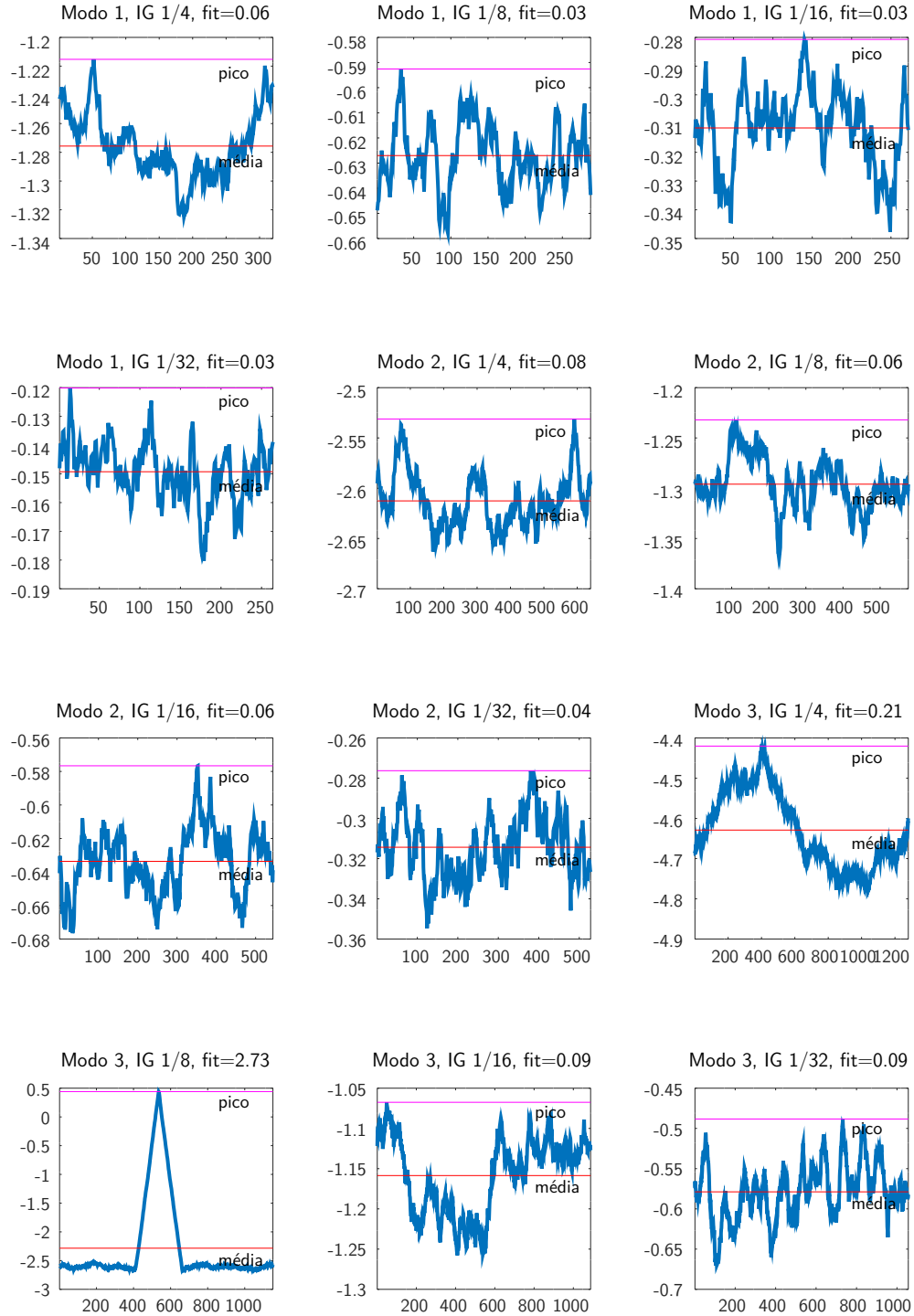
Fonte: o autor

na subseção 5.2.2.2. Para cada uma das tentativas, a função $f_1(k) = |\gamma(\cdot)| - \rho \Phi(\cdot)$ é calculada, e a métrica de ajuste fit é obtida a partir da razão entre o valor máximo de $f_1(k)$ e sua média:

$$fit = \frac{\max f_1(k)}{\overline{f_1(k)}}$$

Portanto, fit é uma medida da *nitidez* do pico de $f_1(k)$. A tentativa que possuir o maior fit é escolhida como aquela que possui o modo {modo, intervalo de guarda} correto.

A Figura 68 apresenta todas as 12 tentativas para um sinal real. É possível observar que a opção que possui a maior métrica fit é o modo 3 com intervalo de guarda 1/8, possuindo o pico bem definido. A métrica para as demais opções é no mínimo 30 vezes inferior que a melhor métrica, o que comprova a eficácia deste método.

Figura 68: Gráficos de $f_1(k)$ para cada modo de transmissão e intervalo de guarda possíveis

Fonte: o autor

5.2.2.2 Estimador do início do símbolo e do desvio de frequência fracionária

Bloco:	Estimador do início do símbolo e do desvio de frequência fracionária
Implementado pelo componente:	<code>funções aux. → sync_ofdm()</code>
Entrada(s):	$r[k]$: sinal amostrado da entrada do receptor (<code>complex float</code>)
Saída(s):	$\hat{\theta}$: posição do início do símbolo dentro de $r[k]$ (<code>int</code>) $\hat{\mathcal{E}}$: desvio fracionário de frequência (<code>float</code>)
Função equivalente no transmissor:	—

O objetivo deste bloco é de estimar a posição de início do símbolo OFDM, $\hat{\theta}$, e o desvio de frequência fracionário, $\hat{\mathcal{E}}$.

Dentro do vetor complexo recebido $r[k]$, $\hat{\theta}$ indica a estimativa ML da posição de início do primeiro símbolo OFDM encontrado no vetor.

O desvio Δf entre a frequências da portadora do transmissor e do conversor de frequência do receptor (conhecido como *Carrier Frequency Offset – CFO*), em Hz, pode ser decomposto em sua parte inteira e fracionária:

$$\Delta f = (\delta + \mathcal{E}) \cdot c_s \quad \delta \in \mathbb{Z}; \quad \mathcal{E} \in \mathbb{R}; \quad -0,5 < \mathcal{E} < 0,5$$

Onde c_s é o espaçamento entre subportadoras OFDM em Hz (ver Tabela 12), δ é o desvio inteiro de frequência em unidades de c_s e \mathcal{E} é o desvio fracionário, também normalizado em c_s .

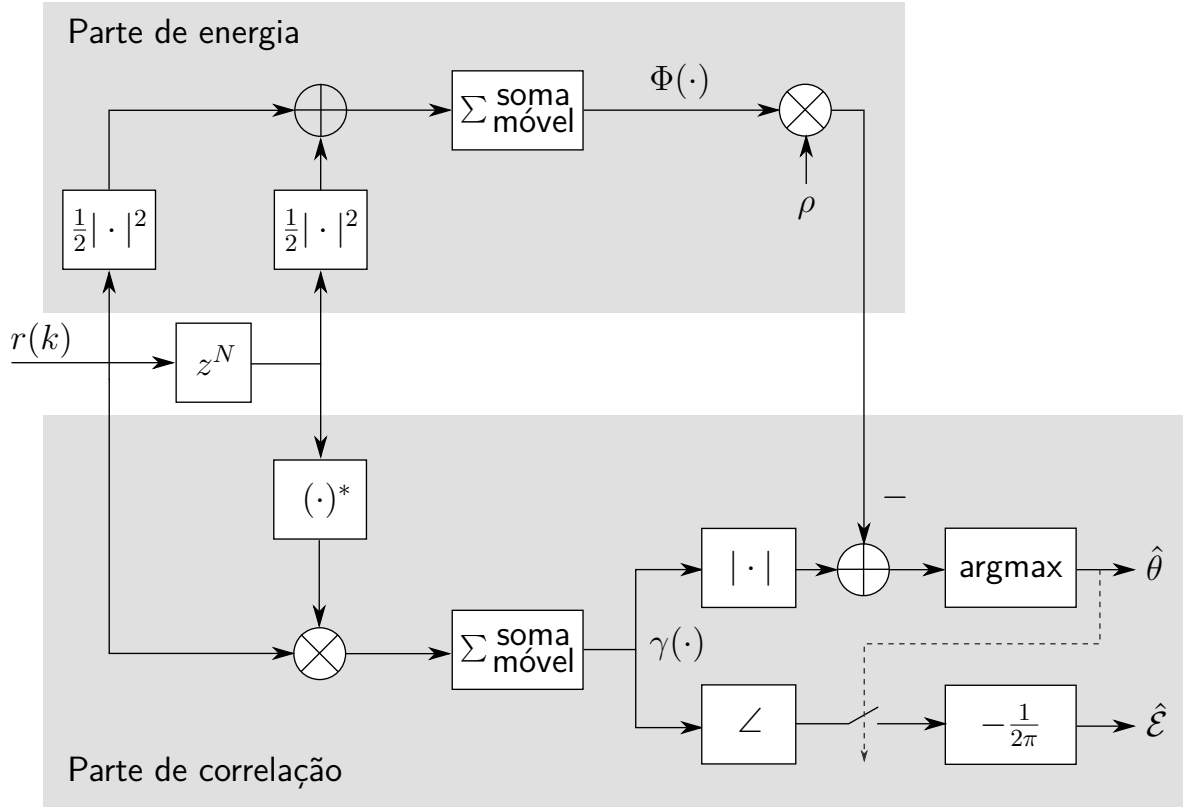
O segunda saída deste bloco é o $\hat{\mathcal{E}}$, uma estimativa ML para o desvio fracionário de frequência \mathcal{E} que ocorre entre os osciladores do transmissor e o receptor.

O bloco implementado utilizada o método de Beek, Sandell e Börjesson (1997), que é um estimador conjunto de máxima verossimilhança. A Figura 69 apresenta o diagrama funcional deste estimador.

Este método se baseia na correlação entre duas partes redundantes que ocorrem em todo símbolo OFDM do ISDB-T: (1) \mathcal{I} — o prefixo cíclico, com L amostras de comprimento; (2) \mathcal{I}' — o final de um símbolo OFDM, de mesmo comprimento. Estas duas partes em que a correlação é calculada estão separadas por N amostras, onde N é o tamanho da FFT empregada no receptor, e formam duas janelas deslizantes sobre $r[k]$. Esta ideia básica é ilustrada na Figura 70.

Considerando um intervalo de observação contendo $2N + L$ amostras consecutivas de $r[k]$, ao se deslizar as janelas supracitadas calculando a correlação $\gamma(m)$ (Equação 5.1) para posição m , $1 \leq m \leq N$, verifica-se que o máximo do módulo da correlação ocorre em

Figura 69: Diagrama do estimador do início do símbolo e do desvio de frequência fracionária



Fonte: o autor, baseado em Beek, Sandell e Börjesson (1997)

$m = \theta$, quando o início da janela está perfeitamente alinhado com o início de um símbolo. Nesta posição, o desvio de frequência fracionário \mathcal{E} é proporcional ao ângulo (\angle) da mesma correlação, conforme a Equação 5.2.

$$\gamma(m) = \sum_{k=m}^{m+L-1} r[k]r^*[k+N] \quad (5.1)$$

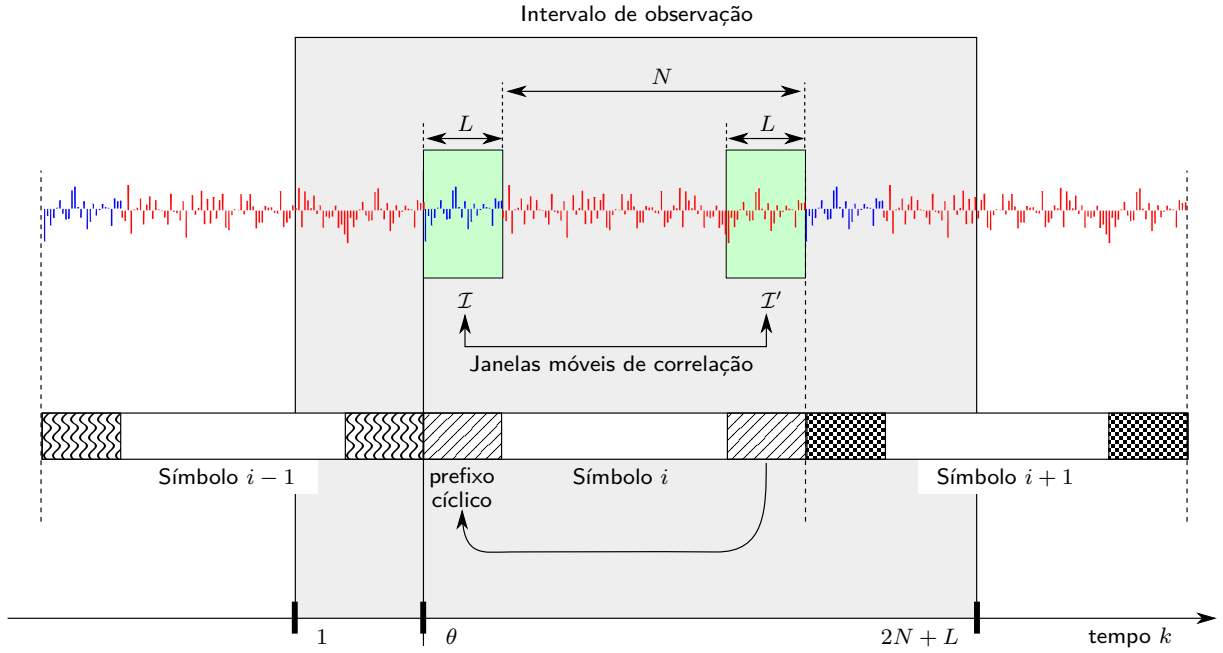
$$\hat{\mathcal{E}} = -\frac{1}{2\pi} \angle \gamma(\theta) \quad (5.2)$$

Porém, conforme Beek, Sandell e Börjesson (1997), a expressão que estima θ pela máxima verossimilhança é ligeiramente diferente da simples correlação, pois inclui um segundo termo que é subtraído de $|\gamma(m)|$, que leva em conta a energia do sinal. A Equação 5.3 apresenta o cálculo de $\hat{\theta}$.

$$\hat{\theta} = \arg \max_{\theta} \{ |\gamma(\theta)| - \rho \cdot \Phi(\theta) \} \quad (5.3)$$

O operador $\arg \max_{\theta} \{f(\theta)\}$ retorna o valor de θ em que ocorre o máximo da função $f(\theta)$. O termo Φ é a energia do sinal recebido (inclui o ruído), e é calculado conforme a

Figura 70: Sinal OFDM e seu prefixo cíclico



Fonte: o autor, baseado em Beek, Sandell e Börjesson (1997)

equação:

$$\Phi(m) = \frac{1}{2} \sum_{k=m}^{m+L-1} |r[k]|^2 + |r[k+N]|^2$$

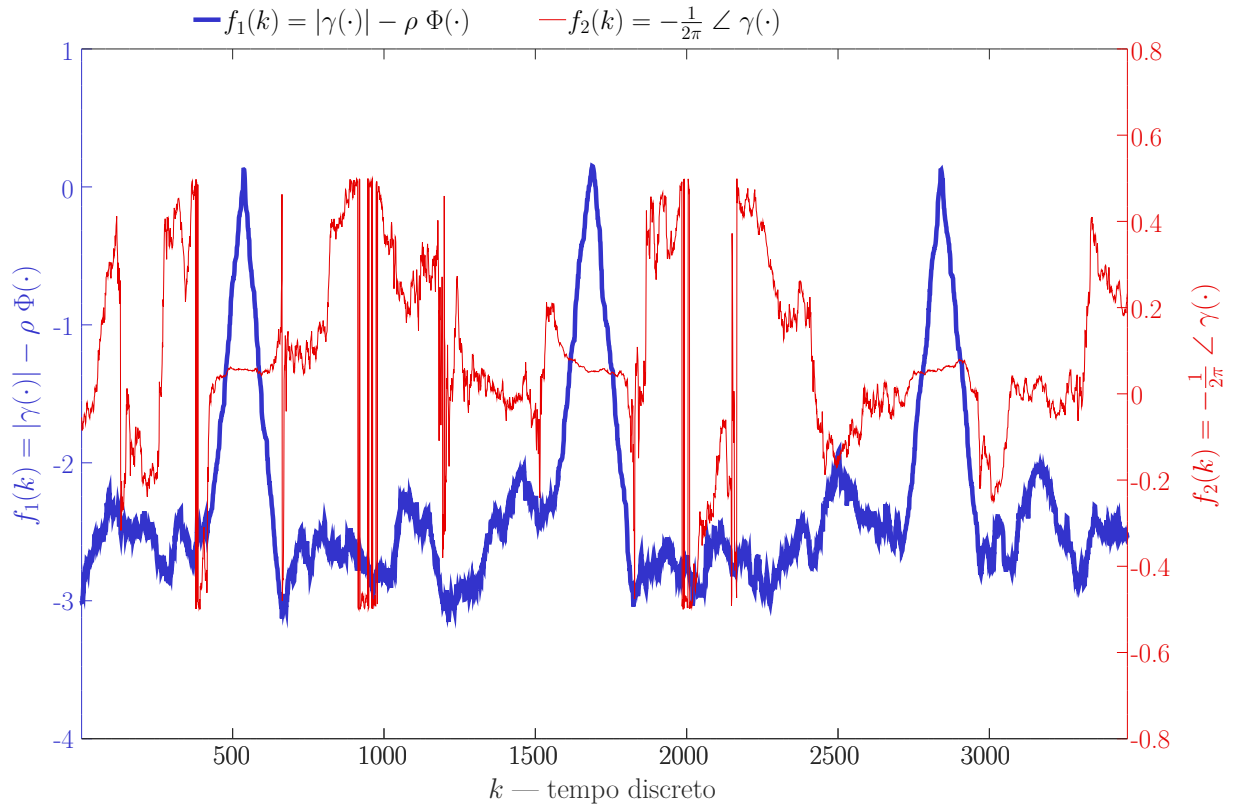
Já o fator ρ representa a proporção do sinal recebido que realmente é sinal (desconsiderando o ruído). É calculado a partir da relação sinal-ruído (*Signal-to-Noise Ratio - SNR*):

$$\rho = \frac{\text{SNR}}{\text{SNR} + 1}$$

No receptor implementado, o SNR não é calculado, mas foi fixado arbitrariamente em 6, não apresentando problemas perceptíveis no funcionamento deste bloco.

