UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES

FÁBIO ROBERTO OLIVEIRA
MARCELO BARAUCE
MARCOS MENINO PINHEIRO

COMUNICAÇÃO SEM FIO USANDO DMX 512

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2012

FÁBIO ROBERTO OLIVEIRA
MARCELO BARAUCE
MARCOS MENINO PINHEIRO

COMUNICAÇÃO SEM FIO USANDO DMX 512

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Departamento Acadêmico de Eletrônica – DAELN – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para o título de Tecnólogo.

Orientador: Prof. M.Sc. Vagner Gonçalves Leitão

CURITIBA
2012

FÁBIO ROBERTO DE OLIVEIRA
MARCELO BARAUCE
MARCOS MENINO PINHEIRO

COMUNICAÇÃO SEM FIO USANDO DMX 512

Este trabalho de conclusão de curso foi apresentado no dia 06 de agosto de 2012, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas de Telecomunicações, outorgado pela Universidade Tecnológica Federal do Paraná. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. M.Sc. César Janeczko
Coordenador de Curso
Departamento Acadêmico de Eletrônica

_____
Prof. Ph.D Décio Estevão do Nascimento
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof ª. M.Sc. Simone Crocetti
Banca

_____
Prof. M.Sc. Vagner Gonçalves Leitão
Orientador

_____
Prof. Ph. D. Augusto Foronda
Banca

A Folha de Aprovação assinada encontra-se na Coordenação do Curso

RESUMO

OLIVEIRA, Fábio Roberto; BARAUCE, Marcelo; PINHEIRO, Marcos Menino. Comunicação Sem Fio usando DMX 512. 2012. 115 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações) – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

Este Trabalho de Conclusão de Curso apresenta a construção de um protótipo de sistema de comunicação sem fio, com a finalidade de controlar equipamentos de iluminação cênica utilizados em eventos. Apresenta pesquisas em diversas áreas como eletrônica analógica e digital, sinais e sistemas, micro-controladores e ferramentas de gestão. Construiu-se dois protótipos usando módulos comerciais ZigBee para conversão e transmissão dos sinais do protocolo DMX 512, que atenderam com êxito a proposta inicial, pois eliminaram a necessidade do uso do cabo entre os equipamentos envolvidos, de forma segura e com baixo consumo de energia.

Palavras-chave: DMX 512. Comunicação Sem Fio. Módulo ZigBee.

ABSTRACT

OLIVEIRA, Fábio Roberto; BARAUCE, Marcelo; PINHEIRO, Marcos Menino. Wireless Communication Using DMX 512. 2012. 115 f. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações) – Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2012.

The present document shows the building of a wireless communication prototype system, in order to control stage lighting equipment used in events. Present research in many areas, such as analog and digital eletronics, signals and systems, micro-controlers and management tools. Were developed two prototypes using commercial ZigBee modules for conversion and transmission of signals from the DMX 512 protocol, which successfully met the initial proposal eliminating the use of cables between equipments, securely and with low power consumption.

Keywords: DMX 512. Wireless Communication. ZigBee Module.

LISTA DE SIGLAS E ACRÔNIMOS

AM	Amplitude Modulada
ANATEL	Agência Nacional de Telecomunicações
ASK	*Amplitude-Shift Keying*
CI	Circuito Integrado
DMX	*Digital MultipleX*
EIA	*Electronic Industries Alliance*
FM	*Frequency Modulation*
FSK	*Frequency-Shift Keying*
ISM	*Industrial, Scientist and Medical*
ISO	*International Organization for Standardization*
OOK	*On-Off Keying*
OSI	*Open Systems Interconnection*
MAB	*Mark After Break*
PIB	Produto Interno Bruto
Proto-Board	*Prototype Board* – Placa de protótipo
PSK	*Phase-Shift Keying*
QAM	*Quadrature Amplitude Modulation*
RS	*Recommendation Standard*
TIA	*Telecommunications Industry Association*
TTL	*Transistor-Transistor Logic* – Lógica transistor-transistor
USITT	*United States Institute for Theatre Technology*
WBS	*Work Breakdown Structure*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
WMAN	*Wireless Metropolitan Area Network*
WPAN	*Wireless Personal Area Network*
WWAN	*Wireless Wide Area Network*

LISTA DE FIGURAS

Figura 1 - Diagrama de blocos de uma conexão DMX 512 convencional 17
Figura 2 - Diagrama de blocos de uma conexão DMX 512 sem fio 19
Figura 3 - Circuito equivalente da impedância em linhas de transmissão 23
Figura 4 - Principais tipos de terminação 24
Figura 5 - Principais topologias de rede 25
Figura 6 - Exemplo de topologia em cadeia com resistores de terminação 25
Figura 7 - *Frame* DMX 512 27
Figura 8 - XLR5 – vista através do cabo 29
Figura 9 - Sinais, portadoras e técnicas de modulação 30
Figura 10 - Modulador FSK 33
Figura 11 - Exemplo de demodulador FSK 33
Figura 12 - Modos de operação dos módulos xbee 39
Figura 13 - Exemplo de comunicação serial com controle de fluxo 39
Figura 14 - Exemplo de comunicação serial sem controle de fluxo 40
Figura 15 - Topologia de redes ZigBee 42
Figura 16 - Circuito recomendado LM-7805 – 5 VCC 49
Figura 17 - Circuito recomendado pelo manual do LM-317 49
Figura 18 - Cálculos dos resistores de polarização do LM-317 50
Figura 19 - Circuito de alimentação do xbee 51
Figura 20 - Diagrama simplificado do MAX-485 52
Figura 21 - Portas PIC16F648A 52
Figura 22 - Circuito usado no transmissor 53
Figura 23 - Circuito usado no receptor 55
Figura 24 - Circuito de configuração do xbee 56
Figura 25 - Circuito final do protótipo xbee 58
Figura 26 - Fluxograma simplificado do programa do transmissor 60
Figura 27 - Fluxograma simplificado do programa do receptor 60
Figura 28 - Fluxograma simplificado do programa 61
Figura 29 - X-CTU – Coordinator configuration 62
Figura 30 - X-CTU – End device configuration 63
Figura 31 - Placa de circuito impresso 65

Figura 32 - Vista superior da placa.....66
Figura 33 - Vista inferior da placa.....67
Figura 34 - Vista inferior – Simulação de raios-X.....68
Figura 35 - Gráfico de Gantt do projeto.....77
Figura 36 - Macro tarefas do WBS.....78
Figura 37 - Definições básicas e de apoio ao projeto.....79
Figura 38 - Protótipo de interface sem fio.....79
Figura 39 - Detalhamento da “Pesquisa e Desenvolvimento”.....79
Figura 40 - Detalhamento da “Construção do protótipo”.....80
Figura 41 - Detalhamento dos “Ensaios e testes”.....80
Figura 42 - Detalhamento da “Documentação do projeto”.....80
Figura 43 - Detalhamento da “Documentação Final do TCC”.....81
Figura 44 - Detalhamento da “Defesa do projeto”.....81
Figura 45 - Diagrama em blocos do CC1100E.....85
Figura 46 - Circuito típico para aplicação e avaliação (950 MHz).....85
Figura 47 - Circuito de conversão para o transmissor.....88
Figura 48 - Diagrama de blocos – RS-485 para xbee.....90
Figura 49 - Circuito de geração dos sinais diferenciais RS-485.....93
Figura 50 - SN7404 usado para as saídas diferenciais.....94
Figura 51 - Protótipo 2 – Transmissor FSK.....95
Figura 52 - Protótipo 2 – Receptor FSK.....96
Figura 53 - Cálculos de R1 e F1, usados no gerador FSK – 1,25 MHz.....96
Figura 54 - Cálculos de R2 e F2, usados no gerador FSK – 750 kHz.....97
Figura 55 - Circuito gerador FSK usando XR-2206.....97
Figura 56 - Cálculos de R1 e F1, usados no gerador FSK – 260 kHz.....99
Figura 57 - Cálculos de R2 e F2, usados no gerador FSK – 300 kHz.....100
Figura 58 - Circuito amplificador AM.....101
Figura 59 - Diagrama simplificado do XR-2211.....103
Figura 60 - Cálculos dos componentes usados no circuito receptor.....103
Figura 61 - Cálculos dos componentes usados no circuito receptor - 2.....104
Figura 62 - Esquema elétrico simplificado – Circuito final.....105

LISTA DE FOTOGRAFIAS

Fotografia 1 - Conexão com o controlador 18
Fotografia 2 - Conexão com o dispositivo 18
Fotografia 3 - Conexão DMX 512 entre controlador e dispositivo controlado 18
Fotografia 4 - Conector XLR-5 (macho) 28
Fotografia 5 - Conector XLR-5 (fêmea) 28
Fotografia 6 - Conector XLR-5 (macho) 28
Fotografia 7 - Conector XLR-5 (fêmea) 28
Fotografia 8 - Módulo xbee.....46
Fotografia 9 - Tamanho da interface xbee 46
Fotografia 10 - Comparativo – CC1100E *vs* tampa de caneta 87

LISTA DE GRÁFICOS

Gráfico 1 - Comprimento do cabo *vs* velocidade de comunicação.....21
Gráfico 2 - Modo diferencial de tensão.....22
Gráfico 3 - Portadora senoidal.....31
Gráfico 4 - Sinal modulante (0010110010).....31
Gráfico 5 - Portadora modulada – modulação OOK.....32
Gráfico 6 - Portadora modificada – modulação FSK.....32
Gráfico 7 - Portadora modificada – modulação PSK.....34
Gráfico 8 - Forma de onda do sinal QAM.....36
Gráfico 9 - Medição de frequência do sinal DMX-512 com osciloscópio.....43
Gráfico 10 - Medição do intervalo entre informações do sinal DMX-512.....44
Gráfico 11 - Medição de tensão do sinal DMX-512.....44
Gráfico 12 - Sinal de 250 kHz.....98
Gráfico 13 - Sinal de saída (entrada 0).....98
Gráfico 14 - Sinal de saída (entrada 1).....99
Gráfico 15 - Sinal modulado em FSK.....99

LISTA DE QUADROS

Quadro 1 - Descrição dos pinos do XLR-5 29
Quadro 2 - Relação entre bits x amplitude e fase – modulação QAM 35
Quadro 3 - Funções dos pinos do xbee 47
Quadro 4 - Características do PIC16F648A 54
Quadro 5 - Relação de componentes para configuração 56
Quadro 6 - Parâmetros configurados nos módulos xbee 64
Quadro 7 - Identificação do modo de associação 70
Quadro 8 - Comparação entre os métodos 83
Quadro 9 - Características resumidas do circuito do método 1 e 2 84
Quadro 10 - Lista de componentes – 950 MHz 86
Quadro 11 - Tabelas verdade do MAX485 92
Quadro 12 - Componentes do modulador FSK – 1250 e 750 kHz 98
Quadro 13 - Componentes do modulador FSK – 300 e 260 kHz 100
Quadro 14 - Componentes utilizados no circuito receptor 104

SUMÁRIO

1 Introdução 14
1.1 Problema 15
1.2 Objetivos 16
1.2.1 Objetivo Geral 16
1.2.2 Objetivos Específicos 16
1.3 Metodologia de Pesquisa 17
2 Fundamentação Teórica 20
2.1 DMX 512 20
2.1.1 Características elétricas do padrão DMX 512 21
2.1.2 Formato e protocolo dos dados de comunicação DMX 512 26
2.1.3 Tipo de conector (XLR-5) 27
2.2 Modulação digital 29
2.2.1 Modulação ASK 30
2.2.2 Modulação FSK 32
2.2.3 Modulação PSK 33
2.2.4 Modulação QAM 34
2.3 Teorema de amostragem de Nyquist 36
2.4 ZigBee / Xbee 36
2.4.1 Introdução ao ZigBee 36
2.4.2 Características Gerais 37
2.4.3 Modos de operação dos dispositivos xbee 38
2.4.4 Topologias de rede 40
3 Projeto Wi-Fi DMX 43
3.1 Premissas 43
3.2 Protótipo xbee 45
3.2.1 Circuito de alimentação 48
3.2.2 Circuitos de transmissão 51
3.2.3 Circuito receptor 53
3.2.4 Circuito para gravação 55
3.2.5 Circuito final 57

3.2.6 Custos do protótipo xbee 59
3.2.7 Programação do PIC16F648A 60
3.2.8 Configuração do módulo xbee 61
3.2.9 Placa de circuito impresso 64
3.3 Testes 68
3.3.1 Funcionamento 68
3.3.2 Consumo 69
3.3.3 Alcance 70
4 Conclusão 71
5 Referências 72
APÊNDICE A - Cronograma previsto 77
APÊNDICE B - WBS do projeto 78
APÊNDICE B.1 - WBS – Macro tarefas 78
APÊNDICE B.2 - WBS detalhado 78
APÊNDICE C - Protótipos 82
APÊNDICE C.1 - Escolha dos protótipos 82
APÊNDICE C.2 - Protótipo alfa 84
APÊNDICE C.3 - Protótipo xbee – pesquisas 87
APÊNDICE C.4 - Protótipo beta 95
APÊNDICE D - Circuito Final 105
APÊNDICE E - Programa usado no PIC16F648A 106

1 INTRODUÇÃO

Nos últimos anos, o modelo de comunicação sem fio, conhecida como *wireless*, está presente em diversos segmentos tecnológicos. Dessa forma a comunicação sem fio se tornou fundamental para usuários que necessitam realizar seus deveres sem perder sua mobilidade. Seguindo esta mesma tendência, o mundo do entretenimento incorporou a tecnologia sem fios através da utilização de vários dispositivos, envolvendo sistemas que variam de modulações FM (*Frequency Modulation*) até protocolos mais sofisticados, como a família 802.11x (sistemas *Wi-Fi)* (RIBEIRO; TORRES; PEREIRA, 2006).

O crescimento da indústria do entretenimento fomenta o investimento em novas tecnologias e a profissionalização do setor tornando este uma fonte importante na geração de empregos diretos e indiretos (BRITO; FONTES, 2002).

Segundo perspectivas observadas recentemente, com o aquecimento do mercado brasileiro, o número de eventos deve continuar crescente nas próximas décadas. Anualmente, são realizados no país mais de 330 mil eventos de entretenimento, envolvendo cerca de 80 milhões de participantes, fato que resultou a geração de cerca de três milhões de empregos (diretos, terceirizados e indiretos), representando uma parcela de aproximadamente 3,1% do PIB (Produto Interno Bruto) do país (HOCHMAN, 2002).

Como consequência da citação anterior, até mesmo os pequenos eventos (reuniões empresariais, aniversários, formaturas...) são responsáveis pela expansão do setor, fato observado no crescente número de empresas que se especializaram nesse nicho de mercado. Cada vez mais observa-se que a promoção de pequenos eventos está perdendo seu ar amadorístico (LACERDA, 2005).

Atualmente, cerca de 90% das empresas do setor de entretenimento são formadas por pequenas e micro empresas, sendo que estas são formadas por um número poucas vezes maior que 20 pessoas. O número reduzido de colaboradores, aliado aos requisitos de qualidade e confiabilidade, implica diretamente em um alto nível de especialização dos profissionais e força a evolução tecnológica dos recursos envolvidos (LACERDA, 2005).

Assim sendo, esta proposta enfoca a criação de um sistema de comunicação via rádio frequência, para o controle dos equipamentos de um modelo

de iluminação cênica utilizado em eventos (DMX-512), buscando eliminar o cabeamento que vai da mesa de controle deste dispositivo até as lâmpadas envolvidas na atividade, propiciando um baixo custo de aquisição. Existem atualmente no mercado equipamentos sem fio que funcionam segundo o padrão 802.11b (11 Mbps) e que transmitem sinais do protocolo DMX512. Esses equipamentos são pouco utilizados, pois são produtos recentes – surgiram há menos de cinco anos – e possuem um alto custo de aquisição (RIBEIRO; TORRES; PEREIRA, 2006).

1.1 PROBLEMA

Eliminar a necessidade de cabeamento entre os equipamentos de iluminação DMX-512 e o console de controle utilizados em salas de espetáculo, concertos ao vivo, ou qualquer outro tipo de evento realizado em ambientes com área inferior a 200 m².

JUSTIFICATIVA

Nas décadas de 80 e 90, quando os sistemas de iluminação ainda eram baseados em comunicação analógica, cada equipamento utilizava um canal dedicado para seu controle, sendo que, deste modo, para controlar 50 equipamentos eram necessários 50 canais diferentes (GOMES, 2011).

Com esse modelo de comunicação, era necessária a criação de um padrão confiável e flexível, pois além dos diversos canais necessários para o controle, não existia um padrão de comunicação, ou seja, a maioria dos fabricantes carregava seus próprios protocolos de comunicação, limitando a compatibilidade entre aparelhos. Isto forçava equipamentos do mesmo fabricante e gerava limitações no orçamento. (TANEMBAUM, 1997; GOMES, 2011)

Com a criação do DMX, além da padronização da comunicação dos equipamentos de iluminação, foi proporcionada a redução do cabeamento e dos equipamentos de controle, pois passou-se a utilizar apenas um cabo com até 512 canais e um ou mais consoles para controlar os equipamentos de iluminação cênica. Contudo, mesmo com as melhorias provocadas pelo DMX, ainda havia o problema

das limitações do ambiente, pois nem todos os eventos permitem a passagem ou a exposição dos cabos entre o palco e a central de controle (GOMES, 2011).

Por questões de segurança, estética e custos, cada vez mais é dada a prioridade para a redução do cabeamento utilizado em eventos. Os cabos utilizados para eventos possuem custos relativamente altos para a aquisição e manutenção, custos de recursos operacionais para montagem e desmontagem e um volume físico elevado que representam custos para armazenagem e transporte.

Em função destas necessidades, a utilização de equipamentos sem fio se torna uma tendência devido à flexibilidade, mobilidade e confiabilidade propiciadas pelas tecnologias atuais.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Elaborar interfaces de comunicação sem fio que permitam a transmissão e recepção de sinais contendo informações do protocolo DMX 512.

1.2.2 Objetivos Específicos

Estudar as definições do protocolo DMX 512.

Estudar as técnicas de modulação para definição da técnica mais apropriada para modular os sinais DMX 512.

Desenvolver um conjunto de protótipos para a transmissão e a recepção dos sinais DMX 512.

Efetuar ensaios para verificar a viabilidade técnica do projeto.

1.3 METODOLOGIA DE PESQUISA

Para a realização dos objetivos desse projeto, foram desenvolvidas duas interfaces de comunicação sem fio, para que fosse realizada a comunicação entre um controlador DMX 512 e um receptor (cliente) DMX 512.

Essas interfaces foram projetadas e construídas em placa de circuito impresso. As interfaces modulam e demodulam os sinais DMX 512 em frequência. A primeira interface irá modular o sinal DMX 512 em um sinal de rádio frequência, que será demodulado pela segunda interface, que irá converter o sinal em um sinal DMX 512, permitindo que as informações geradas pelo controlador atuem nos clientes sem fio (dispositivos finais).

O controlador e receptor DMX 512 não identificam que estão sendo transmitidos por uma interface sem fio, salvo pequenos atrasos que possam ocorrer devido à transmissão sem fio, velocidade de comunicação e modulação utilizada. Esses atrasos não têm ônus para os receptores, uma vez que são na ordem de microssegundos e os dispositivos receptores não são aplicações de tempo real. Com isso, as interfaces podem ser facilmente utilizadas por qualquer receptor e controlador DMX 512.

Foi analisada a necessidade de uso de memória para armazenamento temporário (*buffer*) para que a taxa de transmissão seja compatível entre o protocolo DMX 512 e os sinais de rádio frequência utilizados.