A Figura 71 apresenta duas curvas geradas a partir de um sinal real contendo 3 símbolos OFDM com $N = 1024$ e $L = 128$. A curva 1, em azul e negrito mostra a função $f_1(k) = |\gamma(\cdot)| - \rho \Phi(\cdot)$. A curva 2, em vermelho e traço fino apresenta a função $f_2(k) = -\frac{1}{2\pi} \angle \gamma(\cdot)$. A notação das variáveis são as mesmas da Figura 69. As posições k onde ocorrem os picos da curva $f_1(k)$ são as estimativas ML para as posições de início dos símbolos, $\hat{\theta}$, contidos em $r[k]$. Nos instantes k destes picos, o valor de $f_2(k)$ é a estimativa ML do desvio de frequência fracionário, $\hat{\mathcal{E}}$.

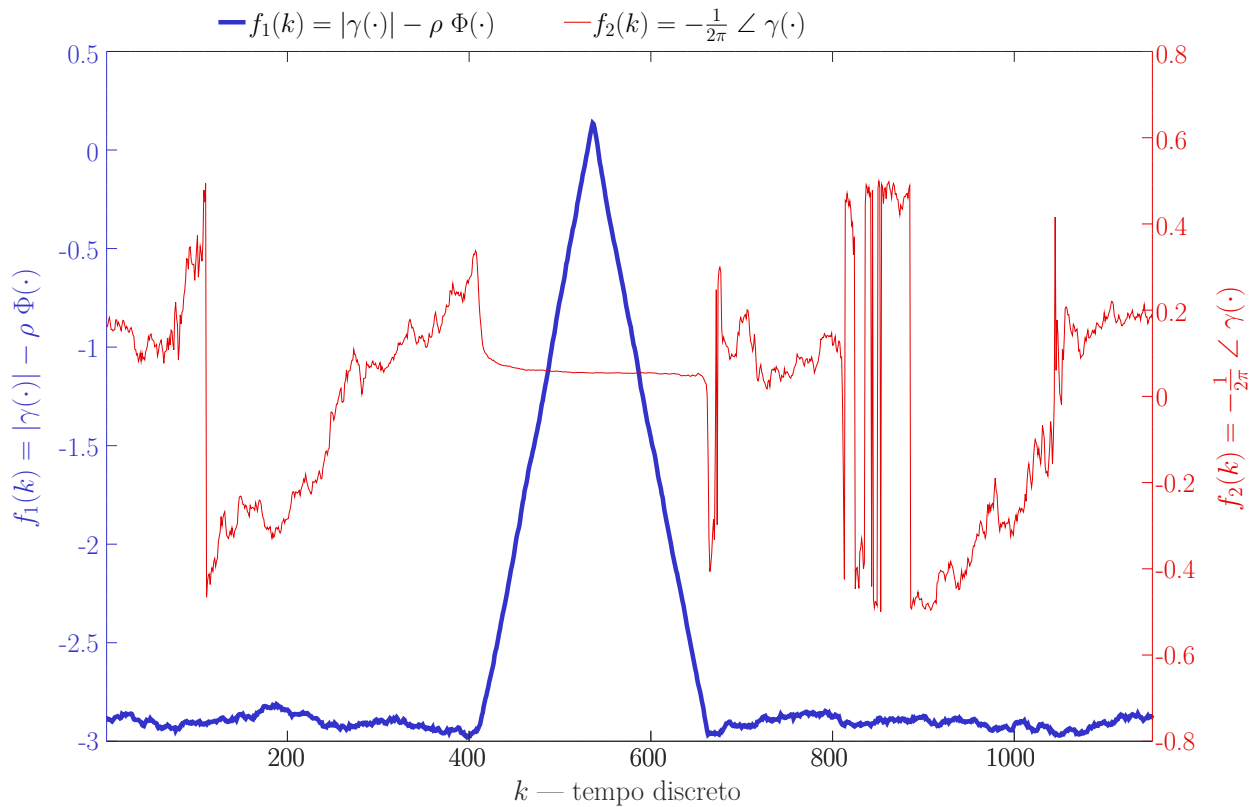
A Figura 72 é gerada com os mesmos dados da Figura 71, porém $\gamma(\cdot)$ e $\Phi(\cdot)$ foram calculados para os últimos 32 símbolos OFDM e o resultado foi tomado como sendo a

Figura 71: Gráfico para obtenção das estimativas $\hat{\theta}$ e $\hat{\mathcal{E}}$ com 3 símbolos OFDM

Fonte: o autor

média destes símbolos. É possível observar no gráfico que o pico de $f_1(k)$ é muito mais bem definido e o valor de $f_2(k)$ é muito mais estável naquela região do pico. Isto torna as estimativas $\hat{\theta}$ e $\hat{\mathcal{E}}$ mais robustas quando o sinal é ruidoso. A implementação deste bloco seguiu esta forma.

Figura 72: Gráfico para obtenção das estimativas $\hat{\theta}$ e $\hat{\mathcal{E}}$ utilizando a média dos últimos 32 símbolos OFDM



Fonte: o autor

5.2.2.3 Corretor do início do símbolo e do desvio de frequência

Bloco: Corretor do início do símbolo e do desvio de frequência	
Implementado pelo componente:	<code>main</code>
Entrada(s):	$r[k]$: sinal amostrado da entrada do receptor (<code>complex float</code>) θ : posição do início do símbolo (<code>int</code>) \mathcal{E} : desvio de frequência fracionário (<code>float</code>) δ : desvio de frequência inteiro (<code>int</code>)
Saída(s):	$y[k]$: vetor com a primeira amostra sincronizada em um símbolo e a frequência corrigida (<code>complex float</code>)
Função equivalente no transmissor:	—

Este bloco corrige o desvio de frequência e alinha o vetor de saída com o início de um símbolo OFDM através do deslocamento do vetor da entrada e pela multiplicação

complexa de cada amostra conforme a equação:

$$y[k] = r[k + \theta] \cdot e^{-j2\pi\Delta f k/f_s}$$

Onde f_s é a frequência de amostragem utilizada pelo receptor, c_s é o espaçamento entre subportadoras OFDM em Hz, e:

$$\Delta f = (\delta + \mathcal{E}) \cdot c_s$$

5.2.2.4 Remoção do intervalo de guarda

Bloco:	Remoção do intervalo de guarda
Implementado pelo componente:	<code>main</code>
Entrada(s):	$x[k]$: vetor complexo de comprimento $N + L$ (<code>complex float</code>)
Saída(s):	$y[k]$: vetor complexo de comprimento N (<code>complex float</code>)
Função equivalente no transmissor:	Adição do intervalo de guarda

Este bloco é executado em cada símbolo OFDM completo, de comprimento $N + L$. O bloco descarta as L primeiras amostras da entrada, entregando em sua saída um vetor de comprimento N contendo apenas a parte útil do símbolo OFDM:

$$y[k] = x[k + L], \quad 0 \leq k \leq N - 1$$

5.2.2.5 FFT

Bloco:	FFT
Implementado pelo componente:	<code>ofdmfft</code>
Entrada(s):	$x[k]$: vetor complexo de comprimento N , no domínio do tempo (<code>complex float</code>)
Saída(s):	$y[k]$: vetor complexo de comprimento N , no domínio da frequência (<code>complex float</code>)
Função equivalente no transmissor:	IFFT

A transformada rápida de Fourier, FFT, de tamanho N é executada utilizando a biblioteca de terceiros *libfftw3f* do pacote *FFTW - Fastest Fourier Transform in the West* (FFTW, s. d.; FRIGO, 1999), convertendo as amostras da entrada para o domínio da frequência.

5.2.2.6 Desmontador do quadro OFDM

Bloco:	Desmontador do quadro OFDM
Implementado pelo componente:	<code>extrator_dados</code>
Entrada(s):	$x[k]$: vetor complexo (<code>complex float</code>) $y[k]$: vetor complexo, sinal piloto (<code>complex float</code>)
Saída(s):	$z[k]$: vetor complexo, sinal TMCC (<code>complex float</code>) $d[k]$: vetor complexo, dados (<code>complex float</code>)
Função equivalente no transmissor:	Montador do quadro OFDM

Este bloco faz a separação do sinal piloto, do sinal TMCC e dos dados contidos em um quadro OFDM. Esta separação se dá conforme a estrutura do quadro OFDM apresentada na subseção 4.6.10. Inicialmente, este bloco só habilita a saída do sinal TMCC, $z[k]$. As demais saídas $y[k]$ e $d[k]$ ficam inibidas até o bloco *Decodificador TMCC* obter os parâmetros da transmissão P_{tx} . Isto ocorre pois a separação de $y[k]$ e $d[k]$ do quadro OFDM depende do tipo de quadro, contido no TMCC.

5.2.2.7 Decodificador TMCC e detector do desvio de frequência inteira

Blocos:	Decodificador TMCC / Detector do desvio de frequência inteira
Implementado pelo componente:	<code>tmcc_decoder</code>
Entrada(s):	$x[k]$: vetor complexo, sinal TMCC de comprimento 204 (<code>complex float</code>)
Saída(s):	P_{tx} : parâmetros da transmissão (<code>struct</code>) δ : desvio de frequência inteiro (<code>int</code>)
Função equivalente no transmissor:	Codificador TMCC

O vetor $b[k]$ contendo os 203 bits² do quadro TMCC são obtidos por um demodulador DBPSK implementado através da seguinte equação, para $0 \leq k \leq 202$:

$$b[k] = \begin{cases} 1 & \text{se } \text{Re}(x[k+1] \cdot x[k]^*) < 0 \\ 0 & \text{caso contrário} \end{cases}$$

A subseção 4.6.9 apresenta as informações contidas nestes 203 bits de $b[k]$, que alimentam a estrutura dos parâmetros da transmissão P_{tx} . As sequências alternadas de

² Os 204 símbolos TMCC são demodulados em 203 bits, pois o primeiro símbolo serve apenas como referência para demodulação diferencial, não sendo convertido em um bit

bits de sincronismo (0011010111101110 e 1100101000010001) presentes no início do TMCC são referências utilizadas no sincronismo do quadro OFDM, para a obtenção das informações do TMCC e para estimar o desvio de frequência inteiro δ .

O bloco *detector do desvio de frequência inteira* trabalha em conjunto com o *decodificador TMCC*. Inicialmente, é criado o conjunto $S_\delta = \{0, -1, 1, -2, 2, \dots, -\delta_{\max}, \delta_{\max}\}$, e o desvio de frequência inteira δ é definido como sendo o primeiro elemento deste conjunto: $\delta = S_\delta[0] = 0$. Se o TMCC não puder ser decodificado dentro do tempo limite T_{\lim} , δ é escolhido como sendo o próximo elemento do conjunto S_δ . Este processo continua até que haja sucesso na decodificação do TMCC, neste momento δ representa o verdadeiro desvio de frequência inteiro.

5.2.2.8 Estimador do canal e equalizador

Blocos:	Estimador do canal / Equalizador
Implementado pelo componente:	equalizer
Entrada(s):	$x[k]$: vetor complexo, dados (complex float) $p[k]$: vetor complexo, pilotos (complex float)
Saída(s):	$y[k]$: vetor complexo, dados equalizados (complex float)
Função equivalente no transmissor:	—

A estimativa da resposta do canal e a equalização são conduzidas no domínio da frequência, após a FFT. Os pilotos espalhados presentes nos quadros OFDM são usados como referência (ver subseção 4.6.8).

A estimativa da resposta do canal \hat{H} nas portadoras onde há pilotos é obtida pela razão entre os sinais dos pilotos espalhados recebidos, $p[k]$, e o sinal piloto de referência (piloto transmitido conforme a subseção 4.6.8), $R[k]$:

$$\hat{H}[k] = \frac{p[k]}{R[k]}, \quad k \in S_{\text{pilotos}}$$

Onde S_{pilotos} é o conjunto de portadoras que contém um sinal piloto e k é o número da portadora.

Para as demais portadoras que não contém pilotos ($k \notin S_{\text{pilotos}}$), a estimativa da resposta do canal é obtida através da interpolação linear das estimativas do canal obtidas dos pilotos mais próximos, $\hat{H}[x]$ e $\hat{H}[x + 12]$:

$$\hat{H}[k] = \hat{H}[x] \cdot \left(1 - \frac{k - x}{12}\right) + \hat{H}[x + 12] \cdot \frac{k - x}{12}, \quad x \leq k \leq x + 12$$

A partir da estimativa da resposta do canal $\hat{H}[k]$, a equalização é feita através da multiplicação complexa do sinal de entrada $x[k]$ com o inverso da estimativa de resposta do canal, $\hat{H}^{-1}[k]$:

$$y[k] = x[k] \cdot \hat{H}^{-1}[k]$$

5.2.2.9 Desintercalador de frequência

Bloco:	Desintercalador de frequência
Implementado pelo componente:	<code>desentr_freq</code>
Entrada(s):	$x[k]$: vetor complexo, dados (<code>complex float</code>)
Saída(s):	$y[k]$: vetor complexo, dados (<code>complex float</code>)
Função equivalente no transmissor:	Intercalador de frequência, somente parte Aleatorização de portadoras

Conforme a subseção 4.6.7, para o segmento zero, o a única etapa do intercalador de frequência executada no transmissor é a *aleatorização de portadoras*. Este bloco desfaz esta aleatorização através do mapeamento inverso, $x[k] \rightarrow y[m]$, onde a relação entre k e m é dada pelas as extensas tabelas apresentadas por ARIB (2014, p. 41) ou ABNT (2007, p. 33).

5.2.2.10 Desintercalador de tempo

Bloco:	Desintercalador de tempo
Implementado pelo componente:	<code>desentr_tempo</code>
Entrada(s):	$x[k]$: vetor complexo, dados (<code>complex float</code>)
Saída(s):	$y[k]$: vetor complexo, dados (<code>complex float</code>)
Função equivalente no transmissor:	Intercalador de tempo

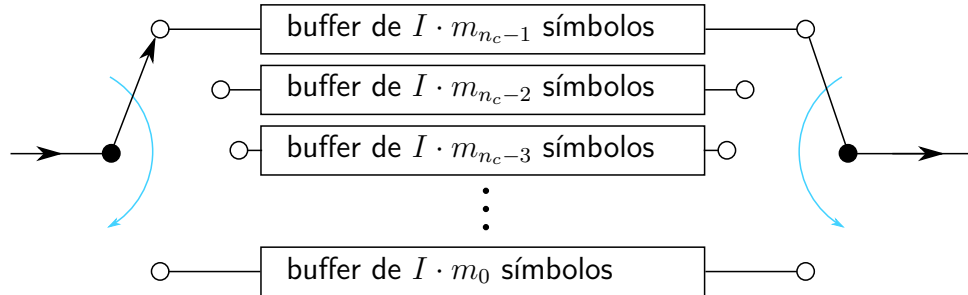
O desintercalador de tempo desfaz o *embaralhamento* dos símbolos no domínio do tempo feito pelo intercalador de tempo no transmissor (subseção 4.6.6).

A Figura 73 apresenta o circuito do desintercalador de tempo, composto por n_c buffers FIFO de comprimentos variáveis e de chaveadores sincronizados na entrada e na saída. Nesta figura:

$$m_i = i \cdot 5 \mod 96, \quad 0 \leq i \leq n_c - 1$$

A constante I é um parâmetro de transmissão contido no TMCC, que pode assumir um dos quatro valores da Tabela 14 para um determinado modo de transmissão.

Figura 73: Funcionamento do desintercalador de tempo



Fonte: o autor

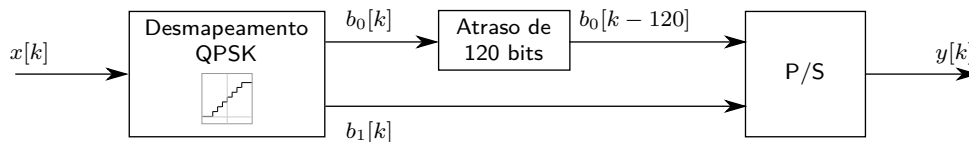
5.2.2.11 Demodulador e desintercalador de bit

Blocos:	Demodulador / Desintercalador de bit
Implementado pelo componente:	demodulador
Entrada(s):	$x[k]$: vetor complexo, dados (<code>complex float</code>)
Saída(s):	$y[k]$: vetor de soft bits demodulados (<code>int</code>)
Funções equivalentes no transmissor:	Modulador / Intercalador de bit

O demodulador e o desintercalador de bit funcionam em conjunto, desfazendo a ação dos blocos equivalentes do transmissor (subseção 4.6.5). Esta seção apresenta a demodulação QPSK, a mais utilizada pelas emissoras para a recepção parcial.

Estes blocos em conjunto operam em 3 etapas: desmapeamento QPSK, atraso de bits, e conversor paralelo/série, conforme apresentado pelo diagrama da Figura 74.

Figura 74: Funcionamento do demodulador QPSK



Fonte: o autor

O desmapeamento QPSK é configurável, podendo fazer a quantização em $l = 2, 4$ ou 8 níveis. A quantização em 2 níveis é chamada de *decisão rígida*, pois faz a comparação do nível do sinal de entrada com um limiar, resultando em sua saída os bits (b_0, b_1) que podem assumir o valor zero ou um.

As quantizações em $l = 4$ ou 8 níveis são chamadas de *decisões brandas*, e produzem *soft bits* na saída que podem estar no intervalo de $0 \leq (b_0, b_1) \leq 3$ ou $0 \leq (b_0, b_1) \leq 7$, respectivamente. O número de níveis de quantização, l , é sempre uma potência de 2 para permitir a representação eficiente de um soft bit por um conjunto de q bits, sendo portanto $l = 2^q$.

Conforme a subseção 3.2.4, o uso da decisão branda no demodulador é vantajosa pois entrega em sua saída a informação da confiabilidade de cada bit, deixando a escolha binária para o decodificador convolucional. Isto traz ganhos de performance importantes para o receptor.

A saídas do desmapeador QPSK são calculadas por:

$$b_0[k] = D(\text{Re}(x[k]))$$

e

$$b_1[k] = D(\text{Im}(x[k]))$$

Onde $D(\cdot)$ é uma função linear de saída limitada ao intervalo $[0, l - 1]$, dada pela equação³:

$$D(X) = \begin{cases} 0 & \text{se } \lfloor aX + b \rfloor < 0 \\ l - 1 & \text{se } \lfloor aX + b \rfloor > l - 1 \\ \lfloor aX + b \rfloor & \text{nos demais casos} \end{cases}$$

As constantes a e b dependem do número de níveis de quantização, l . Considerando que os pontos da constelação, tanto para o eixo real quanto para o imaginário, estão normalizados (-1 e 1), temos:

$$a = \frac{1 - l}{2}$$

e

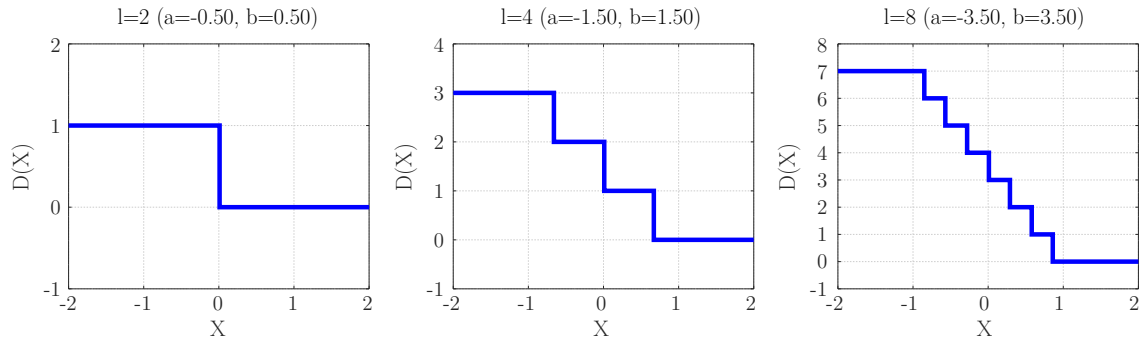
$$b = -a$$

A Figura 75 apresenta o gráfico de $D(X)$ para os três casos: $l = 2$, $l = 4$ e $l = 8$.

O desintercalamento de bit é executado pelo *atraso de 120 bits* na saída $b_0[k]$ do desmapeador QPSK. É implementado através de um buffer do tipo FIFO. O bloco P/S converte o fluxo paralelo em um fluxo serial, intercalando entre as entradas $b_0[k - 120]$ e $b_1[k]$.

³ $\lfloor \cdot \rfloor$ significa a operação de arredondamento para o inteiro mais próximo

Figura 75: Função $D(X)$ para o desmapeamento QPSK em 3 diferentes níveis de quantização



Fonte: o autor

5.2.2.12 Decodificador Viterbi

Bloco:	Decodificador Viterbi
Implementado pelo componente:	<code>viterbi</code>
Entrada(s):	$x[k]$: vetor de soft bits (<code>int</code>)
Saída(s):	$y[k]$: vetor de bits decodificados (<code>int</code>)
Função equivalente no transmissor:	Codificador convolucional

O funcionamento do decodificador Viterbi é descrito em detalhes na subseção 3.2.4. O decodificador Viterbi implementado pode operar com entrada quantizada em $l = 2, 4$ ou 8 níveis, conforme opção configurada no demodulador.

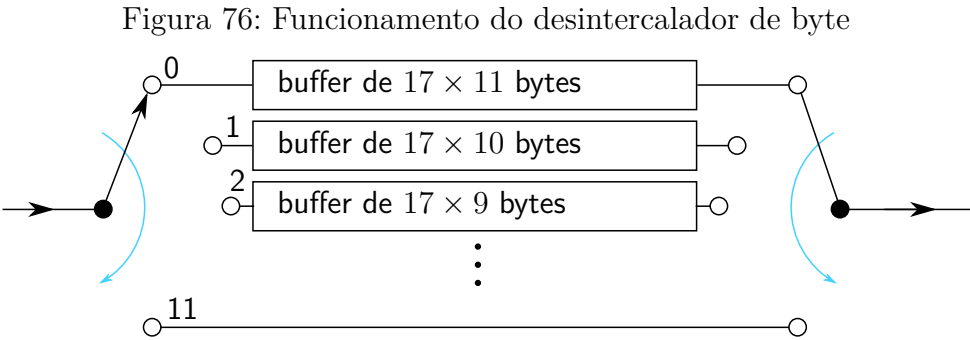
O *traceback* possui comprimento h configurável (definido como 128 por padrão), e pode utilizar o método duplo (ativo por padrão) ou simples.

5.2.2.13 Desintercalador de byte

Bloco:	Desintercalador de byte
Implementado pelo componente:	<code>desentr_byte</code>
Entrada(s):	$x[k]$: vetor de bytes (<code>int</code>)
Saída(s):	$y[k]$: vetor de bytes (<code>int</code>)
Função equivalente no transmissor:	Intercalador de byte

O desintercalador de byte desfaz o *embaralhamento* na ordem dos bytes feito pelo intercalador de byte no transmissor (subseção 4.6.3).

A Figura 76 apresenta o circuito do desintercalador de byte, composto por 12 buffers FIFO de comprimentos decrescentes (187 a 0) e de chaveadores sincronizados na entrada e na saída que comutam a cada byte na entrada de forma cíclica.



Fonte: o autor

5.2.2.14 Reversor da dispersão de energia

Bloco:	Reversor da dispersão de energia
Implementado pelo componente:	dispersor
Entrada(s):	$x[k]$: vetor de bits (<code>int</code>)
Saída(s):	$y[k]$: vetor de bits (<code>int</code>)
Função equivalente no transmissor:	Dispersor de energia

Este bloco faz a operação lógica *OU exclusivo* bit a bit do sinal de entrada com os bits emitidos por um gerador de sinal binário pseudo-aleatório (Pseudo-Random Binary Sequence — PRBS), exatamente da mesma forma como o transmissor faz (ver subseção 4.6.2). Esta operação sendo igual à do transmissor desfaz completamente o embaralhamento provocado no mesmo.

Assim como no transmissor, os registradores de deslocamento do gerador PRBS são reiniciados a cada quadro OFDM com o conteúdo 100101010000000. Os bytes de sincronismo dos pacotes MPEG-2 TS (hexadecimal: 0x47h / binário: 0100 0111 / decimal: 71) não são submetidos a esta operação.

5.2.2.15 Decodificador Reed-Solomon

Bloco:	Decodificador Reed-Solomon
Implementado pelos componentes:	<code>rsdec</code> e <code>rs</code>
Entrada(s):	$r(x)$: pacote/vetor de 204 bytes (<code>int</code>)
Saída(s):	$\hat{m}(x)$: pacote/vetor MPEG-2 TS de 188 bytes decodificados (<code>int</code>)
Função equivalente no transmissor:	Codificador Reed-Solomon

O decodificador Reed-Solomon é o bloco mais complexo do receptor, em termos do número de linhas de código e dos conhecimentos necessários sobre álgebra abstrata para o seu desenvolvimento. A implementação do decodificador utiliza uma vasta gama de funções auxiliares⁴ para se trabalhar com a aritmética de *escalares* no *campo de Galois* $GF(256)$, como a soma, multiplicação, divisão, potenciação e logaritmo. Sobre esta estrutura para o cálculo de escalares, há ainda funções que trabalham com *polinômios* com coeficientes no $GF(256)$, como: soma, multiplicação, escala, divisão, extração de raízes, cálculo de valor e derivada formal.

A base teórica dos códigos Reed-Solomon é dada pela seção 3.1. Já os fundamentos da álgebra abstrata são expostos no Apêndice A. A subseção 3.1.5.3 apresenta um exemplo detalhado, passo a passo da decodificação.

Em síntese, os passos do decodificador Reed-Solomon são:

1. São acrescentados 51 símbolos *zero* no início de $r(x)$, formando o código recebido alongado $r'(x)$, de 255 bytes;
2. A partir de $r'(x)$, calcula-se a síndrome $S(x)$;
3. Obtém-se o polinômio localizador de erros $\Lambda(x)$ através do algoritmo de *Berlekamp-Massey*;
4. Calcula-se as raízes de $\Lambda(x)$ (que são os inversos dos *errors locators* X_i^{-1});
5. Calcula-se o polinômio avaliador de erros $\Omega(x)$;
6. Calcula-se $\Lambda'(x)$, a derivada formal de $\Lambda(x)$;
7. Obtém-se a magnitude dos erros Y_i através do método de *Forney*;
8. Corrige-se $r'(x)$ nas ν posições dadas por X_i , através da soma das magnitudes dos erros Y_i com os símbolos incorretos de $r'(x)$. Com a remoção dos $2t$ bytes no final do bloco corrigido, obtém-se a mensagem recuperada $\hat{m}'(x)$;

⁴ Estas funções auxiliares fazem parte do componente de software `gf256`

9. Os primeiros 51 bytes (zeros) acrescentados no passo 1 são removidos de $\hat{m}'(x)$, obtendo-se enfim a mensagem recuperada $\hat{m}(x)$, de 188 bytes.

As etapas 1 e 9 executam a função de encurtamento do código, conforme detalhado na subseção 3.1.5.1. Os passos 2, 4 e 8 são executados conforme o exposto na subseção 3.1.4.1. A etapa 3 é determinada de acordo com a subseção 3.1.4.2. Já os passos 5, 6 e 7 seguem os procedimentos explicados na subseção 3.1.4.3.

Estas etapas consideram um decodificador RS com os parâmetros $n = 255$ e $k = 239$ – o RS(255, 239). É importante observar que, conforme ARIB (2014, p. 22), o primeiro expoente de α no polinômio gerador $g(x)$ é zero, ou seja, $b = 0$ na Equação 3.4.

Quando ocorrem mais de $t = 8$ erros em um bloco de 188 bytes (um pacote MPEG-2 TS), o decodificador RS não é capaz de corrigi-los, pois a quantidade de erros é superior a capacidade de correção do mesmo. Neste caso, o decodificador entrega em sua saída o mesmo pacote de sua entrada (exceto pela remoção dos bytes de paridade) e ativa o bit *transport error indicator* do pacote MPEG-2 TS de acordo a norma ISO/IEC-13818-1:2000 (ISO/IEC, 2000, p. 19).

O bit *transport error indicator* tem a função de sinalizar o *player* que aquele pacote MPEG-2 TS contém um ou mais símbolos incorretos. Assim, o *player* pode adotar alguma estratégia para minimizar as consequências negativas na mídia reproduzida (como artefatos no vídeo), como decidir por ignorar aquele pacote MPEG-2 TS ao invés de processá-lo.

O decodificador implementado mantém registros de estatísticas de correção de erros, tais como o número de blocos incorrigíveis, o número de blocos e bits processados e corrigidos. Estas informações podem ser consultadas em tempo real, para a verificação do desempenho do receptor.

5.2.2.16 Regenerador MPEG-2 TS

Bloco:	Regenerador MPEG-2 TS
Implementado pelo componente:	<code>tseditor</code>
Entrada(s):	$x[k]$: pacote/vetor MPEG-2 TS de 188 bytes (<code>int</code>)
Saída(s):	$y[k]$: pacote/vetor MPEG-2 TS corrigido, de 188 bytes (<code>int</code>)
Função equivalente no transmissor:	—

Conforme (ARIB, 2015, p. 7.29–7.30), a camada de recepção parcial não deve transmitir a PAT (*Program Association Table*) devido à limitação de largura de banda. Isto faz com que o feixe MPEG-2 TS não seja 100% compatível com a norma MPEG-2, que estabelece que a PAT é obrigatória (ISO/IEC, 2000, p. 8, 110).

A falta da PAT provoca incompatibilidade na reprodução. Constatou-se que os *players VLC* e *Mplayer* seguem a norma MPEG-2 estritamente, recusando a reproduzir o feixe. Verificou-se que um terceiro *player*, o *ffplay*, é mais relaxado em relação à norma, sendo capaz de reproduzir o feixe na maioria das vezes.

Para melhorar esta situação, implementou-se o bloco chamado de *Regenerador MPEG-2 TS*. Este bloco cria uma tabela PAT a partir de dados obtidos da PMT (*Program Map Table*) e a insere periodicamente no feixe MPEG-2 TS. Isto torna o feixe compatível com a norma e permite que outros *players*, que anteriormente recusavam a reproduzir o feixe, sejam capazes de reproduzi-lo corretamente.

5.3 Resultados

5.3.1 Emissoras sintonizadas

De acordo com o *Sistema de Informação dos Serviços de Comunicação de Massa - SISCO* da ANATEL, em Uberlândia atualmente existem 9 canais de televisão com transmissão digital (ANATEL, s. d.). Foram executadas tentativas de sintonia de cada um destes canais, utilizando o receptor desenvolvido com uma antena interna. Todos eles puderam ser sintonizados e decodificados com sucesso. Os parâmetros de cada canal sintonizado foram listados na Tabela 18.

Tabela 18: Canais sintonizados em Uberlândia com o receptor desenvolvido

Canal	Service provider/ name ^a	Modo/ IG	Parâmetros das camadas ^b	Taxa por camada [kbit/s]
17	TV BAND BAND TRIANGULO 1SEG	Modo 3 1/8	A: QPSK; 2/3; 383,04 ms; 1	A: 416,08
			B: 16QAM; 2/3; 191,52 ms; 12	B: 9986,04
			C: Não utilizada	C: 0
			Vídeo/Áudio da camada A:	
			Áudio: HE-AAC; 48000 Hz; stereo	
23	Rede Vida Rede Vida 1Seg	Modo 3 1/8	A: QPSK; 2/3; 383,04 ms; 1	A: 416,08
			B: 16QAM; 2/3; 191,52 ms; 12	B: 9986,04
			C: Não utilizada	C: 0
			Vídeo/Áudio da camada A:	
			Áudio: HE-AACv2; 48000 Hz; stereo	

A tabela continua na próxima página...

Tabela 18 – continuação da página anterior

Canal	Service provider/ name ^a	Modo/ IG	Parâmetros das camadas ^b	Taxa por camada [kbit/s]
27	TV Canção Nova	Modo 3	A: QPSK; 2/3; 383,04 ms; 1	A: 416,08
	Canção Nova 1SEG	1/8	B: 16QAM; 2/3; 383,04 ms; 12	B: 9986,04
			C: Não utilizada	C: 0
Vídeo/Áudio da camada A:			Vídeo: h264; 320x240; 29,97 fps	
			Áudio: HE-AAC; 32000 Hz; stereo	
28	RECORD	Modo 3	A: QPSK; 2/3; 383,04 ms; 1	A: 440,56
	TV PARANAIBA 1SEG	1/16	B: 16QAM; 2/3; 191,52 ms; 12	B: 10573,44
			C: Não utilizada	C: 0
Vídeo/Áudio da camada A:			Vídeo: h264; 320x240; 29,97 fps	
			Áudio: HE-AAC; 48000 Hz; stereo	
29	RCI	Modo 3	A: QPSK; 3/4; 95,76 ms; 1	A: 495,63
	RCI Móvel	1/16	B: 16QAM; 3/4; 95,76 ms; 12	B: 11895,12
			C: Não utilizada	C: 0
Vídeo/Áudio da camada A:			Vídeo: h264; 320x180; 29,97 fps	
			Áudio: HE-AAC; 48000 Hz; stereo	
30	Integra_B_1Seg	Modo 3	A: QPSK; 2/3; 383,04 ms; 1	A: 440,56
	INTEGRAÇÃO-1SEG	1/16	B: 64QAM; 3/4; 191,52 ms; 12	B: 17842,8
			C: Não utilizada	C: 0
Vídeo/Áudio da camada A:			Vídeo: h264; 320x180; 29,97 fps	
			Áudio: HE-AAC; 48000 Hz; stereo	
32	TV VITORIOSA	Modo 3	A: QPSK; 2/3; 383,04 ms; 1	A: 440,56
	TV VITORIOSA 1-SEG	1/16	B: 16QAM; 3/4; 383,04 ms; 12	B: 11895,12
			C: Não utilizada	C: 0
Vídeo/Áudio da camada A:			Vídeo: h264; 320x240; 29,97 fps	
			Áudio: HE-AACv2; 48000 Hz; stereo	

A tabela continua na próxima página...