A figura 1 ilustra um diagrama em blocos de uma conexão DMX-512:

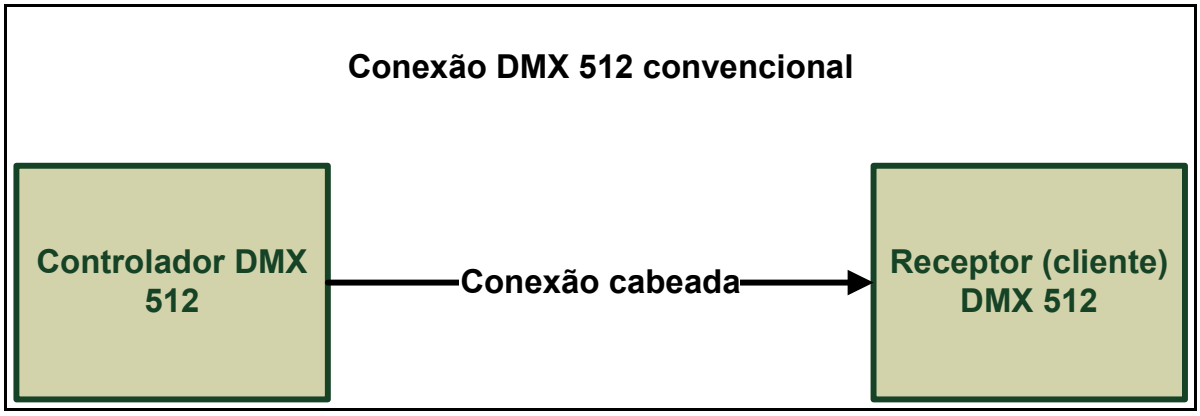


Figura 1 - Diagrama de blocos de uma conexão DMX 512 convencional
Fonte: Autoria própria

As fotografias 1, 2 e 3 foram tiradas de uma conexão DMX-512 entre uma mesa controladora e um projetor de luz.

Fotografia 1 - Conexão com o controlador
Fonte: Autoria própria

Fotografia 2 - Conexão com o dispositivo
Fonte: Autoria própria

Fotografia 3 - Conexão DMX 512 entre controlador e dispositivo controlado
Fonte: Autoria própria

A figura 2 ilustra o diagrama em blocos de uma conexão DMX-512 sem fio:

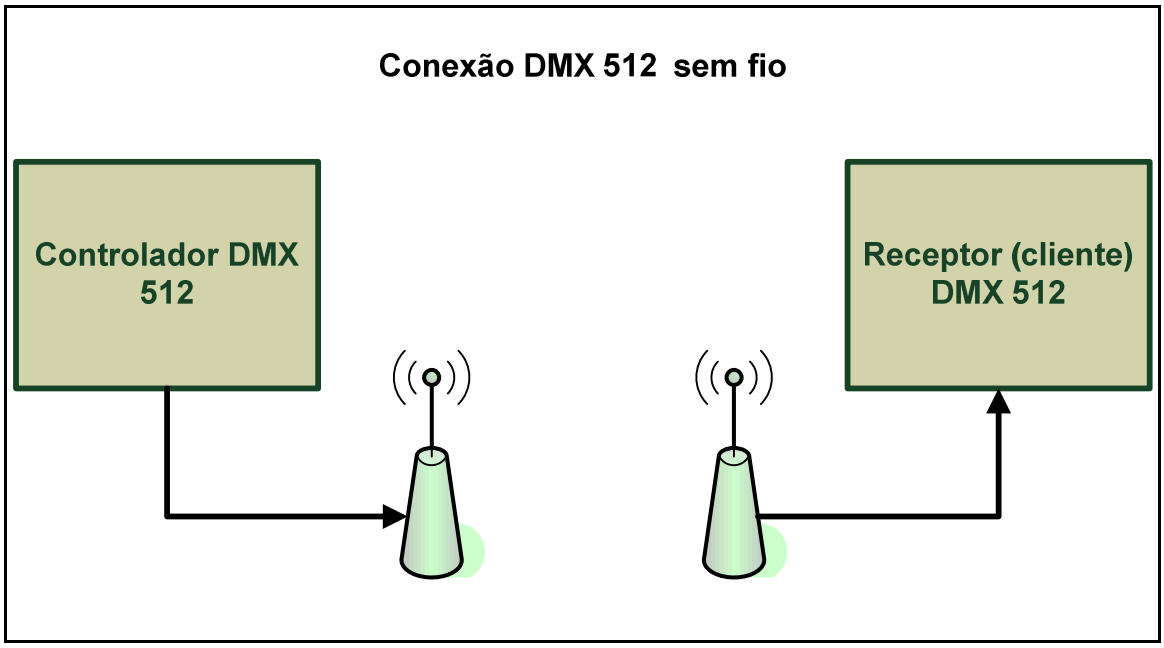


Figura 2 - Diagrama de blocos de uma conexão DMX 512 sem fio
Fonte: Autoria própria

As interfaces foram projetadas para ter sua alimentação elétrica preferencialmente dos dispositivos DMX 512, podendo operar também com fontes de baixa tensão (entre 5 V e 12 V), de forma que possa ser alimentado por baterias ou pilhas recarregáveis, sem a necessidade de cabos para a alimentação. Seguindo estas orientações, os circuitos têm um baixo consumo de energia, visando uma maior autonomia.

Foram registrados os custos do projeto para acompanhamento de despesas e para que seja avaliada sua viabilidade econômica em projetos futuros, em empresas que usem o protocolo DMX 512 para controle de dispositivos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 DMX 512

Na década de 1980, os diversos fabricantes tinham vários protocolos e dispositivos de fabricantes diferentes raramente se comunicavam entre si. Surgem, então, alguns padrões de comunicação, entre eles o modelo da *International Organization for Standardization* (ISO) – Organização Internacional para Padronização, conhecido como *Open Systems Interconnection* (OSI) – Sistema de Interconexão Aberto e os *Recommendation Standard* (RS) – Padrões Recomendáveis. (TANEMBAUM, 1997; UNITED..., 2011).

A *United States Institute for Theatre Technology* (USITT) iniciou o desenvolvimento de um protocolo para resolver o problema de interoperabilidade entre controladores, dispositivos de iluminação e acessórios. Surge com isso o protocolo DMX no ano de 1986, estabelecendo uma solução para esse problema de interoperabilidade (UNITED..., 2011; UNITED..., 2012).

DMX é uma abreviação de *Digital MultipleX*. O protocolo DMX define as seguintes características (UNITED..., 2012):

- Características elétricas (baseadas no RS-485);
- Formato dos dados de comunicação;
- Protocolo de dados;
- Tipo de conector;

As primeiras versões do DMX somente definiam questões de endereçamento e de cabeamento, sendo atualizada para a versão DMX 512, que permite a multiplexação de até 512 informações em um canal (UNITED..., 2012).

2.1.1 Características elétricas do padrão DMX 512

O padrão DMX 512 utiliza como padrão elétrico a norma foi desenvolvida por duas entidades, a *Electronic Industries Alliance* (EIA) e a *Telecommunications Industry Association* (TIA) conhecida como EIA/TIA-485. A norma EIA/TIA-485, chamada anteriormente de RS-485 define parâmetros elétricos para uma comunicação serial (FILARDI, 2011; KRON, 2011; PERRIN, 1999).

As interfaces de comunicação RS-485 usam linhas com tensões diferencias, podendo usar até 32 dispositivos, que podem ser receptores ou transmissores. Operam normalmente em modo *half duplex*, podendo operar em modo *full duplex*, desde que utilizando dois pares trançados de condutores, mantendo a compatibilidade com o padrão RS-422 (FILARDI, 2011; KRON, 2011; PERRIN, 1999).

Dessa forma, os dados podem atingir altas taxas em curtas distâncias (aproximadamente 10 Mbps em 12 m) ou longas distâncias com baixas taxas (aproximadamente 100 kbps em 1200 m) (FILARDI, 2011; KRON, 2011; PERRIN, 1999).

O gráfico 1 representa as taxas de dados em função da distância:

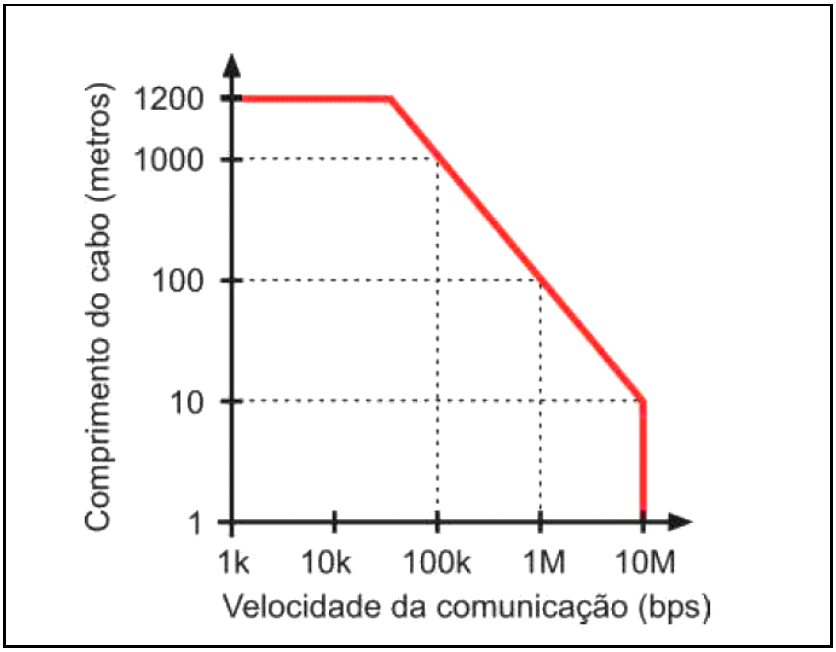


Gráfico 1 - Comprimento do cabo *vs* velocidade de comunicação
Fonte: Kron (2011)

Uma das principais características do RS-485 é a grande imunidade a ruído devido ao uso do modo diferencial de tensão e utilização de resistores para terminação do par diferencial. O gráfico 2 representa o modo diferencial usando um par trançado de fios (FILARDI, 2011; PERRIN, 1999):

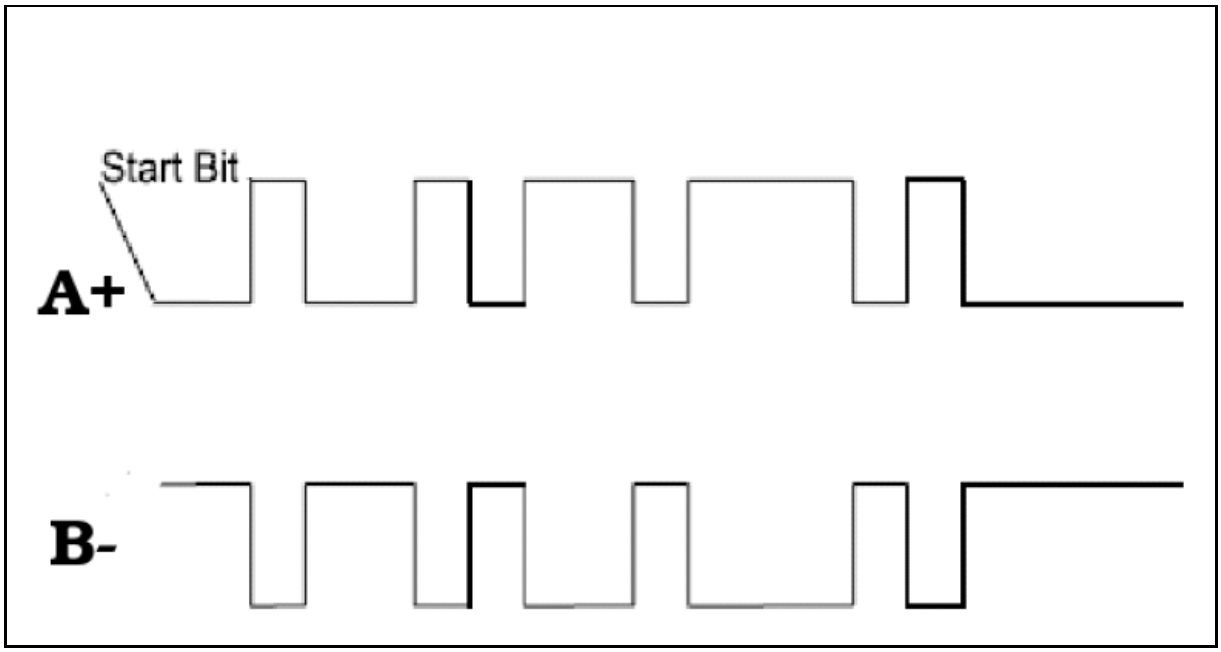


Gráfico 2 - Modo diferencial de tensão
Fonte: Adaptado de Filardi (2011)

No modo diferencial, o sinal é enviado em uma polarização, e o mesmo sinal é enviado pelo outro condutor em outra polarização, conforme pode ser observado na figura 7. Os circuitos receptores identificam a diferença de tensão entre os sinais dos condutores. Nesse caso, quando a tensão do condutor “A+“ for maior que no condutor “B-“, tem-se o nível lógico “1”, caso contrário, ou seja, uma tensão maior no condutor “B-“ maior que no condutor “A+”, tem-se o nível lógico “0”. Uma margem de ±0,2 V é definida para aumentar a tolerância ao ruído (FILARDI, 2011; KRON, 2011).

As terminações de rede são usadas para reduzir efeitos reflexivos em linhas de transmissão, igualando-se a impedância do término da linha com a impedância característica da linha (KRON, 2011; UNIVERSITY..., 2001).

A figura 3 ilustra o circuito equivalente da impedância em linhas de transmissão.

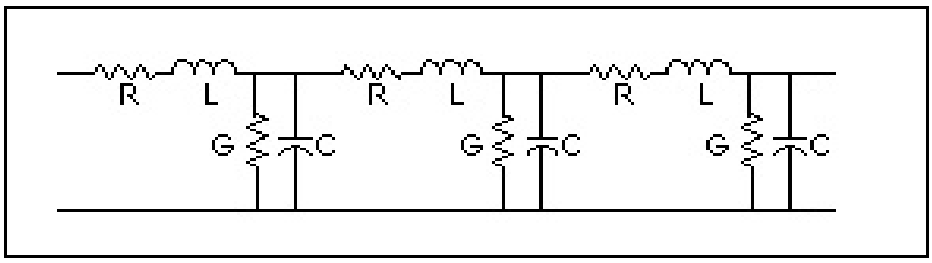


Figura 3 - Circuito equivalente da impedância em linhas de transmissão
Fonte: University... (2001)

Existem diversas técnicas de terminação de redes. Essas técnicas devem ser escolhidas levando-se em considerações os fatores como a distância entre o transmissor e o receptor e as taxas de dados necessárias (KRON, 2011; PERRIN, 1999).

Conexões não terminadas são mais baratas, porém tem a desvantagem de trabalhar com baixas taxas ou com baixas distâncias. As redes paralelas operam com boas taxas ou distância, porém está limitada a somente um transmissor / *driver*. Outra limitação da rede paralela é que o transmissor deve ser posicionado em uma extremidade e o resistor de terminação na outra (KRON, 2011; PERRIN, 1999).

Outro tipo de terminação usado é o bidirecional, que apresenta como principais vantagens manter a integridade dos dados e ter mais de um *driver* / transmissor na rede. Como desvantagens, além do custo, têm a necessidade de terminar os pontos da rede, inclusive cabos que estão desconectados dos aparelhos, pois esses podem ressonar e gerar interferências no sinal transmitido (KRON, 2011; PERRIN, 1999).

A figura 4 ilustra os três tipos de terminação mencionados (KRON, 2011):

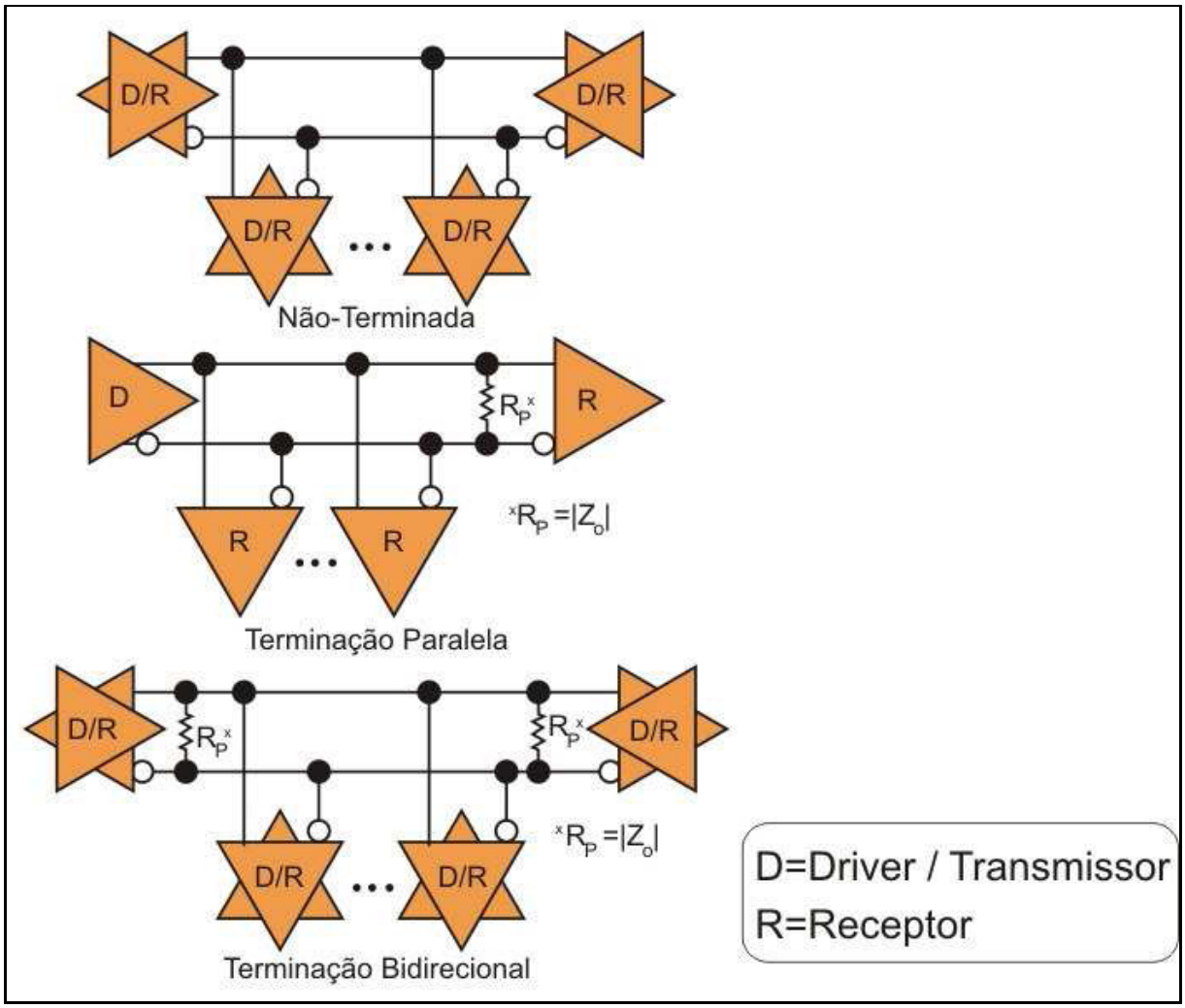


Figura 4 - Principais tipos de terminação
Fonte: Kron (2011)

Redes que usam o RS-485, como as DMX 512, normalmente são terminadas por resistores de 120 Ω colocados no final da linha de transmissão. Os dispositivos possuem uma chave de seleção, em que é possível selecionar se ele está entre o transmissor e o final da linha ou se ele é o final da linha (KRON, 2011; PERRIN, 1999).

As redes RS-485 podem utilizar diversas topologias, sendo a mais usada a *daisy chain* (em cadeia, tradução livre). Essa topologia é recomendada pois possui controle de reflexões e a possibilidade de uso de diversos nós da rede. Outras topologias também podem ser usadas, mas o controle de reflexões em uma rede estrela com diversos nós é complexo, pois as reflexões podem vir de qualquer ponto da rede e descobri-lo é lento e trabalhoso (FILARDI, 2011; KRON, 2011; PERRIN, 1999; TANEMBAUM, 2003).