Tabela 18 – continuação da página anterior

Canal	Service provider/ name ^a	Modo/ IG	Parâmetros das camadas ^b	Taxa por camada [kbit/s]
45	TV CÂMARA	Modo 3 1/16	A: QPSK; 2/3; 383,04 ms; 1	A: 440,56
	TV CAM 1SEG		B: 16QAM; 5/6; 191,52 ms; 12	B: 13216,8
			C: Não utilizada	C: 0
	Vídeo/Áudio da camada A:		Vídeo: h264; 320x180; 29,97 fps	
			Áudio: HE-AACv2; 48000 Hz; stereo	
49	RIT	Modo 3 1/8	A: QPSK; 2/3; 383,04 ms; 1	A: 416,08
	RIT-1Seg		B: 16QAM; 2/3; 191,52 ms; 12	B: 9986,04
			C: Não utilizada	C: 0
	Vídeo/Áudio da camada A:		Vídeo: h264; 320x240; 29,97 fps	
			Áudio: HE-AAC; 48000 Hz; stereo	

^a Nome da emissora e do serviço — obtidos dentro do feixe MPEG-2 TS contido na camada A

^b Os parâmetros são listados na ordem: modulação; taxa do codificador convolucional; profundidade do intercalador de tempo; número de segmentos

Fonte: o autor

É possível observar que todas as emissoras utilizam o modo 3 e transmitem apenas duas camadas, a A e a B, sendo a camada A utilizada para o sinal da recepção parcial, com 1 segmento, e a camada B com os 12 segmentos restantes destinados aos receptores *full-seg*.

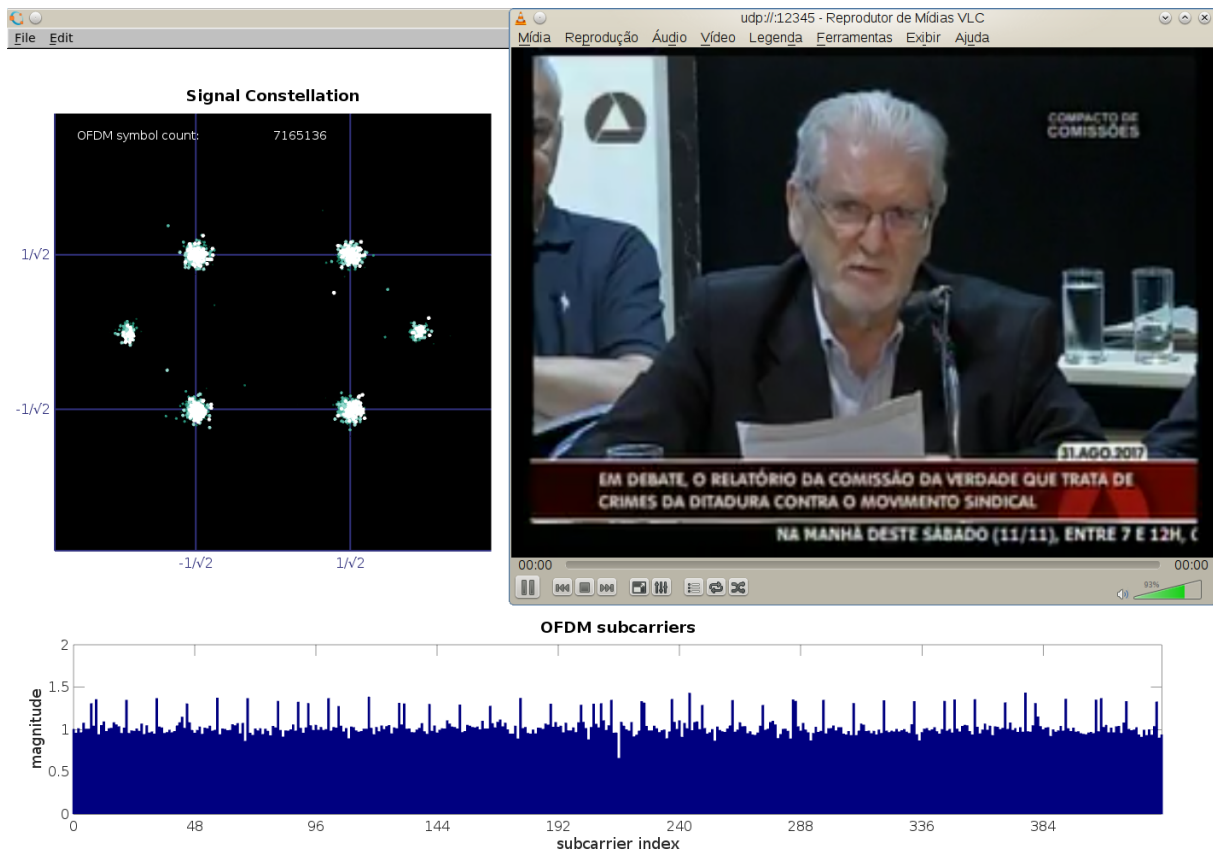
Todas as camadas A utilizaram modulação QPSK com algumas variações de configuração da taxa do codificador convolucional e da profundidade do intercalador de tempo. A taxa de transmissão de dados desta camada foi de 416 a 495 kbit/s.

Nos canais sintonizados, a camada B apresentou as modulações 16QAM e 64QAM, com diferentes taxas do codificador convolucional e profundidades do intercalador de tempo. A taxa de dados desta camada foi de 9986 a 17842 kbit/s.

A Figura 77 mostra o receptor em operação, sintonizado na emissora pública do canal 45. Nesta figura, a janela da frente é o *player* VLC, e a janela de fundo é uma ferramenta auxiliar de monitoração⁵ que exhibe, em tempo real, a constelação e o gráfico de magnitude das 432 portadoras do sinal após o equalizador. Na constelação é possível observar a sobreposição entre a modulação QPSK, utilizada nos dados, e a BPSK, empregada nos sinais pilotos e no TMCC.

⁵ Programa desenvolvido em GNU Octave que se conecta via *socket* ao software de decodificação ISDB-T através do componente `netcontrol`, conforme Figura 67

Figura 77: Receptor em operação e ferramenta de monitoração



Fonte: o autor

Alguns blocos do receptor são fundamentais para o seu desempenho em situações de sinal fraco ou degradado: (1) sincronizador de símbolo e frequência; (2) estimador da resposta do canal e equalizador; (3) decodificador Viterbi; e (4) decodificador Reed-Solomon. As próximas subseções discorrerão sobre o desempenho observado de cada um destes blocos.

5.3.2 Sincronizadores

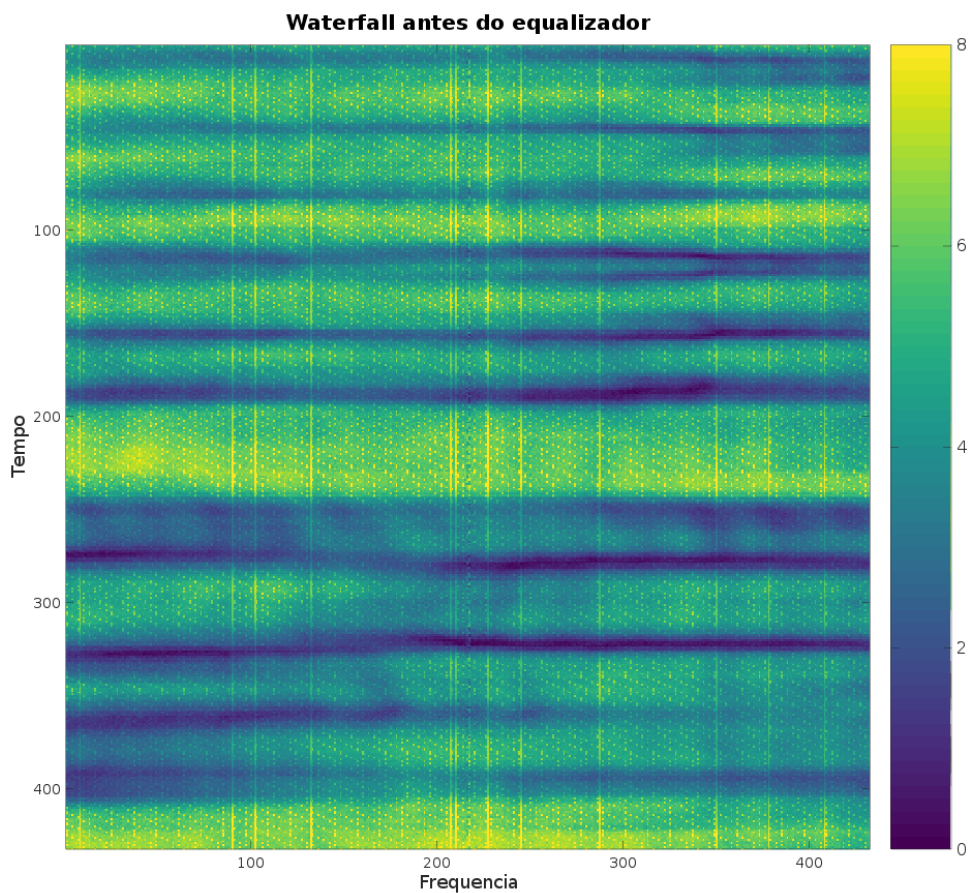
A sincronização de frequência é crítica em sistemas OFDM, onde erros de apenas algumas dezenas de Hertz provocam severas interferências entre subportadoras (*inter-carrier interference* - ICI). Somado a isto, o RTL-SDR de baixo custo possui um cristal com baixa estabilidade de frequência, que provoca desvios significativos da frequência sintonizada causada principalmente pela variação da temperatura (subseção 1.4.1.7). Mesmo neste contexto desafiador, o estimador de início de símbolo e do desvio de frequência em conjunto com os blocos corretores provaram ser robustos frente à estes problemas.

5.3.3 Equalizador

A interpolação linear utilizada no equalizador, apesar de simples, funcionou bem mesmo com o receptor operando com antena interna, onde o desvanecimento no domínio da frequência é claramente observado antes do equalizador.

A Figuras 78 e 79 mostram a magnitude do sinal antes e depois do equalizador, respectivamente, ao longo das subportadoras (eixo horizontal) e ao longo do tempo (eixo vertical) ao se movimentar a antena. É possível observar que o equalizador consegue manter a amplitude do sinal de cada portadora (QPSK neste exemplo) constante. As faixas verticais de maior intensidade são as portadoras fixas TMCC e AC, moduladas em (D)BPSK e transmitidas com magnitude $4/3$ das portadoras de dados (magnitude $\approx 33\%$ maior).

Figura 78: Magnitude do sinal antes do equalizador

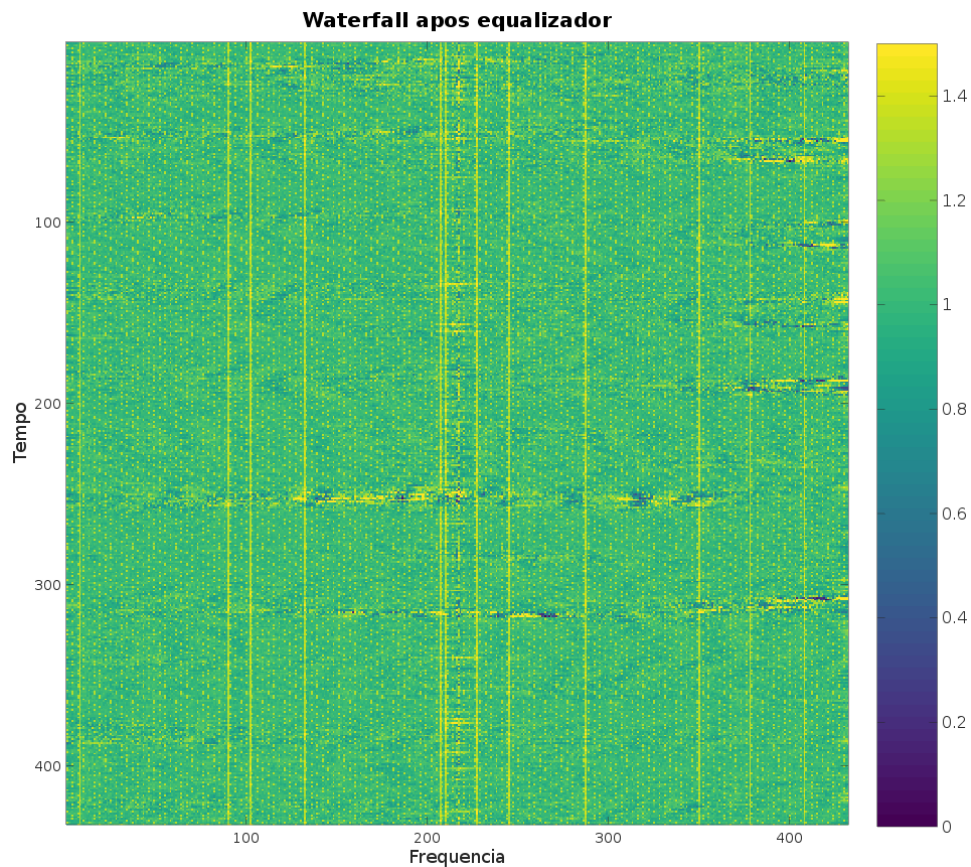


Fonte: o autor

5.3.4 Decodificador Viterbi

O desempenho do algoritmo de Viterbi implementado se mostrou equivalente ao da função *vitdec()* do Matlab.

Figura 79: Magnitude do sinal depois do equalizador



Fonte: o autor

A Figura 80 apresenta o resultado da comparação dos dois decodificadores, utilizando um código de taxa 2/3 e para diferentes quantizações (qbits) de entrada. As curvas do decodificador Viterbi desenvolvido são as curvas *ext viterbi*, pontilhadas e com marcadores. As curvas do decodificador de referência *vitdec()* do Matlab, são exibidas com linhas tracejadas, e coincidem exatamente com as do decodificador Viterbi desenvolvido.

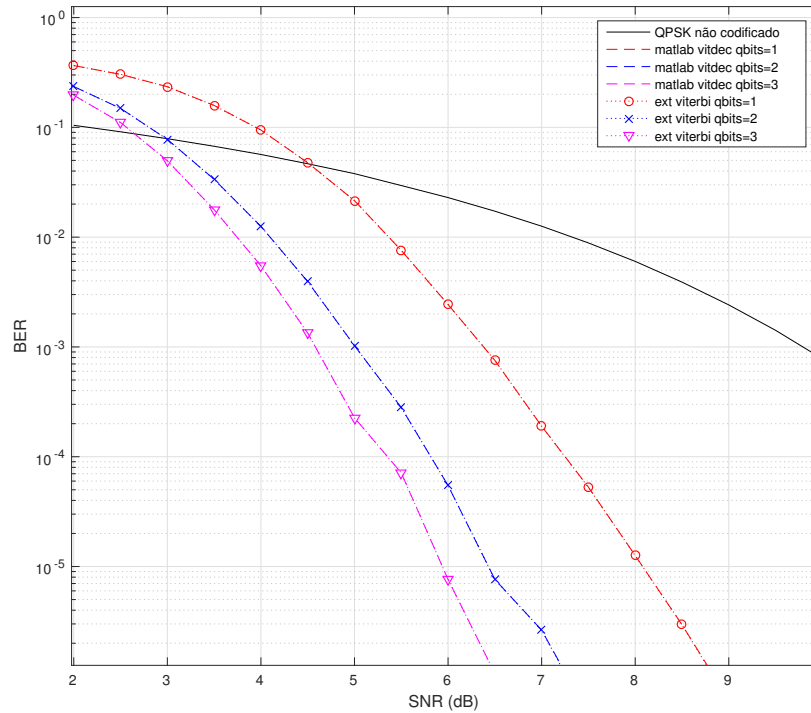
5.3.5 Decodificador Reed-Solomon

O decodificador Reed-Solomon foi testado de forma isolada de maneira abrangente, com diferentes padrões de erros na entrada. Em todos os casos testados, o decodificador se comportou como o esperado, corrigindo erros em até 8 bytes por bloco.

5.3.6 Velocidade de processamento

Em um laptop com processador Intel Core i7 3635QM 3,2 GHz (4 núcleos, 8 threads), o receptor ocupa 20% do tempo de uma das 8 threads de execução do processador

Figura 80: Desempenho do decodificador Viterbi comparado com o do Matlab



Fonte: o autor

para decodificar o sinal 1-seg em tempo real. O uso de memória RAM foi de 7,704 KB ⁶.