A figura 5 mostra algumas topologias e a recomendação de uso do RS-485:

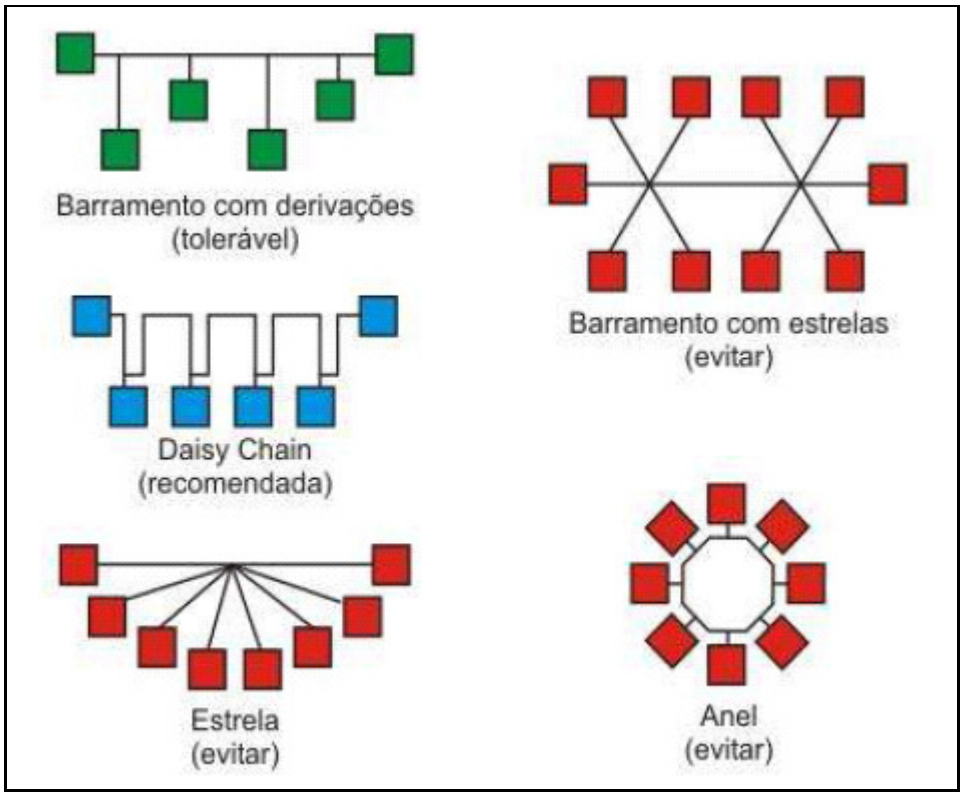


Figura 5 - Principais topologias de rede
Fonte: Kron (2011)

Filardi (2011) mostra, na figura 6, um exemplo de uma rede RS-485 usando topologia em cadeia (*daisy chain*) com seus respectivos resistores de terminação.

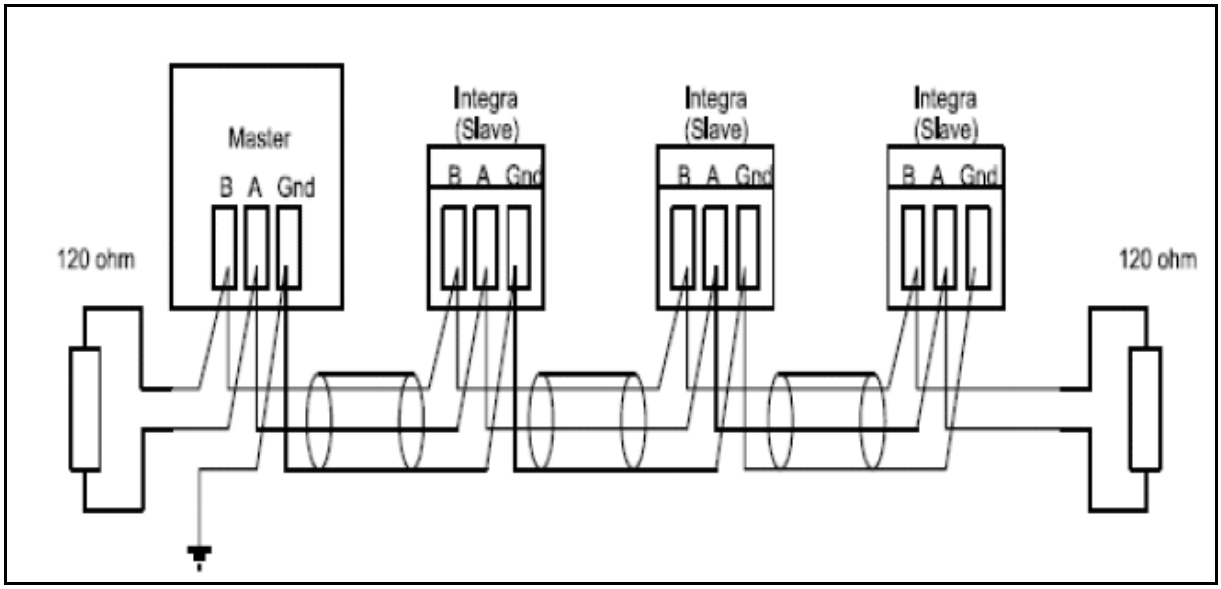


Figura 6 - Exemplo de topologia em cadeia com resistores de terminação
Fonte: Filardi (2011)

2.1.2 Formato e protocolo dos dados de comunicação DMX 512

O formato de dados do padrão DMX 512 é relativamente simples, e consiste em dados transmitidos de forma serial, em valores de 8 bits. Os dados são transmitidos em até 250kbps em quadros que consistem nas seguintes sessões (ERNST, 2008):

- *Break* – quebra – Composto de no mínimo 88 µs (22 ciclos – 22 bits) indica o estado de “0” da linha, no qual o sinal da linha “A+” é menor do que o sinal da linha “B-“ (ERNST, 2008);
- *Mark after break* (MAB) – marca após quebra – composto de 8 µs (2 ciclos – 2 bits), indica o estado de “1” da linha, no qual o sinal da linha “A+” é maior que o sinal da linha “B-“ (ERNST, 2008);
- *Start code data slot* – *slot* de início de código de dados – composto de 44 µs (11 ciclos – 11 bits). O *start* bit é sempre “0”. Os próximos 8 bits são a parte de dados do código de início (*start code*), sendo os 2 últimos bits usados para indicar a parada e são sempre “1”. Os dados do código de inicio normalmente são “1” (ERNST, 2008; PLASA, 2011);
- *Data slots* – *slots* de dados – composto de 44 µs (11 ciclos – 11 bits) é bastante similar ao *start* code, possuindo as mesmas características de *start* bit (1 bit, nível lógico “0”) e *stop* bits (2 bits, nível lógico “1”). Podem ser inseridos atrasos de até 1 segundo entre os *slots* de dados (que devem ir para o estado lógico “1”). Esses atrasos podem ser usados para outras tarefas de processamento (ERNST, 2008; PLASA, 2011);

A figura 7 ilustra como é o sequenciamento do quadro no protocolo DMX (ERNST, 2008):

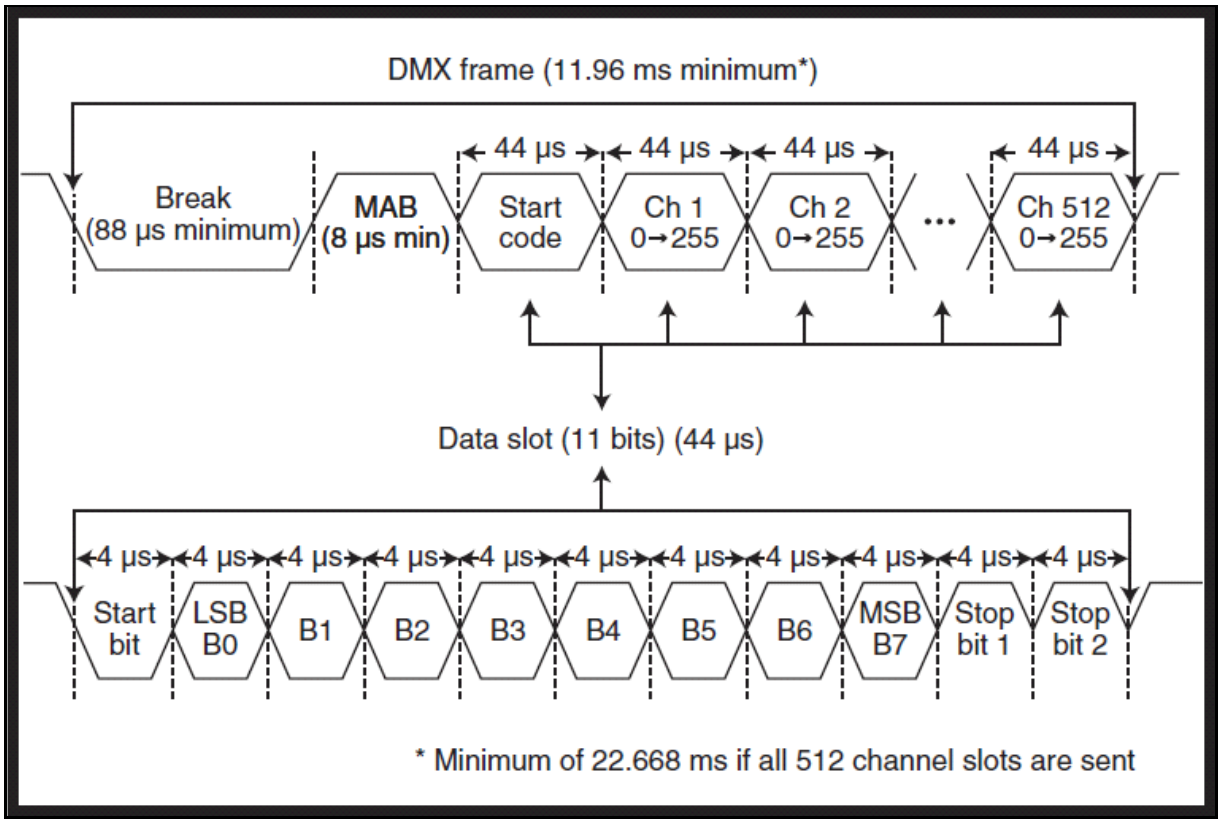


Figura 7 - *Frame* DMX 512
Fonte: Ernest (2008)

2.1.3 Tipo de conector (XLR-5)

Os conectores XLR-5 são comumente usados em equipamentos DMX 512, porém tem crescido o uso de conectores e cabos ethernet nesses sistemas (ERNST, 2008; KRON, 2011).

As fotografias 4, 5, 6 e 7 mostram quatro diferentes tipos de conectores XLR-5:

Fotografia 4 - Conector XLR-5 (macho)
Fonte: Futurlek (2011)

Fotografia 5 - Conector XLR-5 (fêmea)
Fonte: Futurlek (2011)

Fotografia 6 - Conector XLR-5 (macho)
Fonte: Futurlek (2011)

Fotografia 7 - Conector XLR-5 (fêmea)
Fonte: Futurlek (2011)

A figura 8 e o quadro 1 descrevem os pinos e as funções dos pinos no conector XLR-5:

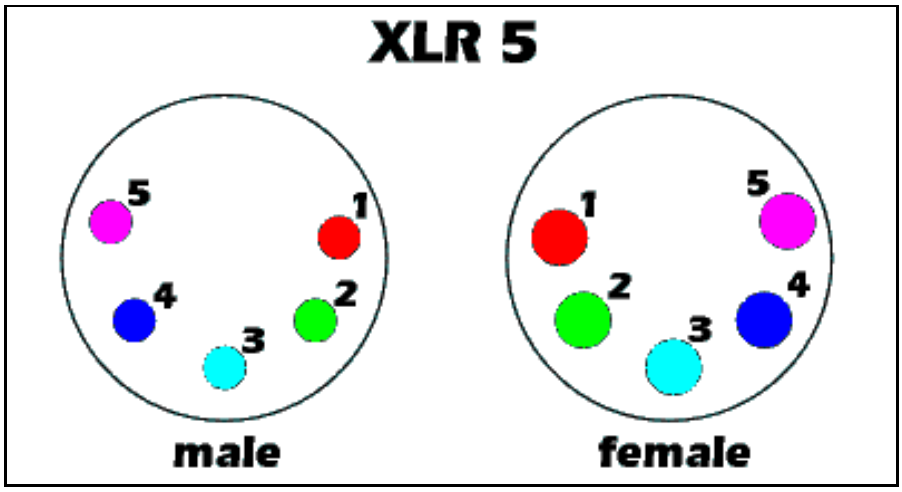


Figura 8 - XLR5 – vista através do cabo
Fonte: Baldwin (2011)

Pino	Função
1	Cabo blindado (terra)
2	Dados (-)
3	Dados (+)
4	Dados secundários (-)
5	Dados secundários (+)

Quadro 1 - Descrição dos pinos do XLR-5
Fonte: Adaptado de Baldwin (2011)

Como os dados secundários nem sempre são utilizados, alguns fabricantes chegaram a utilizar o conector XLR-3, que é usado para aplicações de áudio (como microfones). Isso pode gerar danos sérios devido às diferenças elétricas entre os sinais (ERNST, 2008).

2.2 MODULAÇÃO DIGITAL

A modulação digital, assim como a modulação analógica, consiste em alterar um sinal, que é conhecido como portadora, com um sinal que se deseja transmitir, que é o sinal modulante para que seja transmitido através de uma mídia (LANGTON, 2005; LINEAR, 2004).

Existem diversas técnicas de modulação, e a figura 9 ilustra técnicas que podem ser usadas (LINEAR, 2004):

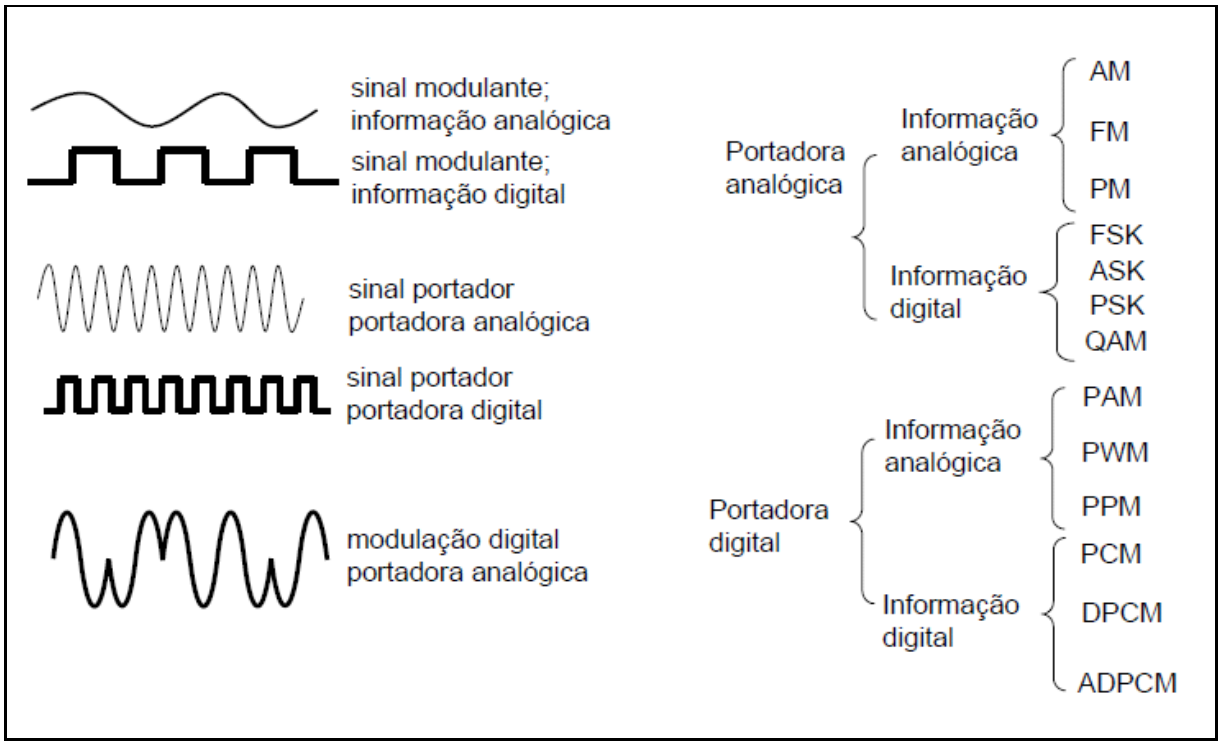


Figura 9 - Sinais, portadoras e técnicas de modulação
Fonte: Linear (2004)

As técnicas de modulação alteram uma ou mais características físicas da portadora para enviar a informação para o receptor. A característica alterada pode ser (LANGTON, 2005):

- A amplitude – modulação de informação digital ASK – *Amplitude-shift keying* – Chaveamento por amplitude;
- A frequência – modulação de informação digital FSK – *Frequency-shift keying* – Chaveamento por frequência;
- A fase – modulação de informação digital PSK – *Phase-shift keying* – Chaveamento por fase;
- A fase e a amplitude – modulação de informação digital QAM – Chaveamento por fase e amplitude

2.2.1 Modulação ASK

Na modulação ASK, como mencionado, a característica da portadora alterada é a amplitude. A forma mais simples do ASK é chamado de OOK (*on-off*

keying – chaveamento liga desliga – tradução livre) no qual o bit “1” é representado pela presença da portadora, enquanto o bit “0” é representado pela ausência da portadora (LANGTON, 2005).

Os gráficos 3, 4 e 5 representam a portadora senoidal, o sinal modulante e a portadora modulada OOK (LANGTON, 2005).

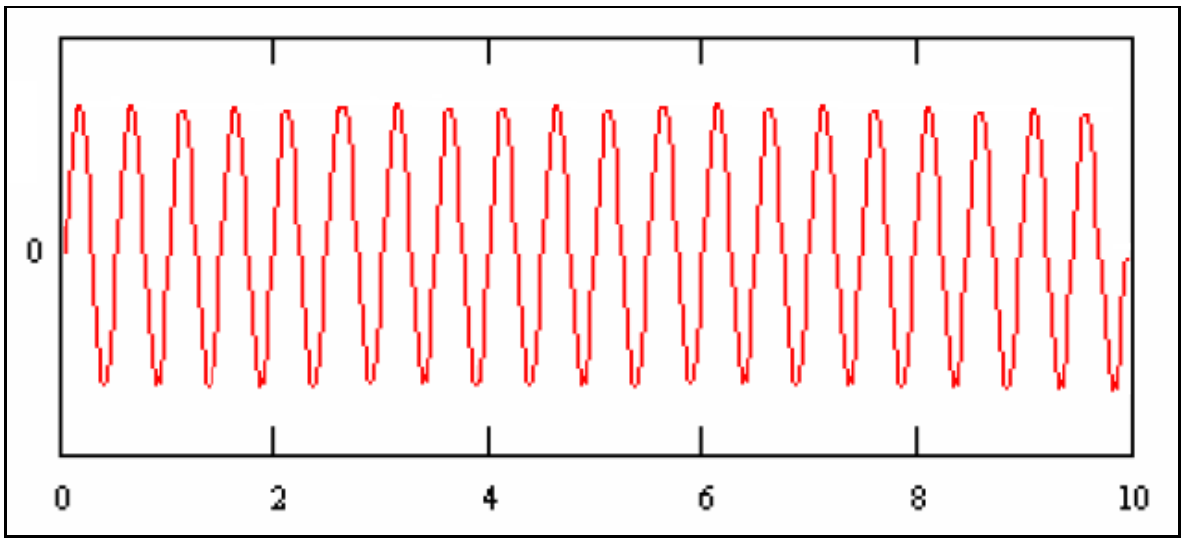


Gráfico 3 - Portadora senoidal
Fonte: Langton (2005)

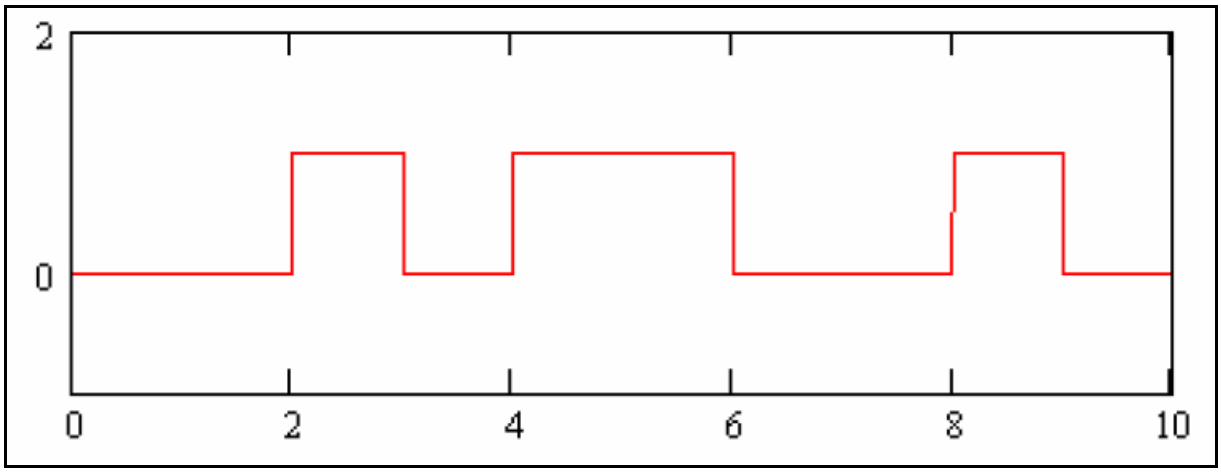


Gráfico 4 - Sinal modulante (0010110010)
Fonte: Langton (2005)

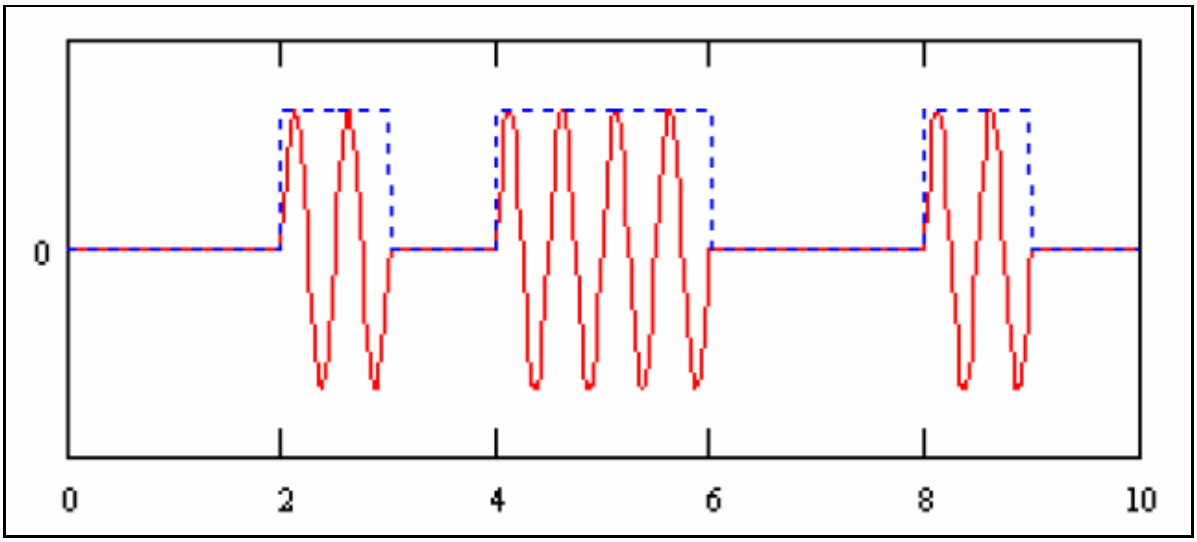


Gráfico 5 - Portadora modulada – modulação OOK
Fonte: Langton (2005)

2.2.2 Modulação FSK

A modulação FSK é bastante similar à modulação ASK, porém, ao invés de apenas uma frequência de portadora, existem duas frequências de portadora. O gráfico 6 ilustra a portadora "*f0"*, que representa o bit "0" e a portadora *"f1"*, que representa o bit "1" (LANGTON, 2002; LANGTON, 2005).