A tabela 19 apresenta o detalhamento do uso da CPU por cada bloco do receptor, relativos aos 20% de ocupação total. Este detalhamento foi obtido através da ferramenta *Callgrind*⁷, que coleta informações sobre o tempo de execução de cada função do software.

A dominância do uso de CPU pelo decodificador Viterbi pode ser explicada principalmente pela operação intensiva do algoritmo, no cálculo de $N_{MR} = 128$ (vide Equação 3.37 ⁸) métricas de ramo para cada bit de saída do decodificador Viterbi.

A taxa de bits na saída do decodificador Viterbi é $B_v = B/R_{RS}$, onde B é a taxa líquida de dados disponível na saída do receptor (ou, de forma equivalente, na entrada do transmissor, conforme Tabela 11), e R_{RS} é a taxa de codificação do codificador Reed-Solomon, igual a 188/204.

Considerando a recepção parcial (1-seg), B é tipicamente igual a 440 kbits/s

⁶ As ocupações de memória e CPU foram obtidas pela ferramenta *top*, disponível no sistema operacional Linux

⁷ <<http://valgrind.org/docs/manual/cl-manual.html>>

⁸ Parâmetros utilizados na Equação 3.37: $K = 7$ (comprimento de restrição do código); $P = 2^k = 2$ (número de estados de uma etapa anterior da treliça que são ligados a um estado de uma etapa posterior da treliça) ; $k = 1$ (numero de entradas do codificador convolucional).

⁹, portando, um valor típico para B_v é de 477 kbits/s. Deste modo, conclui-se que o decodificador Viterbi precisa calcular $N_{MR} \cdot B_v = 128 \cdot 477 \times 10^3 \approx 61$ milhões de métricas a cada segundo.

Uma possível solução para reduzir a utilização da CPU pelo algoritmo Viterbi seria a vetorização de parte do código, através do uso de instruções SIMD do processador.

Tabela 19: Uso relativo da CPU pelos blocos do receptor

Bloco	Uso da CPU
Decodificador Viterbi	82,73 %
Corretor de frequência	3,46 %
Demodulador	2,72 %
Desintercaladores	2,07 %
Equalizador	1,39 %
Decodificador Reed-Solomon	0,97 %
FFT	0,94 %
Reversor da dispersão	0,23 %
Regenerador do MPEG-2 TS	0,06 %
Decodificação TMCC	0,05 %
Demais funções	5,37 %
TOTAL	100,00 %

Fonte: o autor

⁹ Conforme a última coluna da Tabela 18, taxa de dados da camada A, recepção parcial 1-seg

6 Conclusão

Foram apresentados os detalhes da pesquisa e do desenvolvimento do receptor, e seus resultados foram expostos e discutidos, provando que um hardware RDS de baixo custo com atributos modestos como o RTL-SDR pode ser utilizado para desenvolver aplicações complexas como um receptor de TV digital.

O rádio definido por software, com a sua flexibilidade associada, redefiniu a forma como podemos pesquisar e ensinar sistemas de rádio. Apesar do crescente número de kits RDS disponíveis no mercado, o uso em massa desta tecnologia para uma classe de telecomunicações ainda é limitado, principalmente por causa de seu custo

Um sistema RDS de baixo custo pode ser um fator decisivo em lugares com recursos financeiros restritos, podendo possibilitar o ensino, aprendizado e pesquisa de um receptor digital relativamente complexo. Neste sentido, cada estudante em uma turma poderia ter acesso a um hardware RDS, podendo utilizá-lo até mesmo em casa para pesquisa.

Considera-se que, apesar da TV digital estar implantada há 10 anos no Brasil, a apropriação do conhecimento desta tecnologia pela comunidade acadêmica e científica no campo de engenharia de telecomunicações ainda é relevante, especialmente em universidades fora dos centros tecnológicos tradicionais. A relevância se estende para outros sistemas de telecomunicações modernos, muitos dos quais utilizam funções de rádio semelhantes ao do ISDB-T, como o OFDM e os códigos corretores de erro, técnicas de equalização e sincronização. Portanto, os conhecimentos adquiridos e difundidos neste trabalho não se limitam especificamente ao ISDB-T, e podem servir de base para outras pesquisas e desenvolvimentos.

Como sugestão de trabalhos futuros na mesma área, pode-se citar:

- Investigação sobre como reduzir a carga de trabalho da CPU no processamento do decodificador Viterbi, talvez através da vetorização utilizando instruções SIMD do processador
- Aprofundar pesquisa sobre o sincronizador de tempo e frequência, avaliando seu desempenho frente à perturbações do canal.
- Mais pesquisas sobre o equalizador, utilizando outros métodos de interpolação.
- Reescrita do código do receptor em C++, convertendo para blocos do GNU Radio.
- Obtenção, no decodificador Viterbi, de uma estimativa da relação sinal-ruído baseada na métrica de caminho acumulada.

- Cálculo do EVM (*Error Vector Magnitude*, ou magnitude do vetor de erro) no receptor.
- Estimativa do perfil de atraso do canal, a partir da autocorrelação do sinal recebido pelo receptor.
- Pesquisas de outros métodos para o decodificador Reed-Solomon, como o do domínio da frequência.
- Desenvolvimento de um software de processamento de sinais para agregar as faixas sintonizadas por múltiplos RTL-SDR em uma única banda de até 6 MHz. Seria necessária a modificações em seu hardware para sincronizá-los a partir de um único *clock*. Isto permitiria amostrar, por exemplo, todo o espectro ISDB-T (full-seg) a um custo ainda atrativo.
- Desenvolvimento de um transmissor ISDB-T em RDS para uso em conjunto com o receptor ISDB-T em RDS, para experimentações, modificações e proposições de melhorias.

Estendendo para outras áreas correlatas, há as seguintes sugestões:

- Utilizando a estrutura provida pelo receptor ISDB-T proposto, principalmente o OFDM e sua sincronização, desenvolver pesquisas práticas sobre o WiFi ou redes móveis LTE/5G.
- Investigar outros códigos corretores de erros mais poderosos, como o Turbo e o LDPC.

Referências

- ABNT. *NBR 15601*: Televisão digital terrestre - sistema de transmissão. Rio de Janeiro, 2007. 57 p. Citado 14 vezes nas páginas 104, 108, 112, 113, 114, 115, 116, 118, 119, 123, 124, 125, 128 e 141.
- ABRANTES, S. A. *Códigos Correctores de Erros em Comunicações Digitais*. Porto, Portugal: FEUP EDIÇÕES, 2010. 619 p. Citado na página 99.
- ANATEL. *SISCOM - Sistema de Informação dos Serviços de Comunicação de Massa*. s. d. Acessado em 10/11/2017. Disponível em: <<http://sistemas.anatel.gov.br/siscom/>>. Citado na página 148.
- ARIB. *STD-B31*: Transmission system for digital terrestrial television broadcasting. Tokyo, 2014. 180 p. Citado 15 vezes nas páginas 54, 85, 89, 90, 107, 108, 109, 110, 111, 117, 118, 123, 128, 141 e 147.
- ARIB. *Provisions for Carrier Operations*. [S.l.], 2015. Fascículo 5, Vol. 7, Revisão 6.0-E1. Citado na página 147.
- ARS TECHNICA. *How software-defined radio could revolutionize wireless*. 2012. Acessado em 27/10/2017. Disponível em: <<https://arstechnica.com/tech-policy/2012/07/how-software-defined-radio-could-revolutionize-wireless/>>. Citado na página 24.
- BEEK, J. Van de; SANDELL, M.; BÖRJESSON, P. ML estimation of time and frequency offset in OFDM systems. *IEEE Transactions on Signal Processing*, v. 45, n. 7, jul. 1997. DOI: <https://doi.org/10.1109/78.599949>. Citado 4 vezes nas páginas 38, 133, 134 e 135.
- BERLEKAMP, E. Nonbinary BCH decoding (abstr.). *IEEE Transactions on Information Theory*, Institute of Electrical and Electronics Engineers (IEEE), v. 14, n. 2, p. 242–242, mar 1968. DOI: <https://doi.org/10.1109/tit.1968.1054109>. Citado na página 59.
- BERLEKAMP, E. R. *Algebraic Coding Theory*. [S.l.]: World Scientific Publishing Co, 2015. Revised Edition. DOI: <https://doi.org/10.1142/9407>. Citado na página 71.
- BLAHUT, R. E. *Algebraic Codes for Data Transmission*. 2. ed. [S.l.]: Cambridge University Press, 2003. 498 p. DOI: <https://doi.org/10.1017/CBO9780511800467>. Citado 4 vezes nas páginas 65, 70, 72 e 85.
- BOSE, V. G. *Design and implementation of software radios using a general purpose processor*. 116 p. Tese (Ph.D.) — Massachusetts Institute of Technology. Dept. of Electrical Engineering and Computer Science, Cambridge, MA, 1999. Disponível em: <<http://hdl.handle.net/1721.1/9134>>. Citado na página 24.
- CAIN, J.; CLARK JR, G. C.; GEIST, J. Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding. *IEEE Transactions on Information Theory*, Institute of Electrical and Electronics Engineers (IEEE), v. 25, n. 1, p. 97–100, jan 1979. DOI: <https://doi.org/10.1109/tit.1979.1055999>. Citado na página 89.

- CASS, S. Hardware for your software radio. *IEEE Spectrum*, v. 43, n. 10, p. 51–54, out 2006. ISSN 0018-9235. DOI: <https://doi.org/10.1109/MSPEC.2006.1705782>. Citado na página 24.
- CASS, S. A \$40 software-defined radio. *IEEE Spectrum*, IEEE, v. 50, n. 7, p. 22–23, jul 2013. DOI: <https://doi.org/10.1109/mspec.2013.6545114>. Citado 2 vezes nas páginas 25 e 29.
- CLARK JR, G. C.; CAIN, J. B. *Error-Correction Coding for Digital Communications*. [S.l.]: Springer Science, 1981. 422 p. DOI: <https://doi.org/10.1007/978-1-4899-2174-1>. Citado 3 vezes nas páginas 85, 93 e 98.
- COSTELLO, D. J. et al. Applications of error-control coding. *IEEE Transactions on Information Theory*, v. 44, n. 6, p. 2531–2560, out. 1998. ISSN 0018-9448. DOI: <https://doi.org/10.1109/18.720548>. Citado na página 99.
- ELONICS. *E4000 Datasheet*. s. d. NooElec. Acessado em 26/07/2016. Disponível em: <https://www.nooelec.com/files/e4000datasheet.pdf>. Citado na página 34.
- ENCYCLOPEDIA BRITANNICA. *Evariste Galois*. 2017. Acessado em 08/08/2017. Disponível em: <https://www.britannica.com/biography/Evariste-Galois>. Citado na página 168.
- ETSI. *300 744*: Digital video broadcasting (DVB): Frame structure, channel coding and modulation for digital terrestrial television (DVB-T). Valbonne, França, 2009. 66 p. Disponível em: http://www.etsi.org/deliver/etsi_en/300700_300799/300744/01.06.01_60/en_300744v010601p.pdf. Citado 3 vezes nas páginas 46, 47 e 48.
- FFTW. *FFTW Home Page*. s. d. Acessado em 10/11/2016. Disponível em: <http://www.fftw.org/>. Citado na página 138.
- FORNEY, G. D. On decoding BCH codes. *IEEE Transactions on Information Theory*, Institute of Electrical and Electronics Engineers (IEEE), v. 11, n. 4, p. 549–557, oct 1965. DOI: <https://doi.org/10.1109/tit.1965.1053825>. Citado na página 72.
- FORNEY, G. D. The Viterbi algorithm. *Proceedings of the IEEE*, v. 61, n. 3, p. 268–278, mar. 1973. ISSN 0018-9219. DOI: <https://doi.org/10.1109/PROC.1973.9030>. Citado na página 90.
- FORNEY, G. D. The Viterbi Algorithm: A personal history. *The Computing Research Repository (CoRR)*, abs/cs/0504020, 2005. Disponível em: <http://arxiv.org/abs/cs/0504020>. Citado na página 90.
- FRIGO, M. A fast fourier transform compiler. In: *Proceedings of the ACM SIGPLAN 1999 Conference on Programming Language Design and Implementation*. New York, NY, USA: ACM, 1999. (PLDI '99), p. 169–180. ISBN 1-58113-094-5. DOI: <https://doi.org/10.1145/301618.301661>. Citado na página 138.
- GRAYVER, E. *Implementing Software Defined Radio*. Nova Iorque: Springer, 2013. 270 p. DOI: <https://doi.org/10.1007/978-1-4419-9332-8>. Citado na página 22.
- HANZO, L. L.; KELLER, T. *OFDM and MC-CDMA: A Primer*. Chichester, Inglaterra: Wiley-IEEE Press, 2006. 411 p. DOI: <https://doi.org/10.1002/9780470031384>. Citado na página 53.

- HAYKIN, S. *Digital communication systems*. Hoboken, NJ: J. Wiley & Sons, 2014. ISBN 978-0471647355. Citado 3 vezes nas páginas 63, 65 e 85.
- IEDEMA, M. *Getting started with OpenBTS: build open source mobile networks*. Sebastopol, Califórnia: O'Reilly Media, 2015. 122 p. Citado na página 25.
- ISO/IEC. *13818-1:2000 (E)*: Information technology - generic coding of moving pictures and associated audio information: Systems. Genebra, Suíça, 2000. 154 p. Citado 2 vezes nas páginas 108 e 147.
- ITU-R. *Report ITU-R BT.2035-2*: Guidelines and techniques for the evaluation of digital terrestrial television broadcasting systems including assessment of their coverage areas. [S.l.], 2008. Disponível em: <https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-BT.2035-2-2008-PDF-E.pdf>. Citado na página 46.
- JOHANNESSON, R.; ZIGANGIROV, K. S. *Fundamentals of Convolutional Coding*. 1. ed. [S.l.]: Wiley-IEEE Press, 1999. 442 p. DOI: <https://doi.org/10.1109/9780470544693>. Citado 2 vezes nas páginas 85 e 100.
- KEEN, K. *Rtl Power: Basic scripting*. 2017. Acessado em 10/11/2017. Disponível em: <<http://kmkeen.com/rtl-power/>>. Citado na página 37.
- KUMAGAI, J. Radio revolutionaries. *IEEE Spectrum*, v. 44, n. 1, p. 28–32, jan 2007. ISSN 0018-9235. DOI: <https://doi.org/10.1109/MSPEC.2007.273037>. Citado na página 24.
- LACKEY, R. I.; UPMAL, D. W. Speakeasy: the military software radio. *IEEE Communications Magazine*, v. 33, n. 5, p. 56–61, mai 1995. ISSN 0163-6804. DOI: <https://doi.org/10.1109/35.392998>. Citado na página 24.
- LATHI, B. P.; DING, Z. *Modern Digital and Analog Communication Systems*. 4. ed. Nova Iorque: Oxford University Press, 2009. 1004 p. Citado 2 vezes nas páginas 53 e 54.
- LIDL, R. *Introduction to finite fields and their applications*. [S.l.]: Cambridge University Press, 1986. 416 p. Citado na página 165.
- LIN, S.; COSTELLO, D. J. *Error Control Coding*. 2. ed. [S.l.]: Pearson, 2004. 1272 p. Citado 7 vezes nas páginas 71, 72, 73, 85, 93, 98 e 165.
- MASSEY, J. Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, Institute of Electrical and Electronics Engineers (IEEE), v. 15, n. 1, p. 122–127, jan 1969. DOI: <https://doi.org/10.1109/tit.1969.1054260>. Citado 2 vezes nas páginas 59 e 70.
- MINISTÉRIO DAS COMUNICAÇÕES. *Portaria 625*. 2013. Disponível em: <<http://www.anatel.gov.br/legislacao/resolucoes/2013/644-resolucao-625>>. Citado na página 104.
- MINISTÉRIO DAS COMUNICAÇÕES. *Portaria 378*. 2016. Disponível em: <<http://pesquisa.in.gov.br/imprensa/jsp/visualiza/index.jsp?data=25/01/2016&jornal=1&pagina=66&totalArquivos=88>>. Citado na página 101.
- MITOLA, J. Software radios-survey, critical evaluation and future directions. In: *NTC-92: National Telesystems Conference*. [S.l.: s.n.], 1992. p. 13/15–13/23. DOI: <https://doi.org/10.1109/NTC.1992.267870>. Citado na página 24.

MITOLA, J. The software radio architecture. *IEEE Communications Magazine*, v. 33, n. 5, p. 26–38, mai 1995. ISSN 0163-6804. DOI: <https://doi.org/10.1109/35.393001>. Citado na página 24.

MOBILE EXPERTS. *SDR Market Size Study*. 2011. Wireless Innovation Forum. Acessado em 28/10/2017. Disponível em: <http://www.wirelessinnovation.org/assets/documents/mexp-sdr-11%20final.pdf>. Citado na página 23.

MOON, T. K. *Error Correction Coding: Mathematical Methods and Algorithms*. [S.l.]: Wiley-Interscience, 2005. 800 p. DOI: <https://doi.org/10.1002/0471739219>. Citado 2 vezes nas páginas 71 e 72.