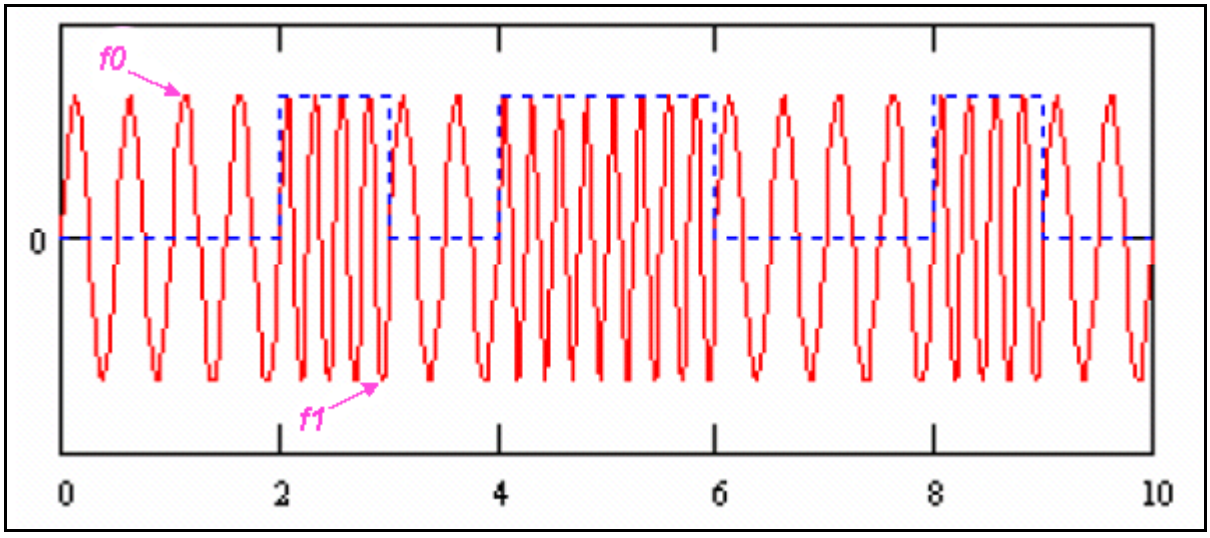


Gráfico 6 - Portadora modificada – modulação FSK
Fonte: Langton (2005)

As figuras 10 e 11 são diagramas de blocos do modulador e demodulador de sinais FSK (FARIAS, 2007).

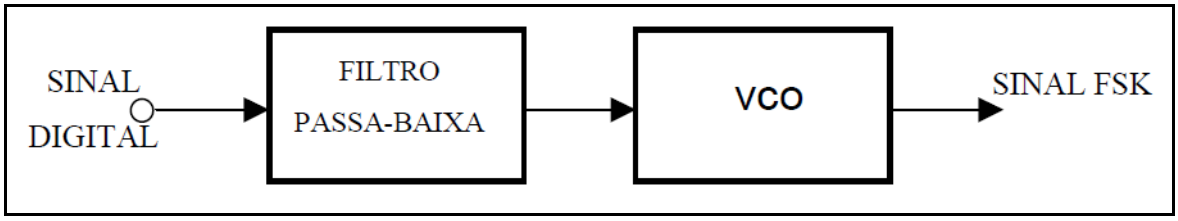


Figura 10 - Modulador FSK
Fonte: Farias (2007)

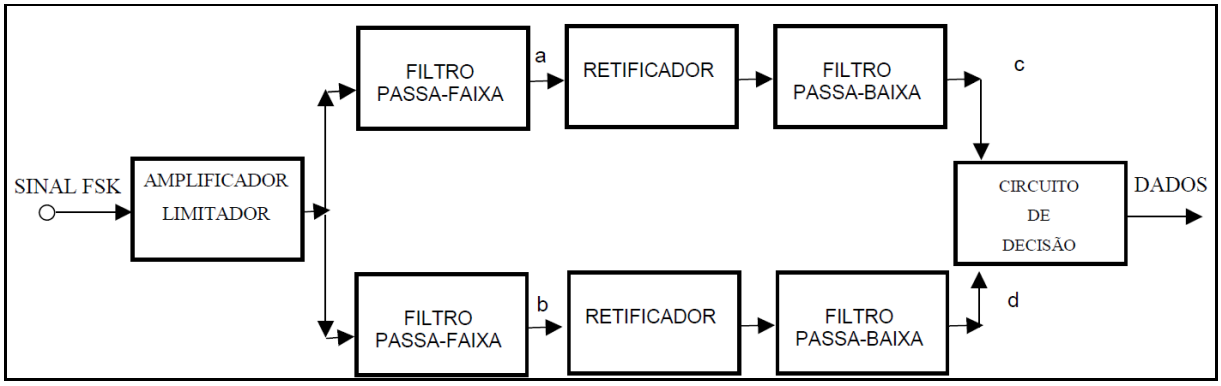


Figura 11 - Exemplo de demodulador FSK
Fonte: Farias (2007)

2.2.3 Modulação PSK

A modulação PSK altera a fase da portadora de acordo com a alteração da informação que precisa ser transmitida. A defasagem pode ser controlada, mas é bastante comum fase ser invertida em 180º, conforme pode ser observado no gráfico 7 (FARIAS, 2007; LANGTON, 2005):

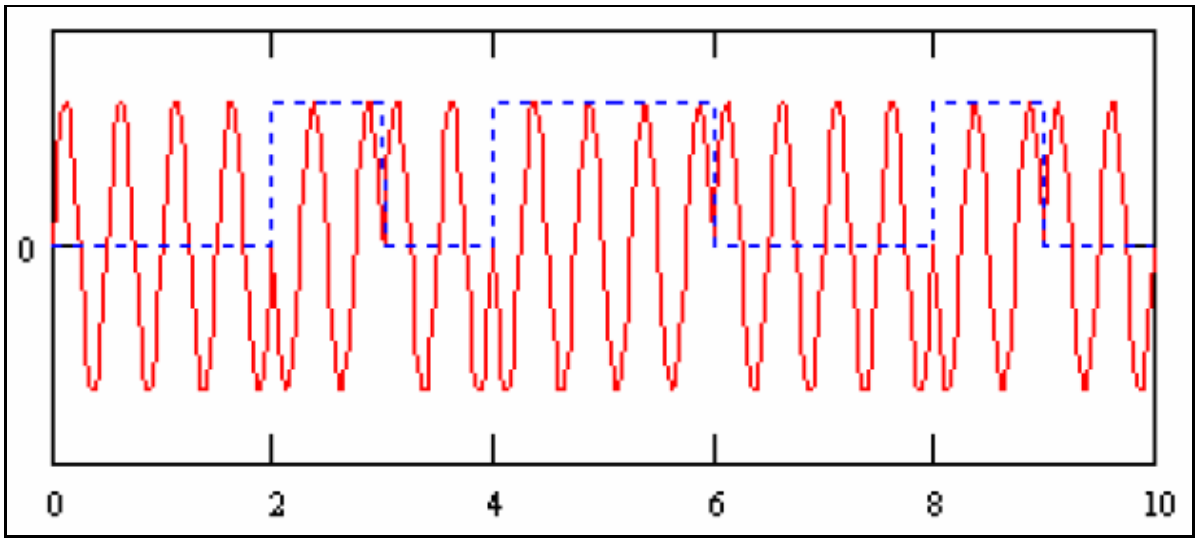


Gráfico 7 - Portadora modificada – modulação PSK
Fonte: Langton (2005)

2.2.4 Modulação QAM

A modulação QAM é uma combinação de duas técnicas de modulação, a modulação ASK e a modulação PSK, alterando o ângulo de fase e a amplitude simultaneamente (LANGTON, 2002; LANGTON, 2005).

A grande vantagem dessa técnica de modulação e enviar um sinal representando mais de 1 bit, aumentando a eficiência do canal de transmissão (FARIAS, 2007).

O quadro 2 apresenta a amplitude e deslocamento de fase, usados para representar 3 bits com a modulação QAM (CARVER, 2012).

Bit value	Amplitude	Phase shift
000	1	None
001	2	None
010	1	1/4
011	2	1/4
100	1	1/2
101	2	1/2
110	1	3/4
111	2	3/4

Quadro 2 - Relação entre bits x amplitude e fase – modulação QAM
Fonte: Carver (2012)

O gráfico 8 mostra a forma de onda gerada pela sequência “001-010-100-011-101-000-011-110” com a modulação QAM da figura 26 (CARVER, 2012).

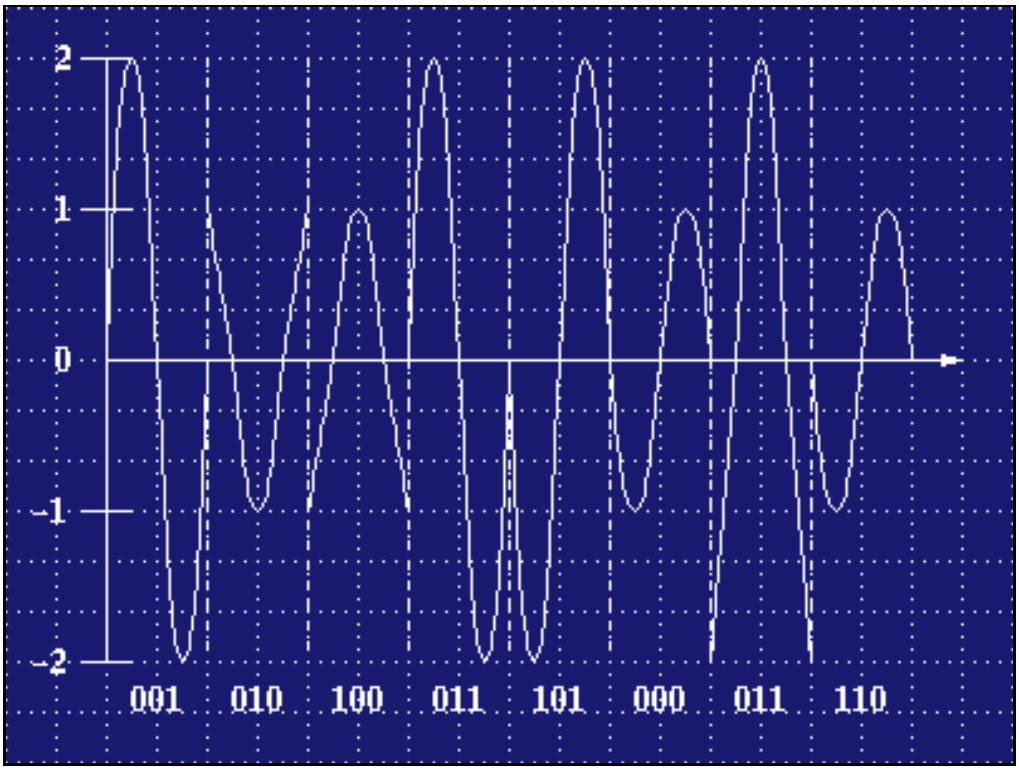


Gráfico 8 - Forma de onda do sinal QAM
Fonte: Carver (2012)

2.3 TEOREMA DE AMOSTRAGEM DE NYQUIST

O teorema de amostragem de Nyquist é fundamental para o desenvolvimento de conversores digitais e codificadores. Nyquist provou que se um sinal for amostrado com duas vezes a sua frequência, ele poderá ser totalmente reconstruído (CASTRO, 2012).

2.4 ZIGBEE / XBEE

2.4.1 Introdução ao ZigBee

A comunicação sem fios está presente em nossa sociedade há anos com as Redes WLAN (*Wireless Local Area Network*), WMAN (*Wireless Metropolitan Area Network*) WWAN (*Wireless Wide Area Network*), onde o objetivo é a transferência de grandes quantidades de dados e voz em altas velocidades (MESSIAS, 2012).

Atualmente existem inúmeros dispositivos e equipamentos que não requerem altas taxas de transmissão e necessitam de baixa latência e baixo consumo de energia (KINNEY, 2012). Dentre as redes WPAN (*Wireless Personal Area Network*) o padrão ZigBee IEEE 802.15.4 desenvolvido pela ZigBee Alliance junto ao IEEE (*Institute of Electrical and Electronics Engineers*) e outras várias outras empresas, propicia uma tecnologia de transmissão de dados com baixo consumo de energia, baixo custo, segurança e confiabilidade (MESSIAS, 2012).

O padrão ZigBee oferece comunicações robustas com operação na banda conhecida como ISM, que se refere a frequência utillizada por setores industriais, científicos e médicos (*Industrial, Scientist and Medical* - ISM), que operam entre 868 MHz e 2,4 GHz permitindo até 16 canais com taxas de transmissão entre 20 kbps a 250 kbps. Além disso, não requer licença para funcionamento, é capaz de hospedar milhares de dispositivos e mais de 65.000 redes, tem excelente imunidade contra interferências e ruído e possui baixo consumo de energia (MESSIAS, 2012).

A camada física foi concebida para ter um elevado grau de integração, simplificando o projeto de implantação reduzindo assim seu custo final. A camada de controle de acesso de mídia (*Media Access Control* – MAC) foi projetada para permitir várias topologias e dispositivos. A camada de rede permite o crescimento sem a necessidade de transmissores de alta potência, pois foi projetada para comportar uma grande quantidade de nós com latências relativamente baixas (KINNEY, 2012).

2.4.2 Características Gerais

Os módulos xbee possuem as seguintes características (KINNEY, 2012):

- Camada Física Dupla (2.4GHz e 868/915 MHz) Taxa de Dados 250 Kbps a 2.4 GHz, 40 Kbps a 915 MHz e 20 Kbps a 868 MHz;
- Otimizado para aplicações com baixo ciclo de trabalho (<0.1%);

- CSMA-CA (canal de acesso) – Elevada capacidade e baixa latência para dispositivos com ciclo de trabalho reduzido como sensores e controles;
- Baixo consumo (duração da bateria de vários meses a anos);
- Múltipla topologia: estrela, ponto-a-ponto, malha, árvore;
- Espaço de endereçamento de até $1{,}845 \times 10^{19}$ dispositivos (64 bits de endereçamento);
- 65.535 redes;
- Protocolo com alta confiabilidade de transferência;
- Alcance: 50m típico (5-500m dependendo do ambiente).

2.4.3 Modos de operação dos dispositivos xbee

Os dispositivos xbee possuem cinco modos de operação, conforme suas folhas de dados (XBEE..., 2009):

- Modo *idle* (modo de espera) – é o modo padrão de funcionamento dos módulos xbee. Nesse modo ele aguarda para entrar no modo de transmissão e recepção, modo *sleep* e modo comando;
- Modo de transmissão e recepção – nesses dois modos os módulos xbee podem enviar ou receber dados respectivamente;
- Modo *sleep* (modo de descanso / baixo consumo) – os módulos xbee podem ser programados para entrar em modo de baixo consumo (*sleep*), após um determinado tempo;
- Modo de comando – permite que sejam programadas as funções do xbee, tipo de criptografia, modulação, modo de transmissão, etc.;

A figura 12 ilustra os modos de operação dos módulos xbee:

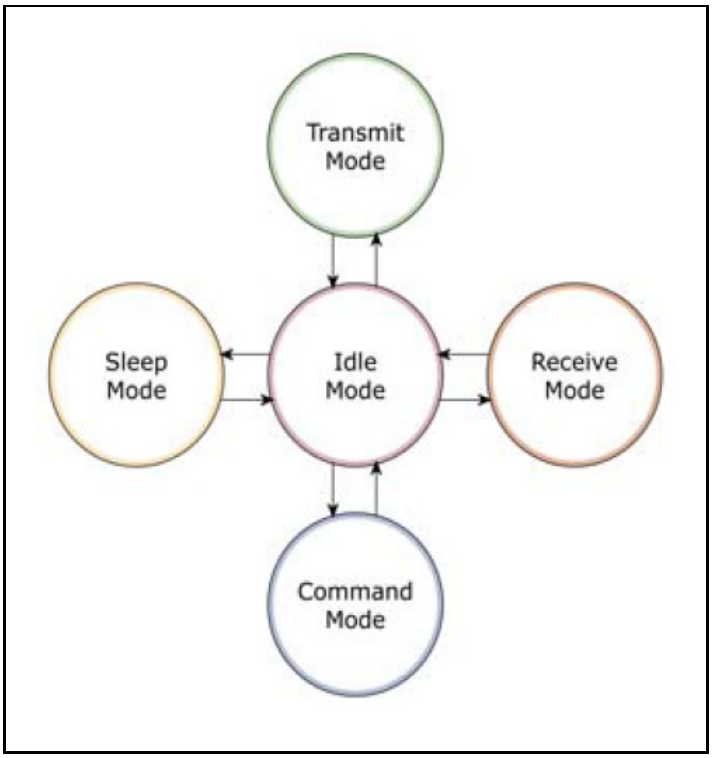


Figura 12 - Modos de operação dos módulos xbee
Fonte: Xbee... (2007)

O envio e recepção de dados no modo de transmissão e recepção podem ser executados de três formas (XBEE..., 2007):

- Modo serial – os dados são enviados e recebidos usando o formato serial, usando níveis elétricos compatíveis com as entradas do xbee. O controle de fluxo acontecerá pelos controles RTS (*request to send* – requisição para enviar) e CTS (*clear to send* – pronto para receber). A figura 13 ilustra uma comunicação serial com controle de fluxo:

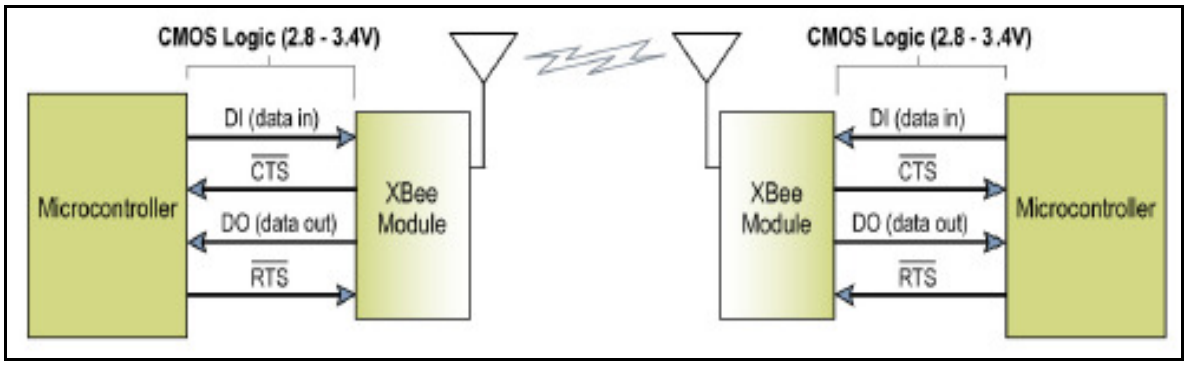


Figura 13 - Exemplo de comunicação serial com controle de fluxo
Fonte: Xbee... (2007)

- Modo transparente – modo de comunicação padrão do xbee. Os dados no formato serial assíncrono são recebidos, usando os níveis elétricos compatíveis com a entrada do xbee e não dependem de controle de fluxo. Uma vez que os dados chegam à porta de recepção do xbee, são transmitidos para o modulo de destino. O módulo de destino, assim que recebe os dados, coloca-os na porta de transmissão. A figura 14 ilustra uma comunicação serial sem controle de fluxo:

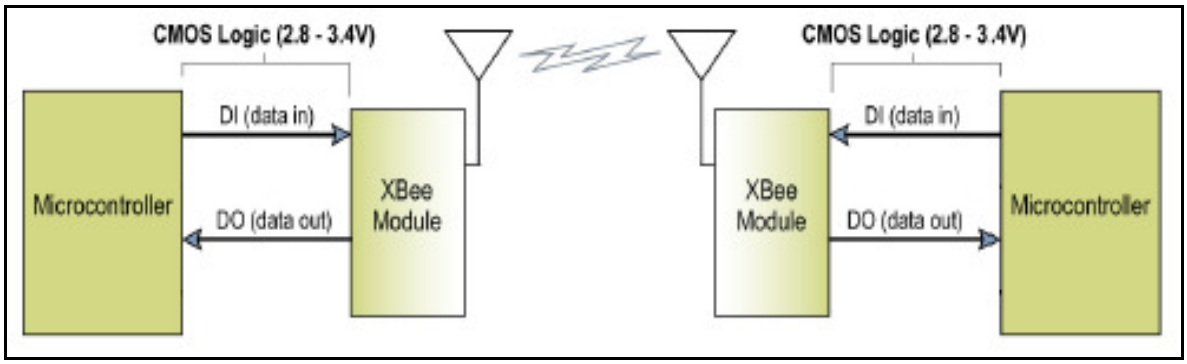


Figura 14 - Exemplo de comunicação serial sem controle de fluxo
Fonte: Adaptado de Xbee... (2007)

- Modo API (a*pplication programming interface* – interface de programação de aplicação) – Nesse modo, o dispositivo irá encapsular os dados para envio e recepção. A grande vantagem do modo API é que ele pode entrar em modo de comando, alterando parâmetros do modulo xbee enquanto envia ou recebe dados;

2.4.4 Topologias de rede

Em uma rede ZigBee são identificados dois tipos de dispositivos:

- FFD – *Full Function Device* (Dispositivos de Funções Completas): são dispositivos mais complexos e necessitam de um *hardware* mais potente para o uso da pilha de protocolos. Eles assumem as funções de Coordenador, Roteador ou até Dispositivo Final (*End Device*). Dispositivos FFD podem se comunicar com quaisquer dispositivos da rede (MESSIAS,

2012). Eles podem funcionar em qualquer topologia e podem conversar com qualquer outro dispositivo ZigBee (KINNEY, 2012).

- RDF – *Reduced Function Device* (Dispositivos de Funções Reduzidas): são dispositivos com funções mais simples na qual sua pilha de protocolos é usada com o mínimo de recursos de *hardware*. Eles podem se comunicar apenas com dispositivos FFD (Coordenador ou Roteador). Numa rede ZigBee eles assumem o papel do dispositivo final (*End Device*) (MESSIAS, 2012). Ele é limitado à topologia em estrela, e se comunica apenas com o coordenador da rede (KINNEY, 2012).

As classes dos dispositivos lógicos em uma rede ZigBee definem o tipo de rede, são elas:

- Coordenador (*Coordinator*) – responsável pela inicialização, distribuição de endereços, manutenção da rede, reconhecimento dos nós, entre outras funções. Pode servir como ponte para outras redes ZigBee (MESSIAS, 2012).
- Roteador (*Router*) – tem a característica de um nó normal na rede, mas pode exercer a função de roteador intermediário de nós, sem a necessidade de um coordenador. Na prática o roteador pode ser usado para amplificar um sinal (MESSIAS, 2012).
- Dispositivo Final (*End Device*) – é onde é feita a conexão com os atuadores, sensores ou equipamentos externos à rede ZigBee (MESSIAS, 2012).

Uma rede IEEE 802.15.4 / ZigBee requer pelo menos um dispositivo de função integral como coordenador da rede. Os dispositivos terminais podem ser dispositivos de funcionalidade reduzida para reduzir o custo do sistema. Todos os dispositivos devem ter endereços de 64 bits, entretanto alguns dispositivos podem utilizar endereços curtos (16 bits) para reduzir o tamanho do pacote de dados (KINNEY, 2012).

Conforme a disposição dos elementos dentro de uma rede ZigBee, ela pode ser classificada em:

- *Mesh* (Malha ou Ponto-a-Ponto): Na topologia *mesh* a rede pode se ajustar automaticamente, tanto na sua inicialização, como na entrada ou

saídas de dispositivos na rede. A rede se auto-organiza para aperfeiçoar o trafego de dados. Com vários caminhos possíveis para a comunicação entre os nós, este tipo de rede pode abranger em extensão, uma longa área geográfica, podendo ser colocada numa fábrica com vários galpões distantes, em um controle de irrigação ou mesmo num prédio com vários andares (MESSIAS, 2012).

- *Cluster Tree* (Árvore): Semelhante à topologia de malha, uma rede em árvore, tem uma hierarquia muito maior e o coordenador assume o papel de nó mestre para a troca de informação entre os nós *Router* e *End Device* (MESSIAS, 2012).
- *Star* (Estrela): É uma das topologias de rede ZigBee mais simples de serem implantadas, é composta de um nó Coordenador, e quantos nós *End Device* forem precisos. Este tipo de rede deve ser instalado em locais com poucos obstáculos à transmissão e recepção dos sinais, como por exemplo, em uma sala sem muitas paredes ou locais abertos (MESSIAS, 2012).

A figura 15 ilustra as redes descritas anteriormente.

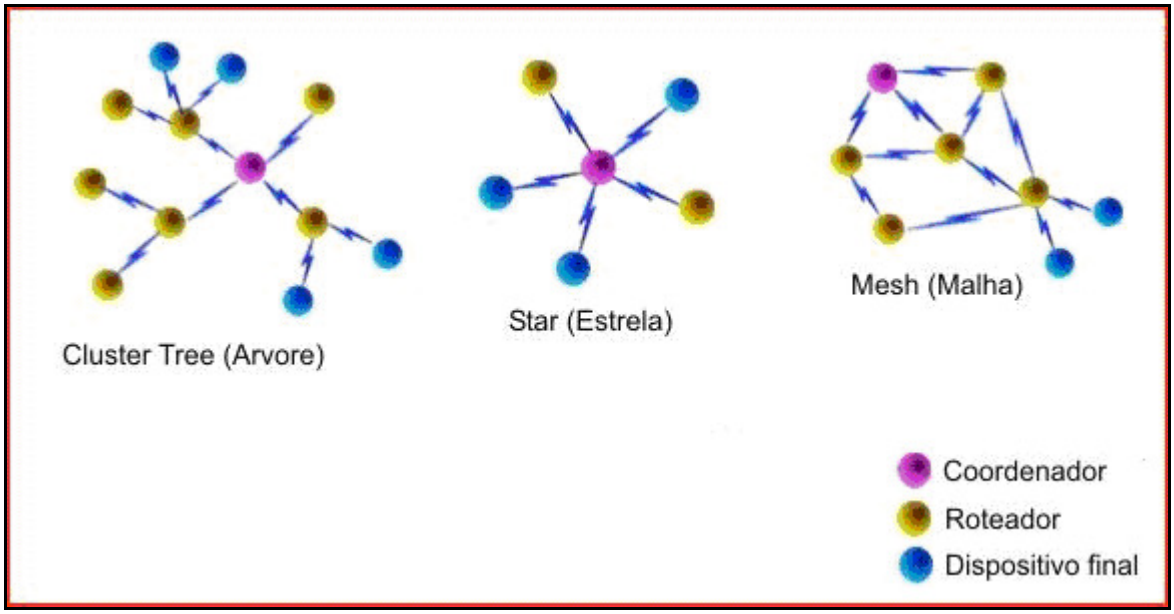


Figura 15 - Topologia de redes ZigBee
Fonte: Adaptado de Messias (2012)

3 PROJETO WI-FI DMX

Este projeto cria uma interface sem fio (*wireless*) para o padrão DMX 512, permitindo a comunicação entre a mesa controladora e o dispositivo controlado.

3.1 PREMISSAS

O trabalho foi iniciado medindo-se os sinais de saída da mesa de controle. Os gráficos 9, 10 e 11 demonstram a frequência, o intervalo entre informações e a tensão do sinal enviado pelo protocolo DMX

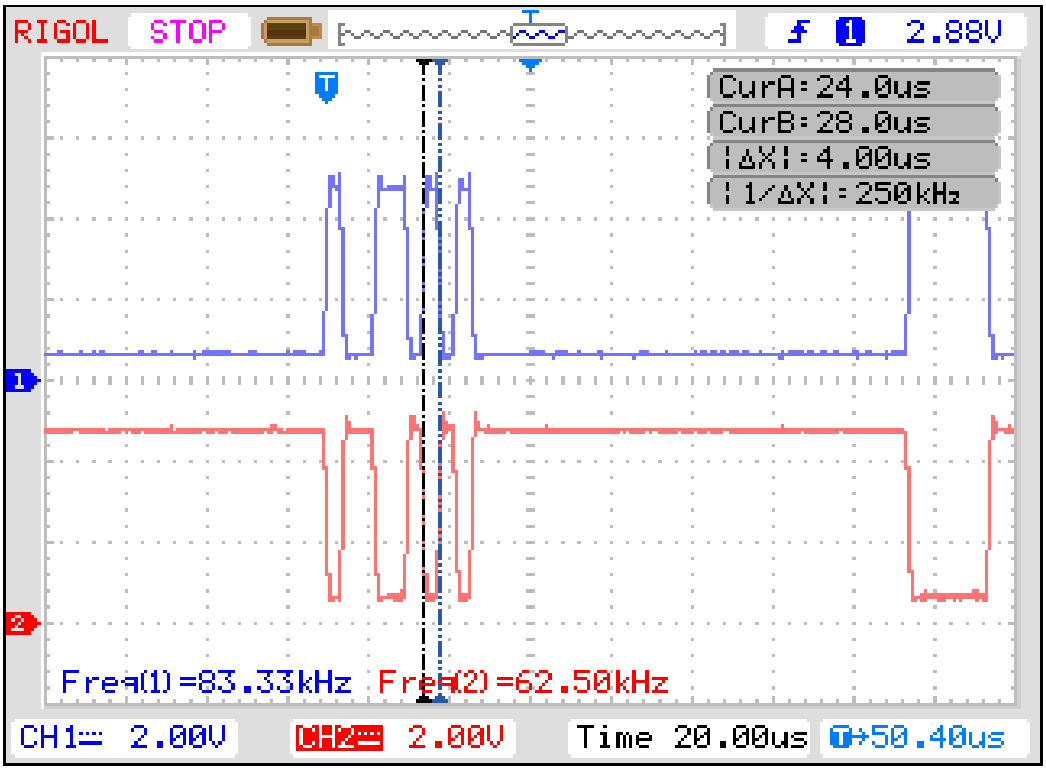


Gráfico 9 - Medição de frequência do sinal DMX-512 com osciloscópio
Fonte: Autoria própria

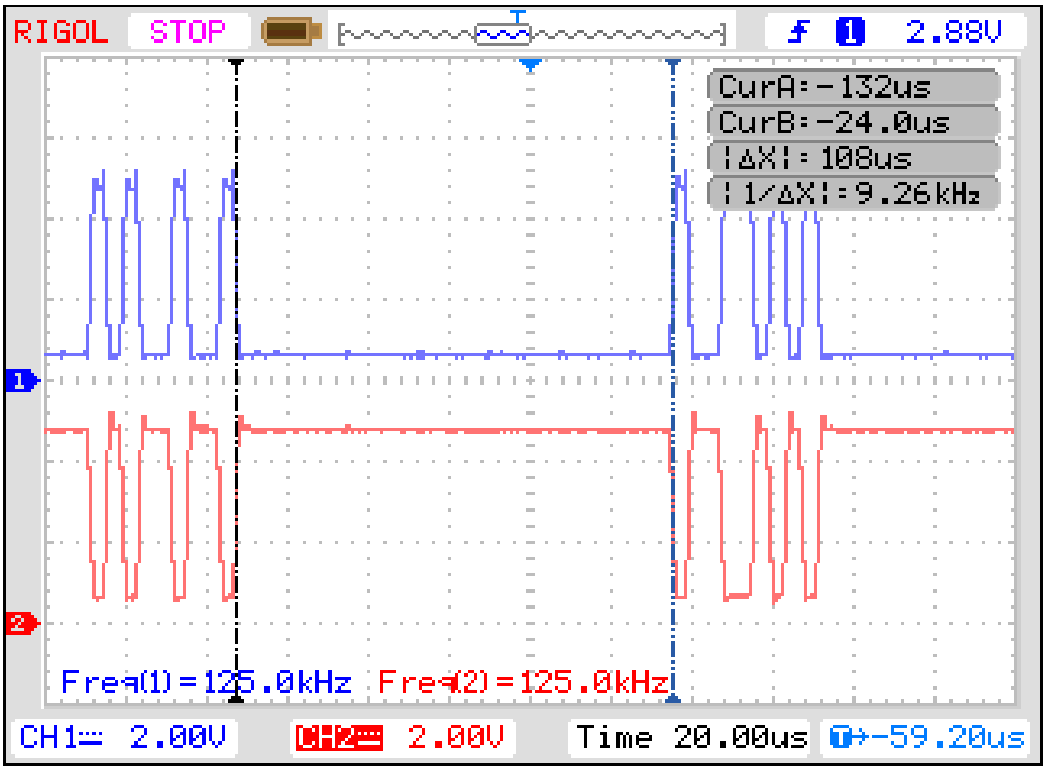


Gráfico 10 - Medição do intervalo entre informações do sinal DMX-512
Fonte: Autoria própria

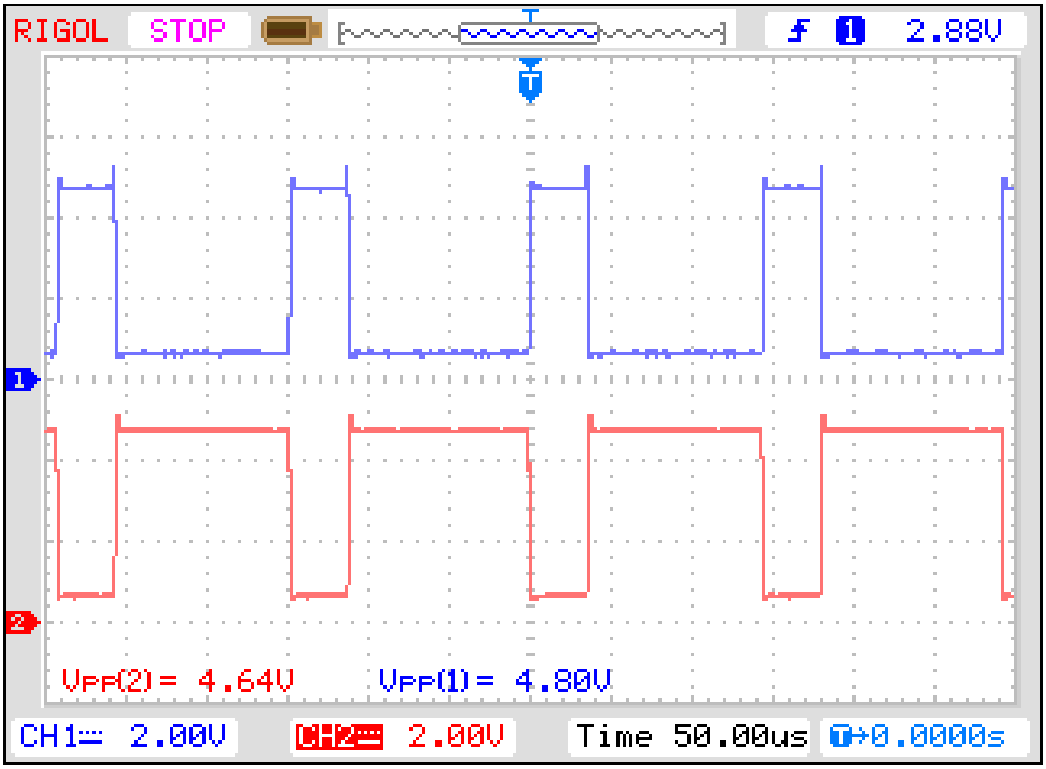


Gráfico 11 - Medição de tensão do sinal DMX-512
Fonte: Autoria própria

Analisando esses sinais, foram definidas algumas premissas do projeto:

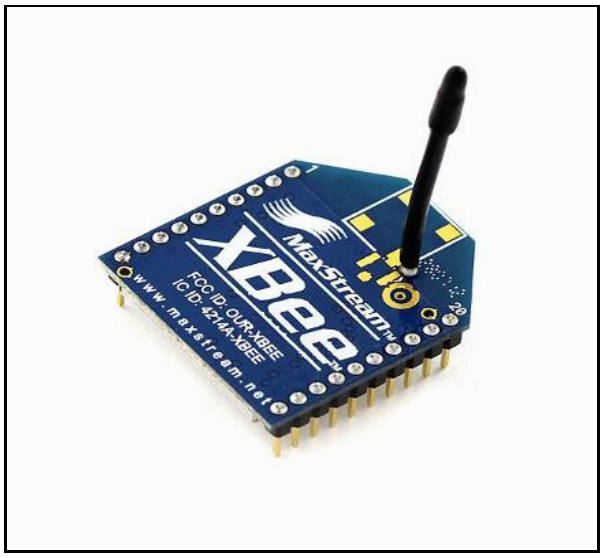
- O protótipo, para trabalhar sem uso de memória, precisa transferir os dados a uma taxa de 250kbps, pois essa é a taxa de transmissão usada pelos módulos DMX-512;
- O protótipo pode usar memória e micro-controlador, pois as informações não são enviadas de forma contínua, podendo ser memorizadas e transferidas a uma taxa menor que 250kbps. O tamanho do *buffer* deve ser grande o suficiente para que não sejam perdidas informações, e isso dependerá da taxa de dados que será transferida;
- As informações podem ser compactadas para que as informações sejam transmitidas a uma taxa menor que 250 kbps;

O projeto tem ainda as seguintes características:

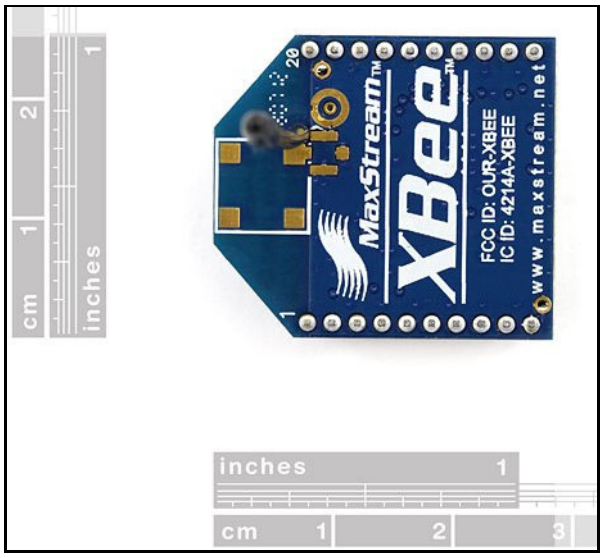
- O protótipo deve consumir uma potência menor que 250 mW, para que possa ser alimentado pela fonte do dispositivo controlador / controlado e / ou fontes recarregáveis / móveis, tendo autonomia para 10 horas contínuas de operação, que é o tempo médio de duração de um evento;
- O custo do protótipo não pode exceder a interfaces similares existentes no mercado;
- Deve possuir tamanho reduzido, devendo ser compatível com os dispositivos controlados;
- O alcance estimado deve ser grande o suficiente para justificar a ausência de cabos;

3.2 PROTÓTIPO XBEE

Após pesquisas realizadas no Apêndice C, escolheu-se o protótipo xbee para a montagem e testes. A fotografia 8 mostra o módulo xbee, e a fotografia 9 mostra o tamanho da interface xbee.



Fotografia 8 - Módulo xbee
Fonte: Sparkfun (2012)



Fotografia 9 - Tamanho da interface xbee
Fonte: Sparkfun (2012)

A primeira análise realizada foi à função de cada pino do módulo xbee, para que fossem definidos os circuitos necessários para a transmissão e recepção. As funções de cada pino foram colocadas no quadro 3:

Pino #	Nome	Direção	Descrição
1	VCC	-	Alimentação 3,3v
2	DOUT	Saída	Saída de dados da UART
3	DIN / $\overline{\text{CONFIG}}$	Entrada	Entrada de dados da UART
4	DO8*	Saída	Saída digital 8
5	$\overline{\text{RESET}}$	Entrada	Inicializa módulo (um pulso nível 0 de pelo menos 200ms)
6	PWM0 / RSSI	Saída	Saída do PWM 0 / Indicador de Força do sinal de RF (RX)
7	PWM1	Saída	Saída do PWM 1
8	(Reservado)	-	Ainda não tem uma função definida (futura implementação)
9	$\overline{\text{DTR}}$ / SLEEP_IRQ / DI8	Entrada	Linha de Controle da Função Sleep ou Entrada digital 8
10	GND	-	Terra
11	AD4 / DIO4	Entrada/Saída	Só Entrada Analógica 4 ou Entrada/Saída Digital 4
12	$\overline{\text{CTS}}$ / DIO7	Entrada/Saída	Controle de Fluxo CTS ou Entrada/Saída Digital 7
13	ON / SLEEP	Saída	Indicador de Estado do Módulo
14	VREF	Entrada	Tensão de Referência para as Entradas A/D
15	Associação / AD5 / DIO5	Entrada/Saída	Indicador de Associação, só Entrada Analógica 5 ou Entrada/Saída Digital 5
16	$\overline{\text{RTS}}$ / AD6 / DIO6	Entrada/Saída	Controle de Fluxo RTS, só Entrada Analógica 6 ou Entrada/Saída Digital 6
17	AD3 / DIO3	Entrada/Saída	Só Entrada Analógica 3 ou Entrada/Saída Digital 3
18	AD2 / DIO2	Entrada/Saída	Só Entrada Analógica 2 ou Entrada/Saída Digital 2
19	AD1 / DIO1	Entrada/Saída	Só Entrada Analógica 1 ou Entrada/Saída Digital 1
20	AD0 /DIO0	Entrada/Saída	Só Entrada Analógica 0 ou Entrada/Saída Digital 0

Quadro 3 - Funções dos pinos do xbee
Fonte: Adaptado de Messias (2012) e Xbee... (2007)

Os módulos do xbee são alimentados com tensão de 2,8 V a 3,3V. O xbee possui três modos de transmissão/recepção: O primeiro modo, chamado de modo serial, permite a transferência de dados seriais com uso de controle de fluxo. O segundo modo, chamado de modo AT ou modo transparente, permite transferir dados seriais, de um módulo para outro de forma transparente (sem controle de fluxo), de forma que um dado que chega ao pino de entrada e enviado e disponibilizado no pino de saída do outro módulo. O terceiro modo, chamado de modo API transfere pacotes de dados que precisam estar encapsulados (XBEE..., 2007).

O padrão elétrico do DMX-512 é o RS-485, encapsulada em *frames* DMX-512. Como os dados DMX-512 são bastante similares aos dados seriais, optou-se

por utilizar o modo transparente para a transferência de dados entre os módulos xbee. Com o modo transparente não há a necessidade de geração dos controles de fluxo CTS e RTS, simplificando os circuitos, (XBEE..., 2009). Considerando esses fatores, foram necessários os seguintes circuitos:

- Um circuito que fará a conversão elétrica dos sinais RS-485 para a entrada serial do módulo xbee (XBEE..., 2009);
- Um circuito que converterá os sinais de saída serial do módulo xbee para sinais RS-485 (XBEE..., 2009);
- Um circuito para gerar as tensões de alimentação necessárias para o funcionamento do protótipo, permitindo que ele seja alimentado externamente por uma tensão de 5 a 24 V, convertendo essa alimentação para os 5 V usados no micro-controlador e nos 3 V exigidos pelos módulos do xbee (XBEE..., 2009);
- Um circuito para a programação dos módulos xbee (SALEIRO; EY, 2008);

As escolhas e justificativas dos circuitos estão detalhadas no Apêndice C.

3.2.1 Circuito de alimentação

Para o circuito de alimentação usou-se reguladores de tensão baseados nos integrados LM-7805 e LM-317. O LM-7805 converte tensões maiores que 5 V em uma tensão de 5 V, sendo que a maior tensão admissível no LM-7805 é de 35 V, usando dissipadores. A corrente de saída do LM-7805 é de até 1 A, que é suficiente para a alimentação dos circuitos, uma vez que eles devem possuir boa autonomia caso sejam utilizadas baterias ou pilhas (FAIRCHILD, 2011; FAIRCHILD, 2012).

O LM-317 será polarizado para que sua saída gere a tensão de 3,0 V necessária para a alimentação do xbee. A folha de dados do LM-7805 e do LM-317 recomenda os circuitos das figuras 16 e 17 (FAIRCHILD, 2011; FAIRCHILD, 2012):

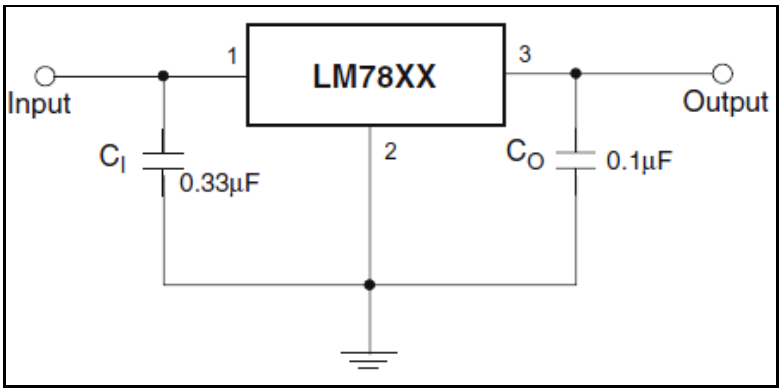


Figura 16 - Circuito recomendado LM-7805 – 5 VCC
Fonte: Fairchild (2012)

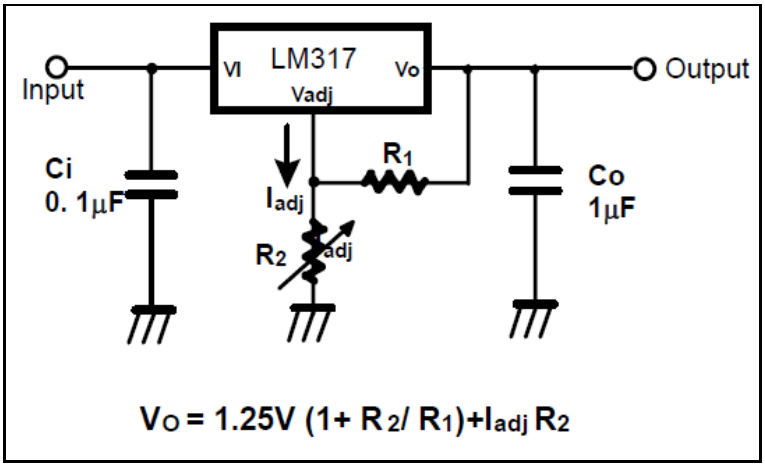


Figura 17 - Circuito recomendado pelo manual do LM-317
Fonte: Fairchild (2011)

O LM-317 precisa de resistores de polarização (R1 e R2), que foram calculados para que a tensão ficasse em aproximadamente 3,0 VCC na sua saída, conforme a figura 18.

$V_O = 1{,}25.(1 + R_2 / R_1) + I_{adj}.R_2$

De acordo com a tabela da folha de dados, $I_{adj} = 48\mu$A;
De acordo com a folha de dados do xbee $2{,}8 \leq V_O \leq 3{,}3$V; $V_O = 3{,}0$

$3{,}0 = 1{,}25.(1 + R_2 / R_1) + 48\mu.R_2 \rightarrow 3{,}0 = 1{,}25 + 1{,}25.(R_2 / R_1) + 48\mu.R_2 \rightarrow$
$1{,}75 = 1{,}25.(R_2 / R_1) + 48\mu.R_2 \rightarrow 1{,}75 = R_2.((1{,}25 / R_1) + 48\mu)$

$$R_2 = \frac{1{,}75}{((1{,}25 / R_1) + 48\mu)}$$

Atribuindo 3,3 kΩ para R_1, temos :

$$R_2 = \frac{1{,}75}{((1{,}25/3300) + 48\mu)} \rightarrow R_2 = \frac{1{,}75}{\left(0{,}427.10^{-3}\right)} = 4{,}1k\Omega$$

Valor comercial de $R_2 = 3{,}9k\Omega$, recalculando V_O, temos :

$$V_O = 1{,}25.(1 + 3900/3300) + 48\mu.3900 \rightarrow V_O = 1{,}25.(2{,}182) + 187{,}2.10^{-3} = \underline{2{,}914V}$$

Figura 18 - Cálculos dos resistores de polarização do LM-317
Fonte: Fairchild (2011)

Juntando as informações dos dois circuitos, tem-se o circuito de alimentação utilizado pelos circuitos, na figura 19.

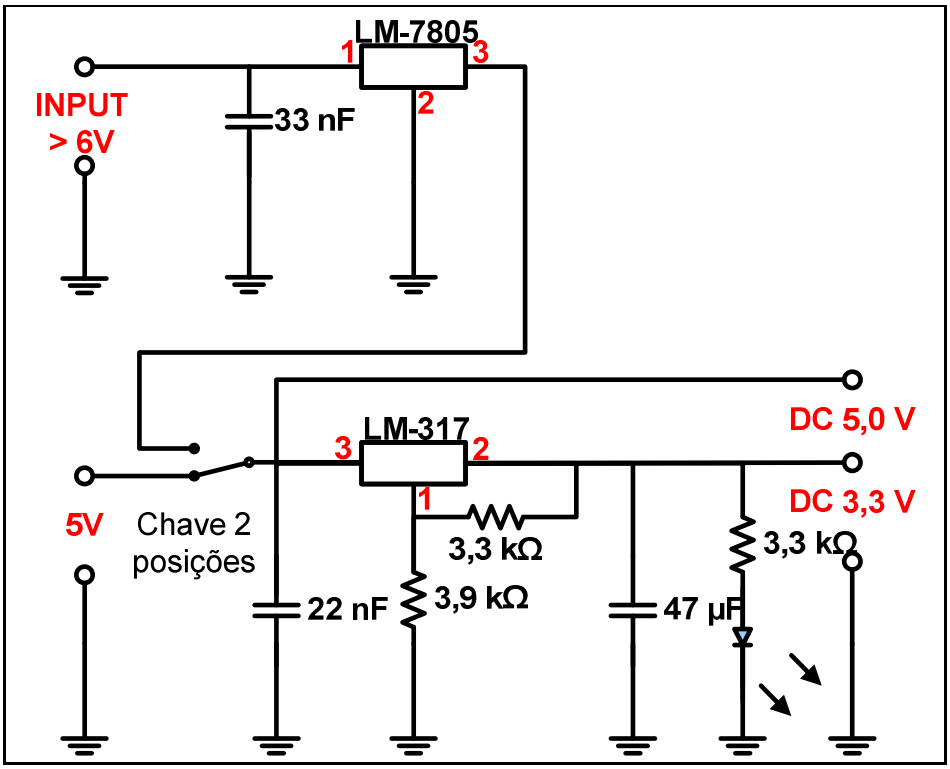


Figura 19 - Circuito de alimentação do xbee
Fonte: Adaptado de Fairchild (2011) e Fairchild (2012)

3.2.2 Circuitos de transmissão

O protocolo DMX-512 utiliza o padrão elétrico RS-485. Os dados são seriais, porém possuem quadros específicos, conforme visto na figura 12 (ERNST, 2008). Dessa forma, é necessário que os quadros do sinal DMX sejam tratados e sejam gerados sinais seriais. Esses sinais seriais são então colocados na porta do módulo xbee, respeitando os níveis de tensão utilizados pelos módulos.

O circuito de transmissão foi dividido em dois circuitos. O primeiro circuito consiste em um circuito de conversão do modo diferencial para o modo TTL. O segundo circuito consiste em um circuito de adequação dos quadros do DMX-512 para sinais seriais.

Para o primeiro circuito foi encontrado o circuito integrado MAX-485, que converte sinais RS-485 em sinais TTL. A figura 20 ilustra o diagrama elétrico simplificado do integrado:

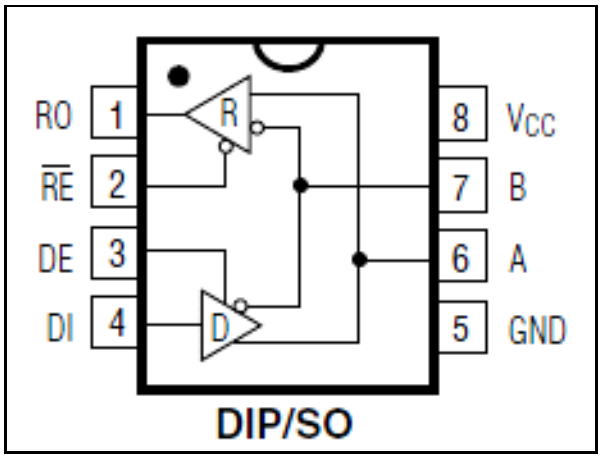


Figura 20 - Diagrama simplificado do MAX-485
Fonte: Maxim (2003)

Para o segundo circuito, que precisa de um tratamento dos quadros do DMX-512, utilizou-se o circuito micro-controlador PIC16F648A. Esse circuito visa:

- Conversão dos quadros (*frames*) DMX-512 em sinais seriais;
- Redução da velocidade de transmissão de 250 kbps para 115,2 kbps para maior estabilidade do circuito;

A figura 21 mostra os pinos do micro-controlador PIC16F648A:

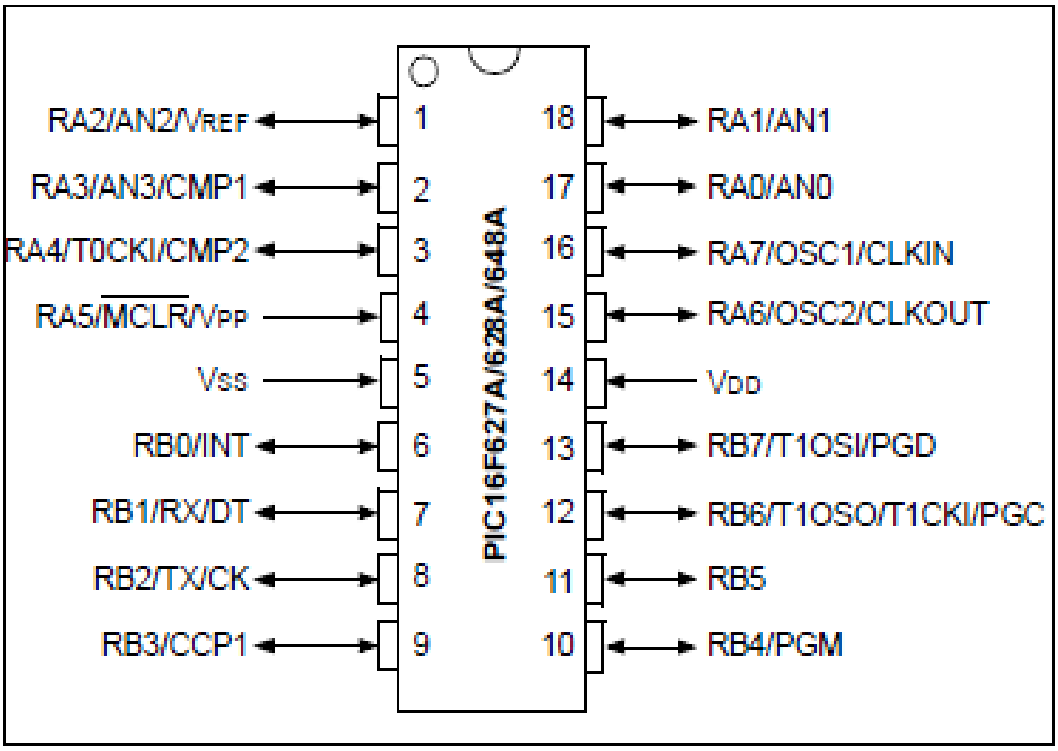


Figura 21 - Portas PIC16F648A
Fonte: Microchip (2006)

O PIC foi programado utilizando a linguagem C e a programação utilizada encontra-se no item 3.2.7 e no Apêndice E. O sinal de saída do PIC16F648A foi colocado em um divisor de tensão para que a tensão ficasse em 2.5 V. Essa tensão fica acima de 2,1 V, que são os 70% dos 3,0 V do sinal de VCC do xbee recomendados pelo manual. Usando dois resistores de mesmo valor, tem-se o valor de aproximadamente 2,5 V na entrada do xbee. A corrente de entrada do xbee é muito baixa, pois sua porta possui impedância de entrada muito alta (acima de um mega ohm) e por esse motivo a impedância de entrada foi desconsiderada para o cálculo dos resistores do divisor de tensão (MAXIM, 2003; MICROCHIP, 2006).

Dessa forma foi definido o circuito que converte o sinal diferencial de entrada do RS-485 para a entrada do xbee, conforme a figura 22:

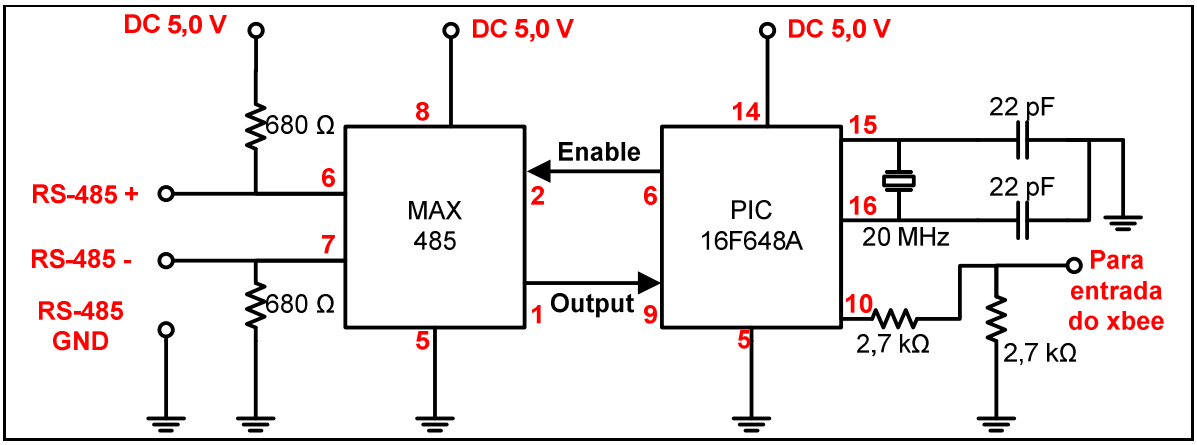


Figura 22 - Circuito usado no transmissor
Fonte: Adaptado de Maxim (2003) e Microchip (2006)

Foi adicionado ao circuito um pino de controle (*enable*) que durante a recepção do sinal no MAX-485 ficará em “0”. O circuito funcionou corretamente e o sinal de entrada foi devidamente transmitido pelo xbee em modo AT após sua programação.