ONYSZCHUK, I. M. Coding gains and error rates from the Big Viterbi Decoder. In: Posner, E. C. (Ed.). *The Telecommunications and Data Acquisition Report*. [s.n.], 1991. p. 170–174. Disponível em: <http://adsabs.harvard.edu/abs/1991tdar.nasa..170O>. Citado na página 99.

OSMOCOM. *RTL-SDR*. 2014. Acessado em 22/08/2016. Disponível em: <http://sdr.osmocom.org/trac/wiki/rtl-sdr>. Citado 2 vezes nas páginas 29 e 31.

OSMOCOM. *Código fonte do arquivo librtlsdr.c*. 2017. Acessado em 10/11/2016. Disponível em: <http://git.osmocom.org/rtl-sdr/tree/src/librtlsdr.c>. Citado na página 35.

PALICOT, J. (Ed.). *Radio engineering: From software to cognitive radio*. Londres: ISTE Wiley, 2011. 378 p. DOI: <https://doi.org/10.1002/9781118602218>. Citado na página 27.

PINTER, C. C. *A book of abstract algebra*. [S.l.]: McGraw-Hill, 1982. 351 p. Citado 2 vezes nas páginas 165 e 172.

PRESIDÊNCIA DA REPÚBLICA. *Decreto 5820*. 2006. Disponível em: http://www.planalto.gov.br/ccivil_03/_Ato2004-2006/2006/Decreto/D5820.htm. Citado na página 101.

PROAKIS, J.; SALEHI, M. *Digital communications*. 5. ed. Boston: McGraw-Hill, 2008. 1150 p. Citado 3 vezes nas páginas 87, 89 e 100.

RAFAEL MICROELECTRONICS. *R820T Datasheet*. s. d. DatasheetsPDF.com. Acessado em 26/07/2016. Disponível em: <http://www.datasheetspdf.com/datasheet/R820T.html>. Citado na página 35.

RAPPAPORT, T. S. *Wireless communications principles and practice*. 2. ed. Boston: Prentice Hall, 2002. 736 p. Citado 2 vezes nas páginas 47 e 49.

REALTEK. *RTL2832U*. s. d. Acessado em 21/03/2017. Disponível em: <http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257>. Citado na página 35.

REED, I. S.; SOLOMON, G. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, Society for Industrial and Applied Mathematics, v. 8, n. 2, p. 300–304, 1960. DOI: <https://doi.org/10.1137/0108018>. Citado 3 vezes nas páginas 59, 60 e 61.

- RTL-SDR.COM. *About RTL-SDR*. 2016. Acessado em 22/08/2016. Disponível em: <<http://www.rtl-sdr.com/about-rtl-sdr>>. Citado na página 31.
- RTLSDR.ORG. *History and Discovery of RTLSDR*. 2013. Acessado em 19/03/2016. Disponível em: <http://www.rtlsdr.org/#history_and_discovery_of_rtlsdr>. Citado 2 vezes nas páginas 25 e 29.
- RUTGERS UNIVERSITY SCHOOL OF ENGINEERING. *Software-Defined Radio: A Hands-On Approach*. s. d. Acessado em 01/11/2017. Disponível em: <<http://www.winlab.rutgers.edu/~spasojev/courses/sdr/resources.html>>. Citado na página 42.
- SAMRA, H. *The OpenBTS project - an open-source GSM base station*. 2008. Linux Weekly News - LWN.net. Acessado em 28/10/2017. Disponível em: <<https://lwn.net/Articles/296949/>>. Citado na página 25.
- SDR FORUM. *SDRF Cognitive Radio Definitions*. 2007. Working Document SDRF-06-R-0011-V1.0.0. Acessado em 24/10/2017. Disponível em: <http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf>. Citado na página 20.
- SERGIENKO, A. B. Software-defined radio in MATLAB simulink with RTL-SDR hardware. In: *2014 International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA)*. [S.l.]: IEEE, 2014. p. 160–161. DOI: <https://doi.org/10.1109/ICCTPEA.2014.6893337>. Citado 2 vezes nas páginas 25 e 42.
- SKLAR, B. Rayleigh fading channels in mobile digital communication systems - part 1: Characterization. *IEEE Communications Magazine*, IEEE, v. 35, n. 7, p. 90–100, jul 1997. DOI: <https://doi.org/10.1109/35.601747>. Citado na página 47.
- SKLAR, B. *Digital Communications: Fundamentals and Applications*. 2. ed. Upper Saddle River, N.J: Prentice Hall, 2001. 1079 p. Citado 3 vezes nas páginas 45, 87 e 98.
- SKLIVANITIS, G. et al. Addressing next-generation wireless challenges with commercial software-defined radio platforms. *IEEE Communications Magazine*, v. 54, n. 1, p. 59–67, jan 2016. ISSN 0163-6804. DOI: <https://doi.org/10.1109/MCOM.2016.7378427>. Citado na página 26.
- STANFORD UNIVERSITY. *EE 387 - Algebraic Error Control Codes*. 2015. Notas de aula. Acessado em 20/09/2017. Disponível em: <<https://web.stanford.edu/class/ee387/handouts/notes20.pdf>>. Citado na página 71.
- STEWART, R. W. et al. *Software Defined Radio using MATLAB & Simulink and the RTL-SDR*. Glasgow, Escócia: Strathclyde Academic Media, 2015. 674 p. Citado 2 vezes nas páginas 20 e 29.
- TOSHI. *Diagrama do circuito do receptor DVB-T+DAB+FM baseado nos chips RTL-2832U e R820T (tradução nossa)*. 2014. Blog. Acessado em 31/10/2017. Disponível em: <http://ggtoshi.at.webry.info/201406/article_6.html>. Citado 3 vezes nas páginas 30, 32 e 33.
- VITERBI, A. J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, v. 13, n. 2, p. 260–269, abr. 1967. ISSN 0018-9448. DOI: <https://doi.org/10.1109/TIT.1967.1054010>. Citado na página 90.

- VITERBI, A. J.; OMURA, J. K. *Principles of Digital Communication and Coding*. [S.l.]: McGraw-Hill College, 1979. 576 p. Citado na página 85.
- WELCH, L. *The Original View of Reed-Solomon Codes*. 1997. Acessado em 14/09/2017. Disponível em: <<http://csi.usc.edu/PDF/RSooriginal.pdf>>. Citado na página 62.
- WICKER, S. B. *Error Control Systems for Digital Communication and Storage*. [S.l.]: Prentice-Hall, 1994. 512 p. Citado 3 vezes nas páginas 70, 71 e 165.
- WICKER, S. B.; BHARGAVA, V. K. (Ed.). *Reed-Solomon Codes and Their Applications*. [S.l.]: Wiley-IEEE Press, 1999. 336 p. DOI: <https://doi.org/10.1109/9780470546345>. Citado 4 vezes nas páginas 59, 62, 65 e 71.
- WIRED MAGAZINE. *GNU Radio opens an unseen world*. 2006. Acessado em 27/10/2017. Disponível em: <<https://www.wired.com/2006/06/gnu-radio-opens-an-unseen-world/>>. Citado 2 vezes nas páginas 17 e 24.

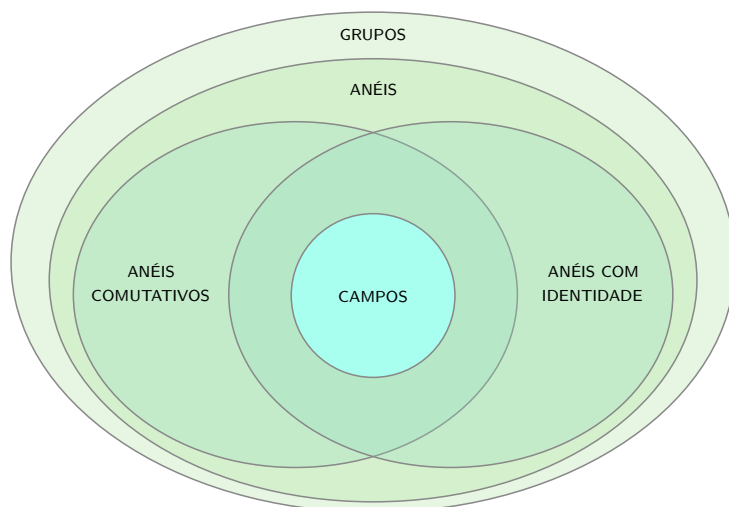
Apêndices

APÊNDICE A – Álgebra abstrata para os códigos Reed-Solomon

Os campos, assim como os anéis e grupos pertencem a uma área importante da álgebra abstrata. Sua base teórica é relativamente complicada para não matemáticos. Lin e Costello (2004, cap. 2) e Wicker (1994, cap. 2 e 3) fornecem uma introdução do tema direcionada para o uso em códigos corretores de erros, enquanto Lidl (1986) e Pinter (1982) aprofundam o tema com a devida formalidade matemática.

Esta seção e suas subseções versarão sobre os conceitos básicos de Grupos, Anéis e Campos necessários para o entendimento do codificador e decodificador Reed-Solomon utilizados nos transmissores e receptores ISDB-T. As relações de algumas estruturas algébricas importantes discutidas nesta seção são mostradas na Figura 81.

Figura 81: Diagrama de Venn para algumas estruturas algébricas



Fonte: o autor

A.1 Grupos

Um grupo $\langle G, * \rangle$ é uma estrutura algébrica composta por um *conjunto* G equipado com uma *operação binária* genérica representada pelo símbolo $*$, que obedece aos seguintes axiomas:

1. Associatividade: $\forall a, b, c \in G: (a * b) * c = a * (b * c)$

2. Elemento identidade: $\forall a \in G, \exists e \in G: a * e = e * a = a$
3. Inverso: $\forall a \in G, \exists a' \in G: a * a' = e$

Um grupo que além destes três axiomas obedece o axioma da *comutatividade*, descrito abaixo, é chamado de *grupo comutativo* ou *grupo abeliano*.

4. Comutatividade: $\forall a, b \in G: a * b = b * a$

A operação binária $*$ faz o mapeamento de quaisquer pares de elementos $a, b \in G$ em um terceiro elemento $c \in G$, na forma $a * b = c$. Como estamos tratando de álgebra abstrata, é importante notar que a operação $*$ é genérica, e dependendo de como o grupo é definido, $*$ pode ser, por exemplo, a operação de adição, multiplicação, composição de funções ou uma transformação geométrica.

O elemento a' é o elemento inverso de a , e dependendo do contexto, pode ser representado por $-a$ (normalmente quando a operação $*$ é a adição) ou a^{-1} (geralmente quando $*$ é a multiplicação). O elemento e é chamado de identidade de G .

O conjunto dos números inteiros sobre a operação de adição forma um grupo $\langle \mathbb{Z}, + \rangle$. Neste caso o elemento identidade é o 0, e o elemento inverso de a é o $-a$. Já o mesmo conjunto dos inteiros sobre a operação de multiplicação não forma um grupo $\langle \mathbb{Z}, \cdot \rangle$, pois, apesar de haver o elemento identidade da multiplicação (1), não existe elementos inversos da multiplicação para cada elemento de \mathbb{Z} .

Outros exemplos de grupos sobre a operação de adição são: o grupo $\langle \mathbb{R}, + \rangle$ formado pelos números reais; o grupo $\langle \mathbb{Q}, + \rangle$ formado pelos números racionais; o grupo $\langle \mathbb{C}, + \rangle$ formado pelos números complexos. Sobre a operação de multiplicação, temos como exemplo o grupo $\langle \mathbb{Q} \setminus \{0\}, \cdot \rangle$ formado pelos números racionais exceto o zero.

A *ordem* de um grupo é a cardinalidade (quantidade de elementos) de seu conjunto associado. Uma maneira de se construir grupos de ordem finita m , é utilizar a *redução módulo m* ¹ logo após a operação do grupo contendo um número finito de elementos em seu conjunto. Neste caso, a operação de adição $a + b$ é calculada através da adição usual seguida da operação módulo m : $(a + b) \bmod m$. A multiplicação $a \times b$ ocorre de forma análoga: $(a \times b) \bmod m$.

É provado que um conjunto com m elementos $\{0, 1, 2, \dots, m-1\} \in \mathbb{Z}$ forma um grupo (finito) sobre a operação adição *módulo m* para qualquer inteiro positivo m . Este grupo é representado por $\langle \mathbb{Z}_m, + \rangle$, seu elemento identidade é o 0, e o elemento inverso de a é $m - a \pmod{m}$.

¹ O *módulo* é uma operação binária, geralmente escrita na forma $r = a \bmod b$, que calcula r como sendo o resto da divisão inteira $a \div b$, de forma que $a = b \cdot q + r$.

Considerando agora a operação multiplicação *módulo* m , é provado que o grupo só existe se m for um inteiro primo p e se o conjunto associado não contiver o elemento 0: $\{1, 2, \dots, p-1\}$. Este grupo é geralmente representado por $\langle \mathbb{Z}_p \setminus \{0\}, \cdot \rangle$.

A.2 Anéis

Um *anel* $\langle R, +, \cdot \rangle$ é uma estrutura algébrica que estende a definição de grupo. Um anel é composto de um conjunto R munido de duas operações binárias $+$ e \cdot que obedecem aos seguintes axiomas:

1. $\langle R, + \rangle$ é um grupo comutativo. O elemento identidade da adição é o 0
2. A operação \cdot é associativa: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
3. A operação \cdot é distributiva sobre a $+$: $a \cdot (b + c) = a \cdot b + a \cdot c$

Um anel que adicionalmente obedece o axioma da *comutatividade*, descrito abaixo, é chamado de *anel comutativo*.

4. Comutatividade da operação \cdot : $a \cdot b = b \cdot a$

Já um anel que obedece ao axioma abaixo, é chamado de *anel com identidade*.

5. O operador \cdot possui o elemento identidade 1

Um anel que obedece a todos os axiomas acima, é chamado de *anel comutativo com identidade*.

O conjunto das matrizes $M_{2 \times 2}$ com elementos inteiros forma um anel não comutativo com identidade. Neste caso, $\mathbf{A} \cdot \mathbf{B} \neq \mathbf{B} \cdot \mathbf{A}$ e o elemento identidade da multiplicação é a matriz identidade $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Se o conjunto de matrizes for redefinido considerando que os elementos das matrizes sejam inteiros pares, esta nova estrutura forma um anel não comutativo sem identidade.

Outro exemplo é o conjunto dos inteiros pares $2\mathbb{Z} = \{\dots, -4, -2, 0, 2, 4, \dots\}$ que forma um anel comutativo sem identidade, pois não possui a identidade da operação multiplicação (1). Já o conjunto dos números inteiros \mathbb{Z} forma um anel comutativo com identidade. Outro exemplo é o conjunto de todos os polinômios com coeficientes obtidos de um anel ou *campo* K , isto forma um *anel de polinômios* $K[x]$.

O anel de polinômios cujos coeficientes pertencem a um campo binário (0 ou 1), denotado por $F_2[x]$ ou $GF(2)[x]$, é uma estrutura importante para os códigos corretores de erros, e será explorada nas seções posteriores, assim como a definição de campo.

A.3 Campos

Um campo $\langle F, +, \cdot \rangle$ é outra estrutura algébrica composta por um conjunto F e equipado com duas operações binárias: a soma (+) e a multiplicação (\cdot), e deve obedecer aos seguintes axiomas:

1. $\langle F, + \rangle$ é um grupo comutativo. O elemento identidade da adição é o 0, e o elemento inverso é denotado por $-a$
2. $\langle F \setminus \{0\}, \cdot \rangle$ é um grupo comutativo ($F \setminus \{0\}$ significa o conjunto F sem o elemento 0). O elemento identidade da multiplicação é o 1, e o elemento inverso é denotado por a^{-1}
3. A operação \cdot é distributiva sobre a $+$: $a \cdot (b + c) = a \cdot b + a \cdot c$

A definição acima é baseada nos grupos comutativos. De forma alternativa, é possível afirmar que um campo é um anel comutativo com identidade, cuja operação de multiplicação possui inversos para todos os seus elementos, exceto o zero.

A subtração $a - b$ em um campo é definida como a adição de a com o elemento aditivo inverso de b : $a + (-b)$. Já a divisão $a \div b$ ocorre através da multiplicação de a pelo elemento multiplicativo inverso de b : $a \cdot b^{-1}$.

O conjunto dos números complexos \mathbb{C} , dos números reais \mathbb{R} e dos números racionais \mathbb{Q} , junto com suas operações usuais de adição e multiplicação, são exemplos de campos contendo um número infinito de elementos. Já o conjunto dos números inteiros \mathbb{Z} não formam um campo, pois seus elementos não possuem inverso multiplicativo definidos dentro do próprio conjunto \mathbb{Z} , por exemplo, $2^{-1} \notin \mathbb{Z}$.

A.4 Campos Finitos $GF(p)$

Os Campos Finitos ou Campos de Galois possuem aplicações importantes nas áreas da criptografia e dos códigos corretores de erros. O nome *Galois* é uma homenagem a Évariste Galois, um matemático francês que desenvolveu seu breve trabalho na primeira metade do século 19, e apesar de morrer jovem, deixou importantes contribuições para a álgebra abstrata (ENCYCLOPEDIA BRITANNICA, 2017).

Um campo finito, denotado por F_q ou $GF(q)$ – GF de *Galois Field* – é um campo construído com um conjunto finito de q elementos. As operações de adição, subtração, multiplicação, divisão são todas definidas no F_q e têm como resultado elementos do próprio conjunto original de q elementos, de acordo como os axiomas dos campos.

Assim como nos grupos, o número de elementos q de um campo finito é conhecido como a *ordem do campo*.

A ordem de um elemento β de um campo (não relacionado com a ordem do campo em si), é definida como sendo o menor inteiro positivo m tal que $\beta^m = 1$. Um elemento que possui ordem $q - 1$ é chamado de *elemento primitivo* do campo.

Todos os elementos de um campo $GF(q)$, com a exceção do elemento zero, podem ser representados, ou gerados, pelas $q - 1$ potências consecutivas do elemento primitivo α : $\{\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{q-2}\}$. Por isso, o elemento primitivo α é também conhecido como o *elemento gerador* de $GF(q)$.

A característica de um campo $GF(q)$ é definida como sendo o menor inteiro positivo m tal que $\sum_1^m a = 0$, $\forall a \neq 0 \in GF(q)$. A característica de um campo finito $GF(q)$ é sempre um inteiro primo.

É provado que só existem campos finitos $GF(q)$ de ordem $q = p^n$, sendo p um inteiro primo e n um inteiro positivo. Para o caso particular mais simples $n = 1$, temos o campo $GF(p)$ de ordem p , chamado de *campo primo*. É possível construir um campo primo com um conjunto de p inteiros $\{0, 1, \dots, p - 1\}$ sob a adição e a multiplicação *módulo* p .

Considere como exemplo um campo finito (primo) de ordem 3 composto pelo conjunto $\{0, 1, 2\}$, com as operações de adição e multiplicação módulo 3. As tabelas 20 e 21 mostram todas as operações de adição e multiplicação possíveis neste campo. Nestas tabelas é fácil identificar o elemento identidade da adição (0) e da multiplicação (1) através da sua linha ou coluna que resultam nos mesmos elementos do cabeçalho.

Os elementos inversos de cada operação também podem ser identificados nas mesmas tabelas, pois são os que resultam nos elementos identidades de cada operação, por exemplo: $2^{-1} = 2$ (o inverso multiplicativo do elemento 2 é o próprio elemento 2); e $-1 = 2$ (o inverso aditivo do elemento 1 é o elemento 2).

 Tabela 20: Tabela adição para $GF(3)$

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

 Tabela 21: Tabela multiplicação para $GF(3)$

\times	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Os campos finitos do tipo $GF(2^n)$, com $n \geq 1$, são os mais utilizados nos códigos corretores de erro, devido a conveniência da quantidade de elementos ser múltipla de 2, em linha com os sistemas digitais que utilizam o bit como unidade básica de informação.

Por exemplo, o campo finito $GF(2)$ composto pelo conjunto $\{0, 1\}$, com as operações de adição e multiplicação módulo 2, possui todas as suas operações definidas pelas tabelas 22 e 23. É possível notar que a operação adição pode ser implementada através do operador lógico *OU-exclusivo*, e a multiplicação pelo operador lógico *E*. Outra propriedade importante do $GF(2)$, é que a subtração e soma são a mesma operação, pois, $\forall a \in GF(2)$, $a + a = 0$ e $a = -a$.

 Tabela 22: Tabela adição para $GF(2)$

+	0	1
0	0	1
1	1	0

 Tabela 23: Tabela multiplicação para $GF(2)$

\times	0	1
0	0	0
1	0	1

A tentativa de se construir um campo $GF(p^n)$ com $n > 1$ pelo método direto explorado aqui para os campos primos é infrutífera. Por exemplo, a Tabela 24 mostra todas as multiplicações possíveis na tentativa de se construir um campo $GF(2^2)$ com os elementos $\{0, 1, 2, 3\}$ e as operações de adição e multiplicação módulo 4. Observa-se que o elemento 2 não possui inverso (na linha ou coluna do elemento 2, não há o elemento identidade 1 como resultado). Portanto, da forma como foi definido aqui, $GF(2^2)$ não forma um campo.

 Tabela 24: Tabela multiplicação para a tentativa de construção do $GF(2^2)$

\times	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

O procedimento correto para se gerar um campo $GF(2^2)$ é dado na subseção A.5.3.

A.5 Campos Finitos $GF(p^n)$

Um subcampo F de um campo K é um subconjunto de K que também é um campo. Mas aqui, a abordagem inversa é mais importante: o campo K é uma *extensão* do campo F . Neste caso, F é chamado de *campo base*. Por exemplo, o campo dos números reais \mathbb{R} é um subcampo dos números complexos \mathbb{C} , portanto \mathbb{C} é dito uma extensão de seu campo base \mathbb{R} .

O conceito de extensão de campos nos dá subsídios para descrever todos os campos de ordem finita, particularmente aqueles da forma $F_q = GF(p^n)$ com p primo e $n > 1$. Tal campo é construído através da extensão do campo base $F_p = GF(p)$ para n dimensões, utilizando polinômios com coeficientes do campo base.

A.5.1 Polinômios sobre $GF(p)$

Considere o polinômio $p(x)$ da Equação A.1. Este polinômio possui grau $n = \deg(p(x))$, e seus coeficientes $a_0, a_1, a_2, \dots, a_n$ são elementos de um campo base primo F_p .

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n \quad (\text{A.1})$$

O conjunto de todos os polinômios de todos os graus, com coeficientes do campo F_p , forma um anel comutativo com identidade, chamado anel de polinômios $F_p[x]$. Este anel possui infinitos elementos (polinômios), e as operações de soma, subtração, multiplicação e divisão (com resto) podem ser feitas através da aritmética do campo base.

A soma ou subtração entre dois polinômios $a(x), b(x) \in F_p[x]$ é calculada da maneira usual, porém as operações entre seus coeficientes devem obedecer as do campo base F_p . O polinômio resultante desta operação possuirá no máximo o maior grau entre $a(x)$ e $b(x)$.

Já a multiplicação entre dois polinômios $a(x), b(x) \in F_p[x]$ produz como resultado um polinômio $c(x)$, cujo grau é $\deg(c(x)) = \deg(a(x)) + \deg(b(x))$.

A divisão $a(x)/b(x)$ entre dois polinômios $a(x), b(x) \in F_p[x]$, com $b(x) \neq 0$, é calculada através da divisão Euclideana, $a(x) = b(x) \cdot q(x) + r(x)$, que produz o quociente $q(x)$ e resto $r(x)$ únicos. Assim como para os números inteiros, o operador *módulo* pode ser utilizado para representar o cálculo do resto da divisão entre $a(x)$ e $b(x)$: $r(x) = a(x) \bmod b(x)$. Temos que $\deg(r(x)) < \deg(b(x))$, ou $r(x) = 0$.

A.5.2 Polinômio irredutível

Um polinômio $p(x)$ cujos coeficientes pertencem ao campo F é dito *irredutível em F* quando ele não puder ser fatorado, ou seja, se não houver outros polinômios $a(x), b(x) \in F[x]$ que permitam $p(x) = a(x) \cdot b(x)$. As fatorações triviais (polinômios constantes) são desconsideradas. O papel dos polinômios irredutíveis nos campos é análogo ao dos números primos.

É importante salientar que um polinômio pode ser irredutível em um campo mas redutível em outro. Por exemplo, $p(x) = x^2 + 1$ é irredutível no campo dos números reais \mathbb{R} , mas é redutível no campo dos números complexos \mathbb{C} , onde seus fatores são $(x + i)(x - i)$,

com $i = \sqrt{-1}$. Por isso, é sempre necessário informar sobre qual campo o referido $p(x)$ é dito irredutível.

A.5.3 Extensão de campos

Quando as operações de soma e multiplicação entre polinômios de $F_p[x]$ são reduzidas (através do operador módulo) por um polinômio $p(x)$ de grau n , irredutível em F_p , é provado que forma-se um campo. Os elementos deste campo são polinômios de grau $\leq n-1$. Portanto, o campo formado é finito, e possui $q = p^n$ elementos representados por polinômios com n coeficientes $a_0, a_1, a_2, \dots, a_{n-1}$, do tipo $a(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$.

Por exemplo, a Tabela 25 lista todos os elementos do campo $F_4 = GF(2^2)$ na forma polinomial e na forma decimal. Como os coeficientes $[a_1a_0] \in GF(2)$ dos polinômios (elementos) do campo são binários, é possível interpretar sua sequência como sendo uma palavra binária e ainda convertê-la para uma representação decimal única, conveniente para a transmissão, processamento e armazenamento por sistemas digitais.

Tabela 25: Elementos do campo $GF(2^2)$

Elemento	a_1	a_0	Representação decimal de $[a_1a_0]$
0	0	0	0
1	0	1	1
x	1	0	2
$x + 1$	1	1	3

A extensão do campo F_q a partir do campo base F_p , com $q = p^n$, utilizando o polinômio irredutível $p(x)$ de grau n , é representada por:

$$F_q = F_p[x]/\langle p(x) \rangle$$

O quociente ‘/’ do anel de polinômios $F_p[x]$ pelo *ideal principal* $\langle p(x) \rangle$ gerado por $p(x)$, forma um *anel quociente*. É provado que um anel quociente é um campo quando o seu ideal for *maximal* (um polinômio irredutível neste caso) (PINTER, 1982, p. 192).

As tabelas 26 e 27 mostram todas as adições e multiplicações possíveis entre os elementos do campo $GF(4) = GF(2)[x]/\langle x^2 + x + 1 \rangle$, formado pelo anel de polinômios $GF(2)[x]$ módulo o polinômio $x^2 + x + 1$, de grau $n = 2$ e irredutível em $GF(2)$.

A construção algébrica aqui esboçada, em síntese, parte de um campo base F_p de ordem prima p , e o estende para o campo F_q de ordem $q = p^n$ através do uso de um anel de polinômios $F_p[x]$ e do emprego de um polinômio $p(x)$ de grau n irredutível

Tabela 26: Tabela soma para o $GF(2^2) = GF(2)[x]/\langle x^2 + x + 1 \rangle$

+	0	1	x	x+1
0	0	1	x	x+1
1	1	0	x+1	x
x	x	x+1	0	1
x+1	x+1	x	1	0

Tabela 27: Tabela multiplicação para o $GF(2^2) = GF(2)[x]/\langle x^2 + x + 1 \rangle$

×	0	1	x	x+1
0	0	0	0	0
1	0	1	x	x+1
x	0	x	x+1	1
x+1	0	x+1	1	x

em F_p . O polinômio $p(x)$ é utilizado como redutor após as operações de multiplicação para garantir que o polinômio resultante tenha grau $\leq n - 1$ e que um campo finito seja formado. É provado que a extensão F_q é um espaço vetorial sobre F_p , com dimensão n e base $\{1, x, x^2, \dots, x^{n-1}\}$.

A.5.4 Polinômio primitivo e elemento gerador do campo

Um *polinômio primitivo* $p(x)$ é um polinômio irredutível com uma característica adicional: ele não pode ser divisor de $x^n + 1$ para $n < q - 1$, onde q é o número de elementos do campo estendido. Por exemplo, no campo $GF(2)[x]/\langle p(x) \rangle$, o polinômio $p(x)$ de grau 8 para ser considerado primitivo, além de ser irredutível, $p(x)$ não pode ser divisor dos polinômios $x^{254} - 1$, $x^{253} - 1$, $x^{252} - 1$, etc.

A utilização de um polinômio primitivo $p(x)$ em uma extensão de campo assegura que os polinômios do conjunto $S = \{f_n(x) = x^n + 1 \mid 0 < n < q - 1\}$ não tenham $p(x)$ como fator, assim uma raiz α de $p(x)$ não é uma raiz para $f_n(x) \in S$, então $f_n(\alpha) = \alpha^n - 1 \neq 0 \Rightarrow \alpha^n \neq 1$ para todo $0 < n < q - 1$. Isto garante que α seja um elemento de ordem $q - 1$, portanto um elemento gerador do grupo multiplicativo do campo, ou seja, todos os elementos do campo, exceto o zero, podem ser representados pela sequência $\alpha^0, \alpha^1, \dots, \alpha^{q-2}$.

Por exemplo, o campo $GF(16) = GF(2)[x]/\langle x^4 + x + 1 \rangle$, baseado em um $p(x) = x^4 + x + 1$ primitivo, possui o elemento gerador α , a raiz $p(\alpha) = 0$. Todos os elementos não zero deste campo podem ser representados pela sequência $\alpha^0, \alpha^1, \dots, \alpha^{14}$, sem repetições, conforme a Tabela 28.

Tabela 28: Geração de todos os elementos não zero do campo $GF(16) = GF(2)[x]/\langle x^4 + x + 1 \rangle$ através das potências da raiz α de $p(x)$

Potências de α	Elemento gerado
α^0	1
α^1	α
α^2	α^2
α^3	α^3
α^4	$\alpha + 1$
α^5	$\alpha^2 + \alpha$
α^6	$\alpha^3 + \alpha^2$
α^7	$\alpha^3 + \alpha + 1$
α^8	$\alpha^2 + 1$
α^9	$\alpha^3 + \alpha$
α^{10}	$\alpha^2 + \alpha + 1$
α^{11}	$\alpha^3 + \alpha^2 + \alpha$
α^{12}	$\alpha^3 + \alpha^2 + \alpha + 1$
α^{13}	$\alpha^3 + \alpha^2 + 1$
α^{14}	$\alpha^3 + 1$
α^{15}	1

O uso de um elemento gerador g facilita o cálculo das multiplicações no campo, pois $g^a \cdot g^b = g^{a+b} = g^s$, onde $0 \leq s \leq q-2$ e $a+b \equiv s \pmod{q-1}$. Assim, a multiplicação entre elementos do campo se dá basicamente por consultas às tabelas de potências e logaritmos de g , armazenadas na memória. Esta operação é muito mais simples e rápida do que a multiplicação tradicional, que requer o produto dos polinômios e a redução (através do resto da divisão) do resultado por $p(x)$.

Além disso, em um $GF(2^n)$, quando o elemento gerador g é a raiz α de $p(x)$ (ou seja, $p(x)$ é primitivo), a construção da tabela de potências do elemento gerador α é facilitada, pois as sucessivas multiplicações por α podem ser implementadas como deslocamentos de bits, e quando a redução do grau do polinômio resultante for necessária, a subtração por $p(x)$ é feita através da operação *OU exclusivo*. Por isso, nas implementações de códigos corretores de erros que utilizam a aritmética do Campo de Galois, é comum o uso de polinômios primitivos.

A.5.5 Elemento gerador de um campo formado com um polinômio não primitivo

Esta seção ilustra que, quando $p(x)$ for irredutível mas não primitivo, um campo é formado, mas a raiz α de $p(x)$ não será um elemento gerador do campo.

Considere o campo $F = GF(2)[x]/\langle p(x) \rangle$ com $p(x) = x^4 + x^3 + x^2 + x + 1$, um polinômio de grau $n = 4$ irredutível no $GF(2)$, mas não primitivo.

A Tabela 29 mostra a tentativa de se gerar todos os elementos do campo, exceto o zero, através de potências sucessivas da raiz α . Observe que o resultado é falho, pois há repetição a cada 5 potências, por exemplo $\alpha^{10} = \alpha^5 = \alpha^0 = 1$. Isto ocorre pois o elemento α do campo F e raiz de $p(x)$ possui ordem 5 (repete a cada 5 potências sucessivas). Isto não significa que F não formou um campo, mas que α não é um elemento gerador deste campo, pois o $p(x)$ escolhido não é primitivo.

Tabela 29: Tentativa de se gerar todos os elementos não zero do campo $F = GF(2)[x]/\langle x^4 + x^3 + x^2 + x + 1 \rangle$ através das potências da raiz α de $p(x)$

Potências de α	Elemento gerado
α^0	1
α^1	α
α^2	α^2
α^3	α^3
α^4	$\alpha^3 + \alpha^2 + \alpha + 1$
α^5	1
α^6	α
α^7	α^2
α^8	α^3
α^9	$\alpha^3 + \alpha^2 + \alpha + 1$
α^{10}	1
α^{11}	α
α^{12}	α^2
α^{13}	α^3
α^{14}	$\alpha^3 + \alpha^2 + \alpha + 1$
α^{15}	1

É possível verificar que o campo F deste exemplo, apesar de não possuir a raiz α de $p(x)$ como elemento gerador, possui outros 8 elementos geradores: $(x+1)$, (x^2+1) , (x^2+x) , (x^2+x+1) , (x^3+1) , (x^3+x) , (x^3+x+1) e (x^3+x^2+x) . A Tabela 30, por exemplo, mostra as potências sucessivas do elemento $\beta = (x+1) \in F$: $\{(x+1)^0, (x+1)^1, \dots, (x+1)^{q-1}\}$.

Observa-se que $(x + 1)$ é de fato um elemento gerador, pois suas potências $\beta^0 \dots \beta^{14}$ correspondem a todos os $q - 1$ elementos não zero do campo, sem repetições.

Tabela 30: Geração de todos os elementos não zero do campo $F = GF(2)[x]/\langle x^4 + x^3 + x^2 + x + 1 \rangle$ através das potências do elemento $\beta = x + 1$

Potências de $\beta = x + 1$	Elemento gerado
β^0	1
β^1	$x + 1$
β^2	$x^2 + 1$
β^3	$x^3 + x^2 + x + 1$
β^4	$x^3 + x^2 + x$
β^5	$x^3 + x^2 + 1$
β^6	x^3
β^7	$x^2 + x + 1$
β^8	$x^3 + 1$
β^9	x^2
β^{10}	$x^3 + x^2$
β^{11}	$x^3 + x + 1$
β^{12}	x
β^{13}	$x^2 + x$
β^{14}	$x^3 + x$
β^{15}	1

APÊNDICE B – Potências de α para o $GF(256)$

A Tabela 31 mostra todos os elementos do campo $GF(2^8) = GF(2)[x]/\langle x^8 + x^4 + x^3 + x^2 + 1 \rangle$. Os elementos não zero foram gerados através das potências de α , a raiz do polinômio primitivo $x^8 + x^4 + x^3 + x^2 + 1$.

Tabela 31: Elementos do $GF(2^8) = GF(2)[x]/\langle p(x) = x^8 + x^4 + x^3 + x^2 + 1 \rangle$

Potência	Decimal	Binária	Polinomial
—	0	00000000	0
α^0	1	00000001	1
α^1	2	00000010	α
α^2	4	00000100	α^2
α^3	8	00001000	α^3
α^4	16	00010000	α^4
α^5	32	00100000	α^5
α^6	64	01000000	α^6
α^7	128	10000000	α^7
α^8	29	00011101	$\alpha^4 + \alpha^3 + \alpha^2 + 1$
α^9	58	00111010	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha$
α^{10}	116	01110100	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2$
α^{11}	232	11101000	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3$
α^{12}	205	11001101	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + 1$
α^{13}	135	10000111	$\alpha^7 + \alpha^2 + \alpha + 1$
α^{14}	19	00010011	$\alpha^4 + \alpha + 1$
α^{15}	38	00100110	$\alpha^5 + \alpha^2 + \alpha$
α^{16}	76	01001100	$\alpha^6 + \alpha^3 + \alpha^2$
α^{17}	152	10011000	$\alpha^7 + \alpha^4 + \alpha^3$
α^{18}	45	00101101	$\alpha^5 + \alpha^3 + \alpha^2 + 1$
α^{19}	90	01011010	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha$
α^{20}	180	10110100	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2$
α^{21}	117	01110101	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + 1$
α^{22}	234	11101010	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha$
α^{23}	201	11001001	$\alpha^7 + \alpha^6 + \alpha^3 + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{24}	143	10001111	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{25}	3	00000011	$\alpha + 1$
α^{26}	6	00000110	$\alpha^2 + \alpha$
α^{27}	12	00001100	$\alpha^3 + \alpha^2$
α^{28}	24	00011000	$\alpha^4 + \alpha^3$
α^{29}	48	00110000	$\alpha^5 + \alpha^4$
α^{30}	96	01100000	$\alpha^6 + \alpha^5$
α^{31}	192	11000000	$\alpha^7 + \alpha^6$
α^{32}	157	10011101	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{33}	39	00100111	$\alpha^5 + \alpha^2 + \alpha + 1$
α^{34}	78	01001110	$\alpha^6 + \alpha^3 + \alpha^2 + \alpha$
α^{35}	156	10011100	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2$
α^{36}	37	00100101	$\alpha^5 + \alpha^2 + 1$
α^{37}	74	01001010	$\alpha^6 + \alpha^3 + \alpha$
α^{38}	148	10010100	$\alpha^7 + \alpha^4 + \alpha^2$
α^{39}	53	00110101	$\alpha^5 + \alpha^4 + \alpha^2 + 1$
α^{40}	106	01101010	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha$
α^{41}	212	11010100	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2$
α^{42}	181	10110101	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + 1$
α^{43}	119	01110111	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{44}	238	11101110	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha$
α^{45}	193	11000001	$\alpha^7 + \alpha^6 + 1$
α^{46}	159	10011111	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{47}	35	00100011	$\alpha^5 + \alpha + 1$
α^{48}	70	01000110	$\alpha^6 + \alpha^2 + \alpha$
α^{49}	140	10001100	$\alpha^7 + \alpha^3 + \alpha^2$
α^{50}	5	00000101	$\alpha^2 + 1$
α^{51}	10	00001010	$\alpha^3 + \alpha$
α^{52}	20	00010100	$\alpha^4 + \alpha^2$
α^{53}	40	00101000	$\alpha^5 + \alpha^3$
α^{54}	80	01010000	$\alpha^6 + \alpha^4$
α^{55}	160	10100000	$\alpha^7 + \alpha^5$
α^{56}	93	01011101	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{57}	186	10111010	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha$
α^{58}	105	01101001	$\alpha^6 + \alpha^5 + \alpha^3 + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{59}	210	11010010	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha$
α^{60}	185	10111001	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + 1$
α^{61}	111	01101111	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{62}	222	11011110	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{63}	161	10100001	$\alpha^7 + \alpha^5 + 1$
α^{64}	95	01011111	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{65}	190	10111110	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{66}	97	01100001	$\alpha^6 + \alpha^5 + 1$
α^{67}	194	11000010	$\alpha^7 + \alpha^6 + \alpha$
α^{68}	153	10011001	$\alpha^7 + \alpha^4 + \alpha^3 + 1$
α^{69}	47	00101111	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{70}	94	01011110	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{71}	188	10111100	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$
α^{72}	101	01100101	$\alpha^6 + \alpha^5 + \alpha^2 + 1$
α^{73}	202	11001010	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha$
α^{74}	137	10001001	$\alpha^7 + \alpha^3 + 1$
α^{75}	15	00001111	$\alpha^3 + \alpha^2 + \alpha + 1$
α^{76}	30	00011110	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{77}	60	00111100	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$
α^{78}	120	01111000	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3$
α^{79}	240	11110000	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4$
α^{80}	253	11111101	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{81}	231	11100111	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + \alpha + 1$
α^{82}	211	11010011	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha + 1$
α^{83}	187	10111011	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{84}	107	01101011	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha + 1$
α^{85}	214	11010110	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + \alpha$
α^{86}	177	10110001	$\alpha^7 + \alpha^5 + \alpha^4 + 1$
α^{87}	127	01111111	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{88}	254	11111110	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{89}	225	11100001	$\alpha^7 + \alpha^6 + \alpha^5 + 1$
α^{90}	223	11011111	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{91}	163	10100011	$\alpha^7 + \alpha^5 + \alpha + 1$
α^{92}	91	01011011	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{93}	182	10110110	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{94}	113	01110001	$\alpha^6 + \alpha^5 + \alpha^4 + 1$
α^{95}	226	11100010	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha$
α^{96}	217	11011001	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + 1$
α^{97}	175	10101111	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{98}	67	01000011	$\alpha^6 + \alpha + 1$
α^{99}	134	10000110	$\alpha^7 + \alpha^2 + \alpha$
α^{100}	17	00010001	$\alpha^4 + 1$
α^{101}	34	00100010	$\alpha^5 + \alpha$
α^{102}	68	01000100	$\alpha^6 + \alpha^2$
α^{103}	136	10001000	$\alpha^7 + \alpha^3$
α^{104}	13	00001101	$\alpha^3 + \alpha^2 + 1$
α^{105}	26	00011010	$\alpha^4 + \alpha^3 + \alpha$
α^{106}	52	00110100	$\alpha^5 + \alpha^4 + \alpha^2$
α^{107}	104	01101000	$\alpha^6 + \alpha^5 + \alpha^3$
α^{108}	208	11010000	$\alpha^7 + \alpha^6 + \alpha^4$
α^{109}	189	10111101	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{110}	103	01100111	$\alpha^6 + \alpha^5 + \alpha^2 + \alpha + 1$
α^{111}	206	11001110	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha$
α^{112}	129	10000001	$\alpha^7 + 1$
α^{113}	31	00011111	$\alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{114}	62	00111110	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{115}	124	01111100	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$
α^{116}	248	11111000	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3$
α^{117}	237	11101101	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + 1$
α^{118}	199	11000111	$\alpha^7 + \alpha^6 + \alpha^2 + \alpha + 1$
α^{119}	147	10010011	$\alpha^7 + \alpha^4 + \alpha + 1$
α^{120}	59	00111011	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{121}	118	01110110	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha$
α^{122}	236	11101100	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2$
α^{123}	197	11000101	$\alpha^7 + \alpha^6 + \alpha^2 + 1$
α^{124}	151	10010111	$\alpha^7 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{125}	51	00110011	$\alpha^5 + \alpha^4 + \alpha + 1$
α^{126}	102	01100110	$\alpha^6 + \alpha^5 + \alpha^2 + \alpha$
α^{127}	204	11001100	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2$
α^{128}	133	10000101	$\alpha^7 + \alpha^2 + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{129}	23	00010111	$\alpha^4 + \alpha^2 + \alpha + 1$
α^{130}	46	00101110	$\alpha^5 + \alpha^3 + \alpha^2 + \alpha$
α^{131}	92	01011100	$\alpha^6 + \alpha^4 + \alpha^3 + \alpha^2$
α^{132}	184	10111000	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3$
α^{133}	109	01101101	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + 1$
α^{134}	218	11011010	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha$
α^{135}	169	10101001	$\alpha^7 + \alpha^5 + \alpha^3 + 1$
α^{136}	79	01001111	$\alpha^6 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{137}	158	10011110	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{138}	33	00100001	$\alpha^5 + 1$
α^{139}	66	01000010	$\alpha^6 + \alpha$
α^{140}	132	10000100	$\alpha^7 + \alpha^2$
α^{141}	21	00010101	$\alpha^4 + \alpha^2 + 1$
α^{142}	42	00101010	$\alpha^5 + \alpha^3 + \alpha$
α^{143}	84	01010100	$\alpha^6 + \alpha^4 + \alpha^2$
α^{144}	168	10101000	$\alpha^7 + \alpha^5 + \alpha^3$
α^{145}	77	01001101	$\alpha^6 + \alpha^3 + \alpha^2 + 1$
α^{146}	154	10011010	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha$
α^{147}	41	00101001	$\alpha^5 + \alpha^3 + 1$
α^{148}	82	01010010	$\alpha^6 + \alpha^4 + \alpha$
α^{149}	164	10100100	$\alpha^7 + \alpha^5 + \alpha^2$
α^{150}	85	01010101	$\alpha^6 + \alpha^4 + \alpha^2 + 1$
α^{151}	170	10101010	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha$
α^{152}	73	01001001	$\alpha^6 + \alpha^3 + 1$
α^{153}	146	10010010	$\alpha^7 + \alpha^4 + \alpha$
α^{154}	57	00111001	$\alpha^5 + \alpha^4 + \alpha^3 + 1$
α^{155}	114	01110010	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha$
α^{156}	228	11100100	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2$
α^{157}	213	11010101	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + 1$
α^{158}	183	10110111	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{159}	115	01110011	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha + 1$
α^{160}	230	11100110	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + \alpha$
α^{161}	209	11010001	$\alpha^7 + \alpha^6 + \alpha^4 + 1$
α^{162}	191	10111111	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{163}	99	01100011	$\alpha^6 + \alpha^5 + \alpha + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{164}	198	11000110	$\alpha^7 + \alpha^6 + \alpha^2 + \alpha$
α^{165}	145	10010001	$\alpha^7 + \alpha^4 + 1$
α^{166}	63	00111111	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{167}	126	01111110	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$
α^{168}	252	11111100	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$
α^{169}	229	11100101	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + 1$
α^{170}	215	11010111	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{171}	179	10110011	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha + 1$
α^{172}	123	01111011	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{173}	246	11110110	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha$
α^{174}	241	11110001	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + 1$
α^{175}	255	11111111	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{176}	227	11100011	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha + 1$
α^{177}	219	11011011	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{178}	171	10101011	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha + 1$
α^{179}	75	01001011	$\alpha^6 + \alpha^3 + \alpha + 1$
α^{180}	150	10010110	$\alpha^7 + \alpha^4 + \alpha^2 + \alpha$
α^{181}	49	00110001	$\alpha^5 + \alpha^4 + 1$
α^{182}	98	01100010	$\alpha^6 + \alpha^5 + \alpha$
α^{183}	196	11000100	$\alpha^7 + \alpha^6 + \alpha^2$
α^{184}	149	10010101	$\alpha^7 + \alpha^4 + \alpha^2 + 1$
α^{185}	55	00110111	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{186}	110	01101110	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha$
α^{187}	220	11011100	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2$
α^{188}	165	10100101	$\alpha^7 + \alpha^5 + \alpha^2 + 1$
α^{189}	87	01010111	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{190}	174	10101110	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha$
α^{191}	65	01000001	$\alpha^6 + 1$
α^{192}	130	10000010	$\alpha^7 + \alpha$
α^{193}	25	00011001	$\alpha^4 + \alpha^3 + 1$
α^{194}	50	00110010	$\alpha^5 + \alpha^4 + \alpha$
α^{195}	100	01100100	$\alpha^6 + \alpha^5 + \alpha^2$
α^{196}	200	11001000	$\alpha^7 + \alpha^6 + \alpha^3$
α^{197}	141	10001101	$\alpha^7 + \alpha^3 + \alpha^2 + 1$
α^{198}	7	00000111	$\alpha^2 + \alpha + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{199}	14	00001110	$\alpha^3 + \alpha^2 + \alpha$
α^{200}	28	00011100	$\alpha^4 + \alpha^3 + \alpha^2$
α^{201}	56	00111000	$\alpha^5 + \alpha^4 + \alpha^3$
α^{202}	112	01110000	$\alpha^6 + \alpha^5 + \alpha^4$
α^{203}	224	11100000	$\alpha^7 + \alpha^6 + \alpha^5$
α^{204}	221	11011101	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{205}	167	10100111	$\alpha^7 + \alpha^5 + \alpha^2 + \alpha + 1$
α^{206}	83	01010011	$\alpha^6 + \alpha^4 + \alpha + 1$
α^{207}	166	10100110	$\alpha^7 + \alpha^5 + \alpha^2 + \alpha$
α^{208}	81	01010001	$\alpha^6 + \alpha^4 + 1$
α^{209}	162	10100010	$\alpha^7 + \alpha^5 + \alpha$
α^{210}	89	01011001	$\alpha^6 + \alpha^4 + \alpha^3 + 1$
α^{211}	178	10110010	$\alpha^7 + \alpha^5 + \alpha^4 + \alpha$
α^{212}	121	01111001	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + 1$
α^{213}	242	11110010	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha$
α^{214}	249	11111001	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + 1$
α^{215}	239	11101111	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{216}	195	11000011	$\alpha^7 + \alpha^6 + \alpha + 1$
α^{217}	155	10011011	$\alpha^7 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{218}	43	00101011	$\alpha^5 + \alpha^3 + \alpha + 1$
α^{219}	86	01010110	$\alpha^6 + \alpha^4 + \alpha^2 + \alpha$
α^{220}	172	10101100	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2$
α^{221}	69	01000101	$\alpha^6 + \alpha^2 + 1$
α^{222}	138	10001010	$\alpha^7 + \alpha^3 + \alpha$
α^{223}	9	00001001	$\alpha^3 + 1$
α^{224}	18	00010010	$\alpha^4 + \alpha$
α^{225}	36	00100100	$\alpha^5 + \alpha^2$
α^{226}	72	01001000	$\alpha^6 + \alpha^3$
α^{227}	144	10010000	$\alpha^7 + \alpha^4$
α^{228}	61	00111101	$\alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{229}	122	01111010	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha$
α^{230}	244	11110100	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2$
α^{231}	245	11110101	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + 1$
α^{232}	247	11110111	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1$
α^{233}	243	11110011	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha + 1$

A tabela continua na próxima página...

Tabela 31 – continuação da página anterior

Potência	Decimal	Binária	Polinomial
α^{234}	251	11111011	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha + 1$
α^{235}	235	11101011	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + \alpha + 1$
α^{236}	203	11001011	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha + 1$
α^{237}	139	10001011	$\alpha^7 + \alpha^3 + \alpha + 1$
α^{238}	11	00001011	$\alpha^3 + \alpha + 1$
α^{239}	22	00010110	$\alpha^4 + \alpha^2 + \alpha$
α^{240}	44	00101100	$\alpha^5 + \alpha^3 + \alpha^2$
α^{241}	88	01011000	$\alpha^6 + \alpha^4 + \alpha^3$
α^{242}	176	10110000	$\alpha^7 + \alpha^5 + \alpha^4$
α^{243}	125	01111101	$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + 1$
α^{244}	250	11111010	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha$
α^{245}	233	11101001	$\alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 + 1$
α^{246}	207	11001111	$\alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + \alpha + 1$
α^{247}	131	10000011	$\alpha^7 + \alpha + 1$
α^{248}	27	00011011	$\alpha^4 + \alpha^3 + \alpha + 1$
α^{249}	54	00110110	$\alpha^5 + \alpha^4 + \alpha^2 + \alpha$
α^{250}	108	01101100	$\alpha^6 + \alpha^5 + \alpha^3 + \alpha^2$
α^{251}	216	11011000	$\alpha^7 + \alpha^6 + \alpha^4 + \alpha^3$
α^{252}	173	10101101	$\alpha^7 + \alpha^5 + \alpha^3 + \alpha^2 + 1$
α^{253}	71	01000111	$\alpha^6 + \alpha^2 + \alpha + 1$
α^{254}	142	10001110	$\alpha^7 + \alpha^3 + \alpha^2 + \alpha$