3.2.3 Circuito receptor

O circuito receptor, assim como o transmissor, precisou ser dividido em duas partes. A primeira parte consiste em um circuito que possui duas finalidades:

- Conversão dos sinais seriais em quadros (*frames*) DMX-512;
- Aumento da velocidade de transmissão de 250 kbps para 115,2 kbps para que o sinal siga o padrão DMX-512;

Para ajustar a velocidade de saída do módulo xbee (115,2 kbps) para os 250 kbps o circuito micro-controlador irá memorizar um número específico de canais e transmitir, remontando o quadro do sinal DMX-512. Como o micro-controlador não possui memória suficiente para memorizar todos os canais foi determinado um número de canais para memorização. Como o dispositivo controlado possuía 12 canais, memorizou-se 12 canais, porém o micro-controlador possui memória para um número maior de canais, conforme pode ser observado no quadro 4:

		PIC16LF628A	PIC16LF648A
Clock	Maximum Frequency of Operation (MHz)	20	20
Memory	Flash Program Memory (words)	2048	4096
	RAM Data Memory (bytes)	224	256
	EEPROM Data Memory (bytes)	128	256
Peripherals	Timer module(s)	TMR0, TMR1, TMR2	TMR0, TMR1, TMR2
	Comparator(s)	2	2
	Capture/Compare/ PWM modules	1	1
	Serial Communications	USART	USART
	Internal Voltage Reference	Yes	Yes
Features	Interrupt Sources	10	10
	I/O Pins	16	16
	Voltage Range (Volts)	2.0-5.5	2.0-5.5
	Brown-out Reset	Yes	Yes
	Packages	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN	18-pin DIP, SOIC, 20-pin SSOP, 28-pin QFN

Quadro 4 - Características do PIC16F648A
Fonte: Microchip (2006)

O sinal de saída do PIC16F648A e o sinal de controle foram colocados no MAX-485, gerando o circuito da figura 23:

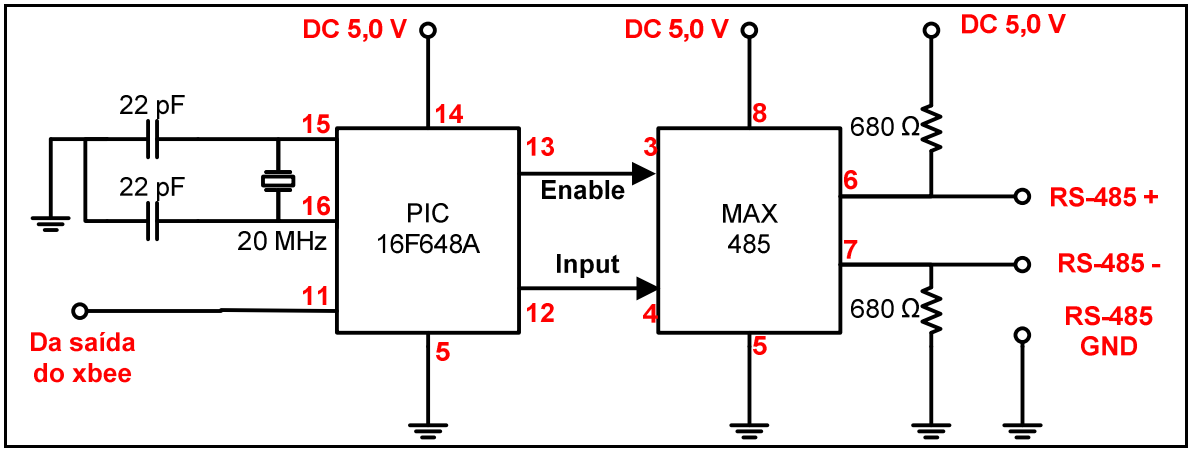


Figura 23 - Circuito usado no receptor
Fonte: Adaptado de Maxim (2003) e Microchip (2006)

O sinal de controle irá para “1” sempre que for iniciada uma sequência de quadros DMX-512, habilitando a entrada para a recepção dos dados e enviando os dados para as saídas do MAX-485. Os dados das saídas foram corretamente interpretados pelo dispositivo controlado.

3.2.4 Circuito para gravação

A finalidade desse circuito é poder efetuar a configuração do xbee para que ele funcione de acordo com as necessidades do projeto. O circuito para a gravação foi montado usando as abordagens que Saleiro e Ey (2008) fizeram, usando um *buffer* inversor com resistores de *pull up* como divisores de tensão de forma a ter aproximadamente 3,3 V nos sinais enviados pelas portas seriais.

A figura 24 demonstra o circuito que foi usado para definir os parâmetros dos módulos *xbee*.

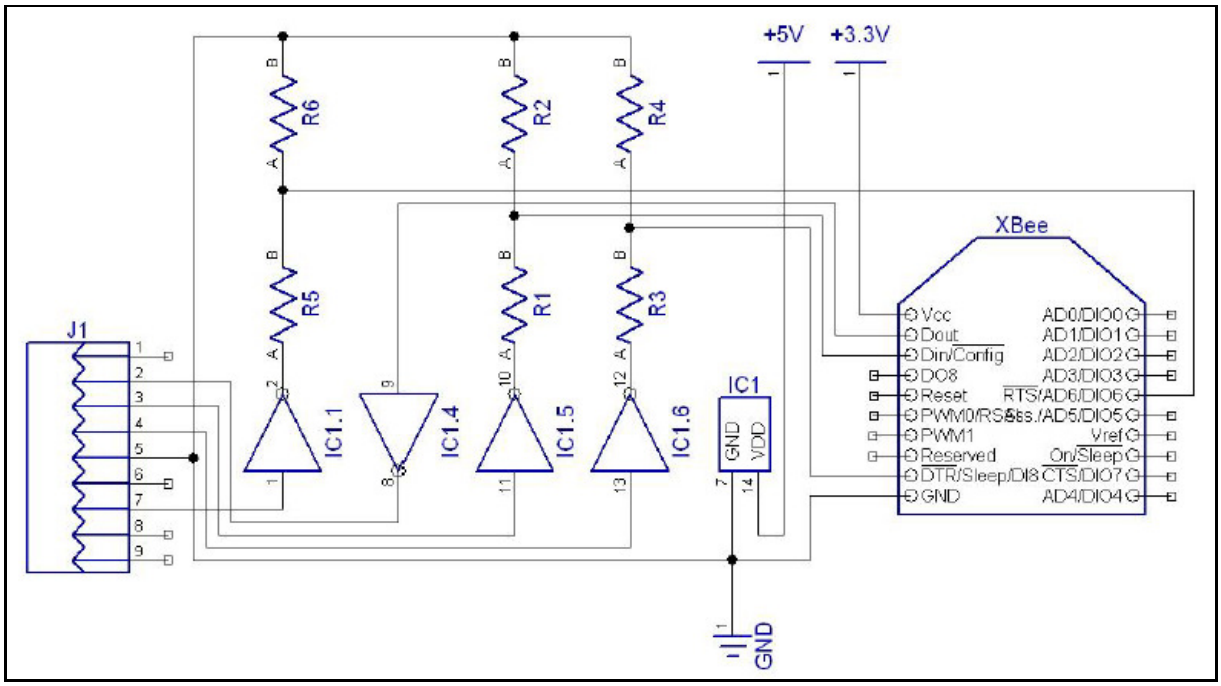


Figura 24 - Circuito de configuração do xbee
Fonte: Saleiro e Ey (2008)

O quadro 5 apresenta a relação dos componentes utilizados no circuito de configuração dos módulos xbee:

Componente	Código	Valor	Quantidade
Resistores	R1, R3, R5	1,8 k ohm	3
Resistores	R2, R4, R6	3,3 k ohm	3
Circuito Integrado	IC1	74LS06	1
Soquete	n/a	10 pinos	2
Conector	n/a	DB9 / RS 232	1
Cabo	n/a	UTP Cat 5	1 m

Quadro 5 - Relação de componentes para configuração
Fonte: Adaptado de Saleiro e Ey (2008)

O circuito foi montado em *proto-board*, usando uma fonte externa de 12 V e os circuitos integrados LM-7805 e LM317 descritos anteriormente no item 3.2.1 para fornecer as tensões requeridas. A configuração ocorreu sem maiores problemas e o circuito foi usado para todas as gravações dos módulos xbee.

A maior dificuldade de uso dessa interface de gravação é a disponibilidade de portas RS-232 em computadores atuais. Para a utilização da porta RS-232, foi necessário um *hardware* mais antigo que possuía portas seriais RS-232.

3.2.5 Circuito final

Para reduzir os custos de produção da placa de circuito impresso, uniu-se o circuito de transmissão com o circuito de recepção. Com isso, tem-se uma placa, que pode ser usada como transmissor, como receptor, ou ambas as funções, dependendo da programação feita nos microcontroladores e módulos xbee.

Para melhorar o diagnóstico foram adicionados dois LED's a cada módulo do xbee. Um deles indicará a comunicação entre eles (amarelo) e o outro indicará o modo de operação do xbee (vermelho). Foram usados os seguintes pinos do xbee (MESSIAS, 2012; XBEE..., 2007):

- Pino 1 – VCC – Alimentação 3,3V;
- Pino 2 – DOUT– *Digital Output*– Saída digital;
- Pino 3 – DIN – *Digital Input* – Entrada digital;
- Pino 10 – GND – *Ground* – Aterramento;
- Pino 13 – ON/SLEEP – *Power Mode* – Modo de funcionamento;
- Pino 15 – AI – *Association Indicator* – Indicador de associação de rede;

O circuito também previu uma chave para selecionar o tipo de alimentação que será utilizada no circuito. Na primeira posição da chave o circuito pode ser alimentado com qualquer tensão entre 5 e 35 V. Na segunda posição o circuito precisa de no mínimo 5 V de alimentação para garantir a alimentação dos componentes e do xbee. Foi colocado um LED verde como indicador de ligado / desligado do circuito.

O circuito final pode ser visto na figura 25 e com detalhes no Apêndice D – Circuito Final:

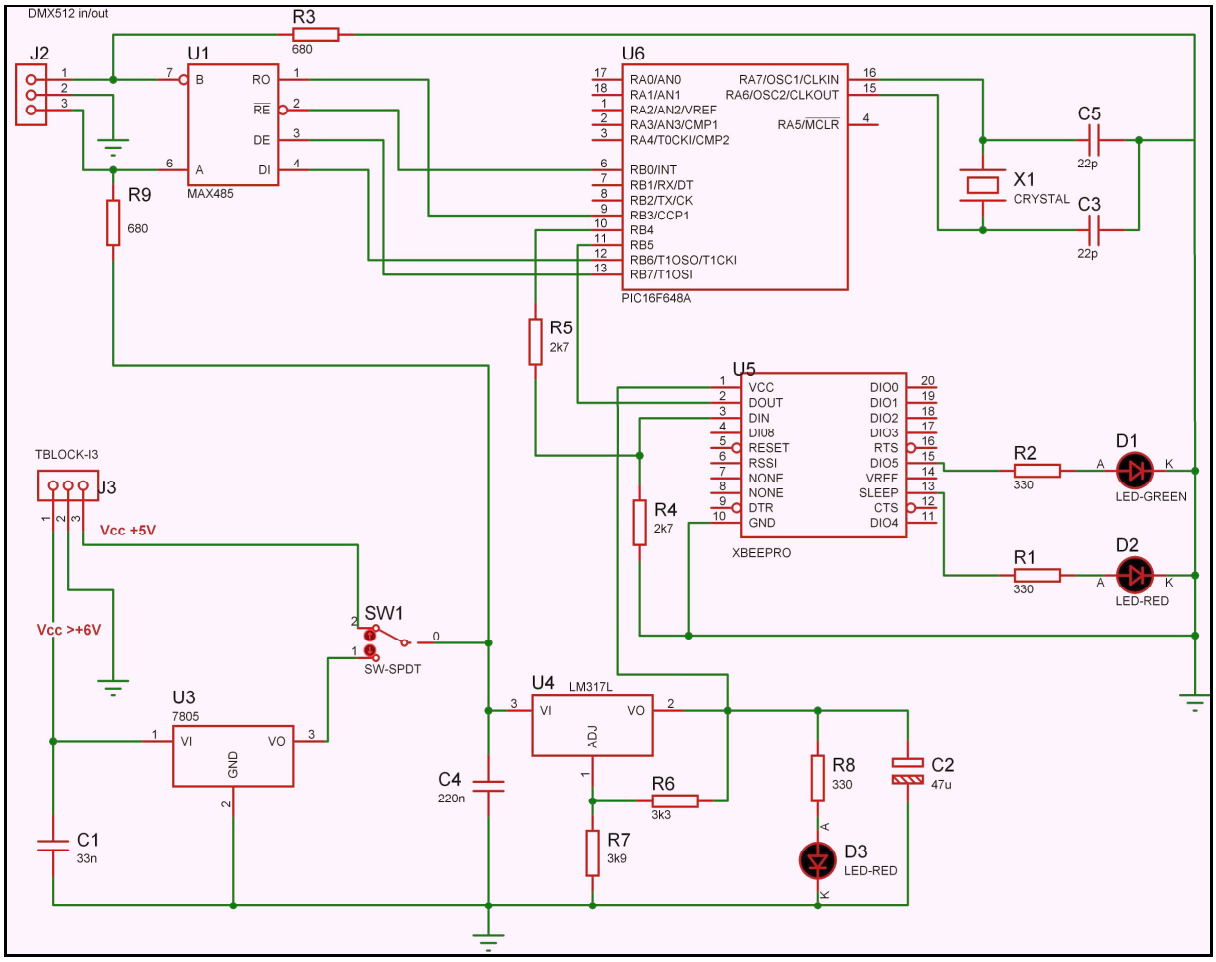


Figura 25 - Circuito final do protótipo xbee
Fonte: Autoria própria

Com base nesse circuito foi criado o protótipo xbee que está descrito no item 3.2.9. A relação de componentes usados pelo circuito e os custos estão descritos no item 3.2.6. O circuito final funcionou corretamente, porém é importante salientar que ele possui as seguintes limitações:

- O número de informações transmitidas foi reduzido devido à alteração na velocidade. Essa alteração não foi perceptível pelo dispositivo controlado;
- O número de canais transmitidos foi reduzido de 512 para 12, para simplificar a programação e melhorar o sincronismo. O número de canais pode ser maior (estimado 128 canais, usando esse micro-controlador). Como os dispositivos controlados normalmente não precisam de todos os canais de controle, essa alteração não teve impacto sobre o funcionamento em nossos testes, porém podem ter impacto caso se usem mais canais.

3.2.6 Custos do protótipo xbee

A tabela 1 demonstra os custos relativos ao protótipo xbee:

Tabela 1 - Custo do protótipo xbee feitos em Curitiba no ano de 2012

Componente	Quantidade	Custo Unitário	Custo Total
Capacitor 22 pF	4	R$ 0,17	R$ 0,68
Capacitor 33 nF	2	R$ 0,39	R$ 0,78
Capacitor 220 nF	2	R$ 0,32	R$ 0,64
Capacitor 47 µF	2	R$ 0,10	R$ 0,20
CI LM-317	2	R$ 1,40	R$ 2,80
CI LM-7805	2	R$ 1,28	R$ 2,56
CI MAX-485	2	R$ 6,44	R$ 12,88
CI PIC-16F648A	2	R$ 7,15	R$ 14,30
Cristal 20 MHz	2	R$ 0,75	R$ 1,50
Conector 3 pinos (externo)	4	R$ 2,54	R$ 10,16
Conector p/ MAX-485	2	R$ 0,97	R$ 1,94
Conector p/ PIC16F648A	2	R$ 1,25	R$ 2,50
Conector p/ xbee	4	R$ 1,20	R$ 4,80
Conector XLR-3 fêmea	4	R$ 1,95	R$ 7,80
Conector XLR-3 macho	4	R$ 3,00	R$ 12,00
LED Amarelo	2	R$ 0,12	R$ 0,24
LED Verde	2	R$ 0,10	R$ 0,20
LED Vermelho	2	R$ 0,10	R$ 0,20
Módulo xbee série 1	2	R$ 50,00	R$ 100,00
Placa de circuito impresso	2	R$ 35,00	R$ 70,00
Resistores 330 Ω	6	R$ 0,03	R$ 0,18
Resistores 680 Ω	4	R$ 0,03	R$ 0,12
Resistores 2,7 kΩ	4	R$ 0,03	R$ 0,12
Resistores 3,3 kΩ	2	R$ 0,03	R$ 0,06
Resistores 3,9 kΩ	2	R$ 0,03	R$ 0,06
Outros (fios, estanho, etc)	1	R$ 10,00	R$ 10,00
Custo total			R$ 256,72
Custo total por módulo			R$ 128,36

Fonte: Autoria própria

3.2.7 Programação do PIC16F648A

A programação do PIC16F648 foi desenvolvida em C. Iniciou-se a construção do programa com os fluxogramas dos programas que precisavam ser desenvolvidos, que podem ser vistos nas figuras 26 e 27:

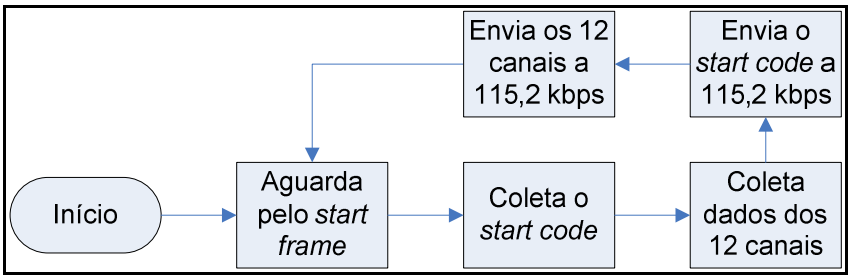


Figura 26 - Fluxograma simplificado do programa do transmissor
Fonte: Autoria própria

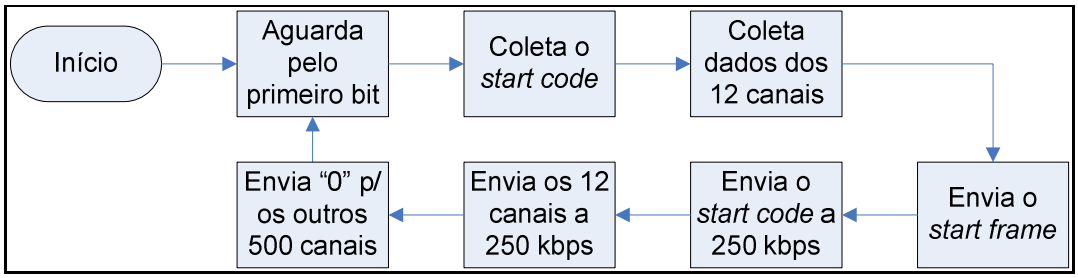


Figura 27 - Fluxograma simplificado do programa do receptor
Fonte: Autoria própria

O hardware e os softwares usados para a compilação e gravação dos PICs eram de propriedade da equipe, e possuem um custo estimado de R$ 200,00. O hardware e os softwares utilizados foram:

- O gravador Brenner 8 (USB) – hardware que permite que as instruções geradas pelo compilador CCS sejam gravadas no PIC;
- CCS PIC C Compiler – Compilador usado para converter a instruções em C em instruções que são aceitas pelo micro-controlador;
- US-Burn – Programa usado para a gravação do código gerado pelo CCS no micro-controlador PIC16F648A, usando o gravador PIC Brenner 8;

O código foi desenvolvido separadamente para o transmissor e receptor. Os dois códigos foram unificados. Esse novo código irá aguardar um tempo

determinado pelo *start frame* no pino 9 e depois desse tempo, se o *start frame* existir, irá coletar os dados. Se não existir irá aguardar um determinado tempo pelo *start bit* do *start code* no pino 11. Se essa condição também não for atendida o programa volta para o início, aguardando novamente pelo *start frame* no pino 9. O fluxograma simplificado do programa após a unificação está na figura 28:

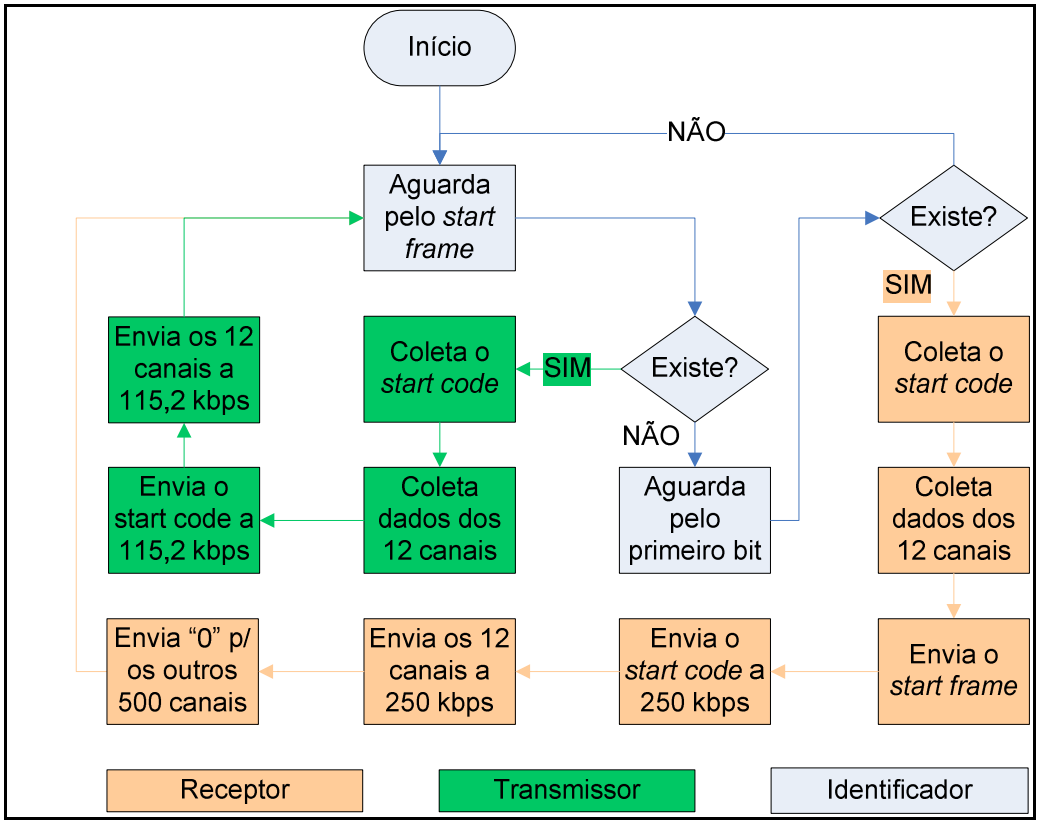


Figura 28 - Fluxograma simplificado do programa
Fonte: Autoria própria

Os blocos em azul descrevem os processos de identificação (transmissor ou receptor). Os blocos em verde descrevem os processos realizados pelo transmissor e os blocos em laranja descrevem os processos realizados pelo receptor. O código em C comentado do programa está no Apêndice E.

3.2.8 Configuração do módulo xbee

Para a configuração dos módulos xbee, além do hardware construído, foi necessário o uso do software X-TCU, que é disponibilizado pelo fabricante. Ele permite uma série de configurações para os módulos xbee (XBEE..., 2007).

Para o protótipo xbee é necessário configurar os módulos para que façam uma comunicação do tipo ponto a ponto, definindo os endereços dos módulos, das redes e demais características de operação, como o modo de transmissão (coordenador ou dispositivo final), velocidade de operação e tipo de criptografia. A figura 29 apresenta a programação definida no xbee *coordinator*.

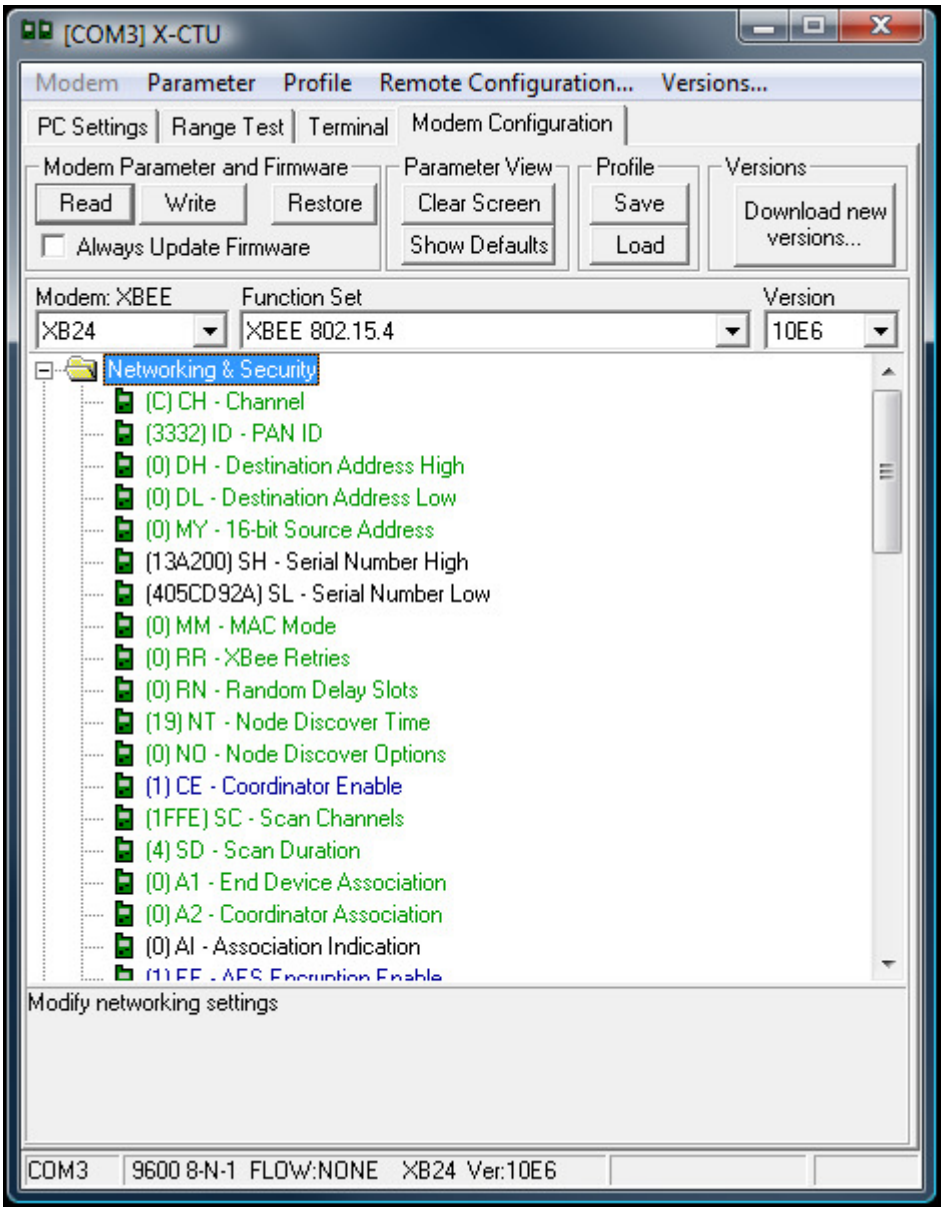


Figura 29 - X-CTU – Coordinator configuration
Fonte: Autoria própria

A figura 30 demonstra como ficou a programação da interface do *end device*:

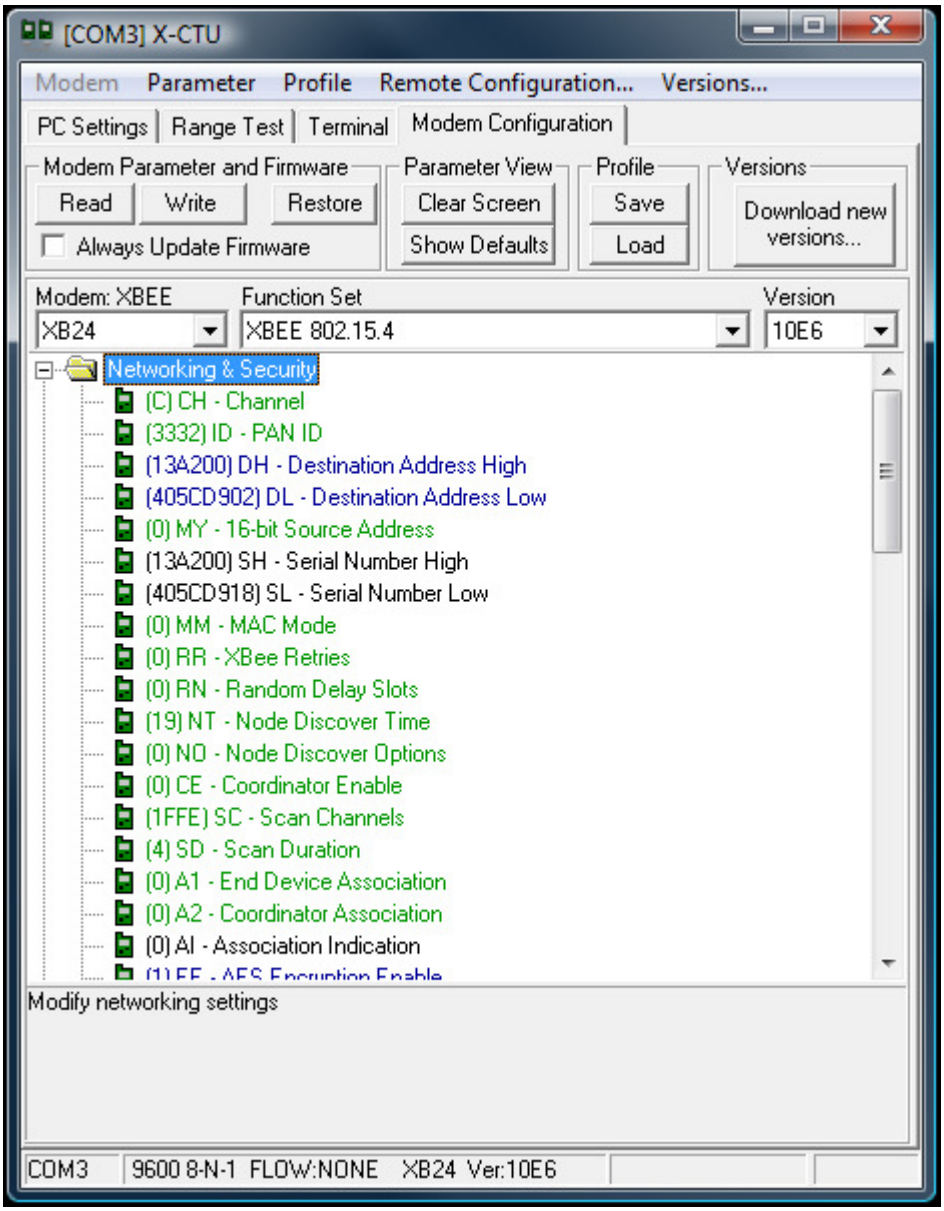


Figura 30 - X-CTU – End device configuration
Fonte: Autoria própria

A relação com os parâmetros alterados nos módulos xbee estão descritas no quadro 6:

Coordenador (coordinator)	
Dado	**Valor configurado**
PAN ID	3332
SH - Serial Number High	13A200
SL - Serial Number Low	405CD92A
DH - Destination Address High	13A200
DL - Destination Address Low	405CD918
Coordinator enable	1
AES encryption Enable	AES
KY - AES Encryption key	BABAFACACAFEDAD0DED0FEDEF0DEF1CA
BD - Bandwidth	7 - 115,2 kbps
Dispositivo Final (enddevice)	
Dado	**Valor configurado**
PAN ID	3332
SH - Serial Number High	13A200
SL - Serial Number Low	405CD918
DH - Destination Address High	13A200
DL - Destination Address Low	405CD92A
Coordinator enable	0
AES encryption Enable	AES
KY - AES Encryption key	BABAFACACAFEDAD0DED0FEDEF0DEF1CA
BD - Bandwidth	7 - 115,2 kbps

Quadro 6 - Parâmetros configurados nos módulos xbee
Fonte: Autoria própria

3.2.9 Placa de circuito impresso

O circuito definido anteriormente no item 3.2.5, foi passado para um software para auxiliar na confecção da placa de circuito impresso. O desenho da placa de circuito impresso pode ser visto na figura 31:

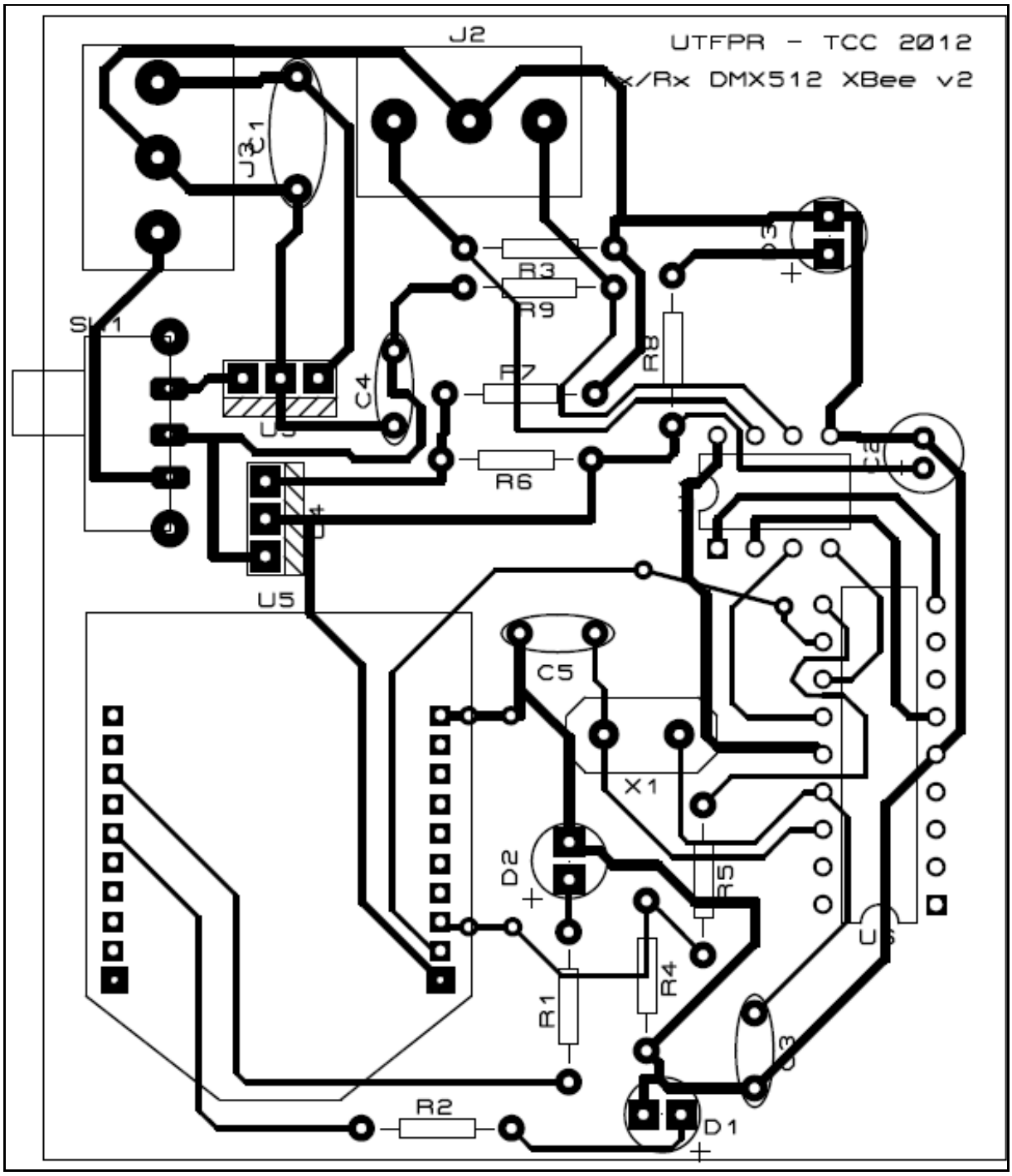


Figura 31 - Placa de circuito impresso
Fonte: Autoria própria

As figuras 32, 33 e 34 mostram como foi projetada a disposição dos componentes na placa de circuito impresso:

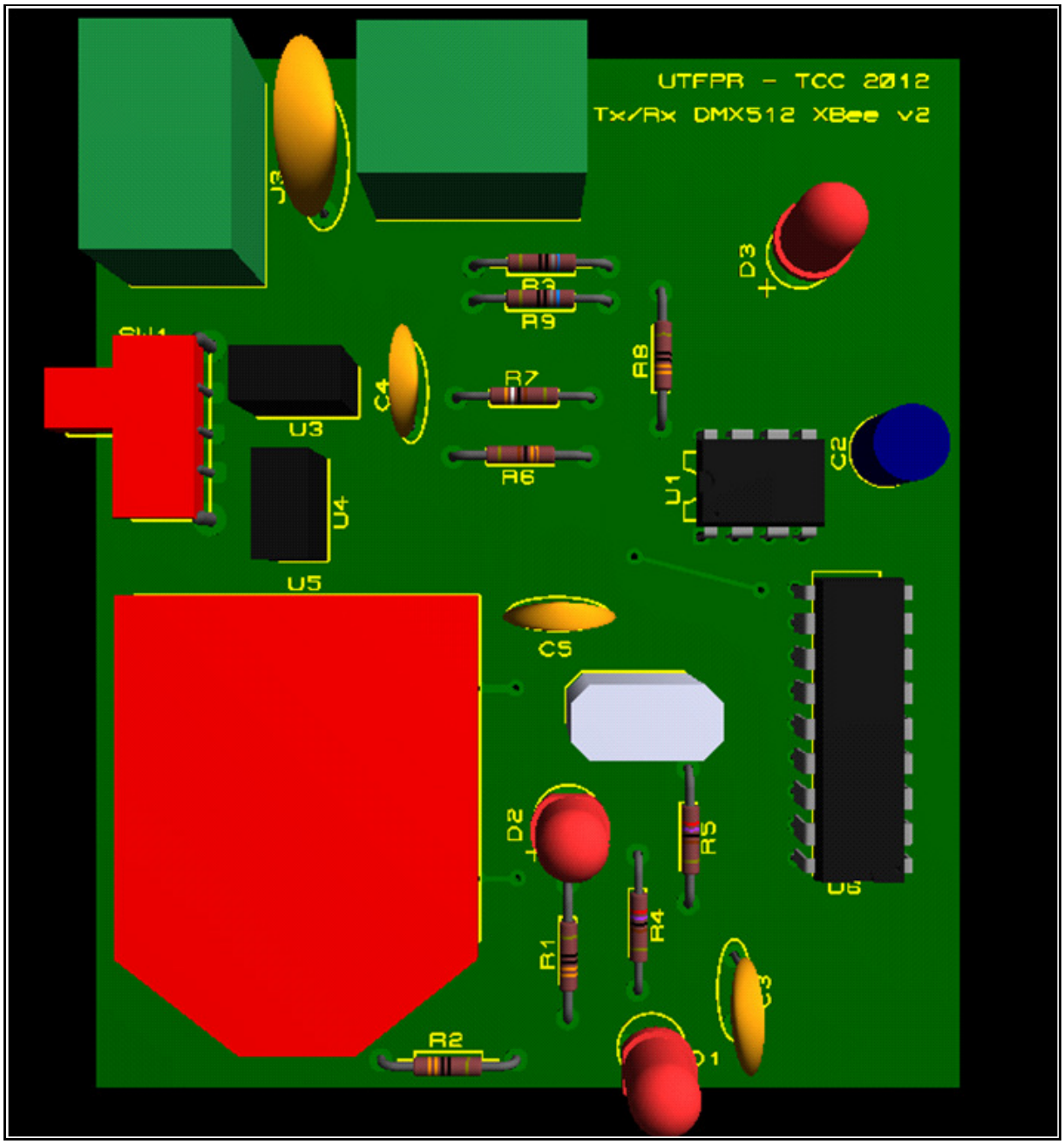


Figura 32 - Vista superior da placa
Fonte: Autoria própria

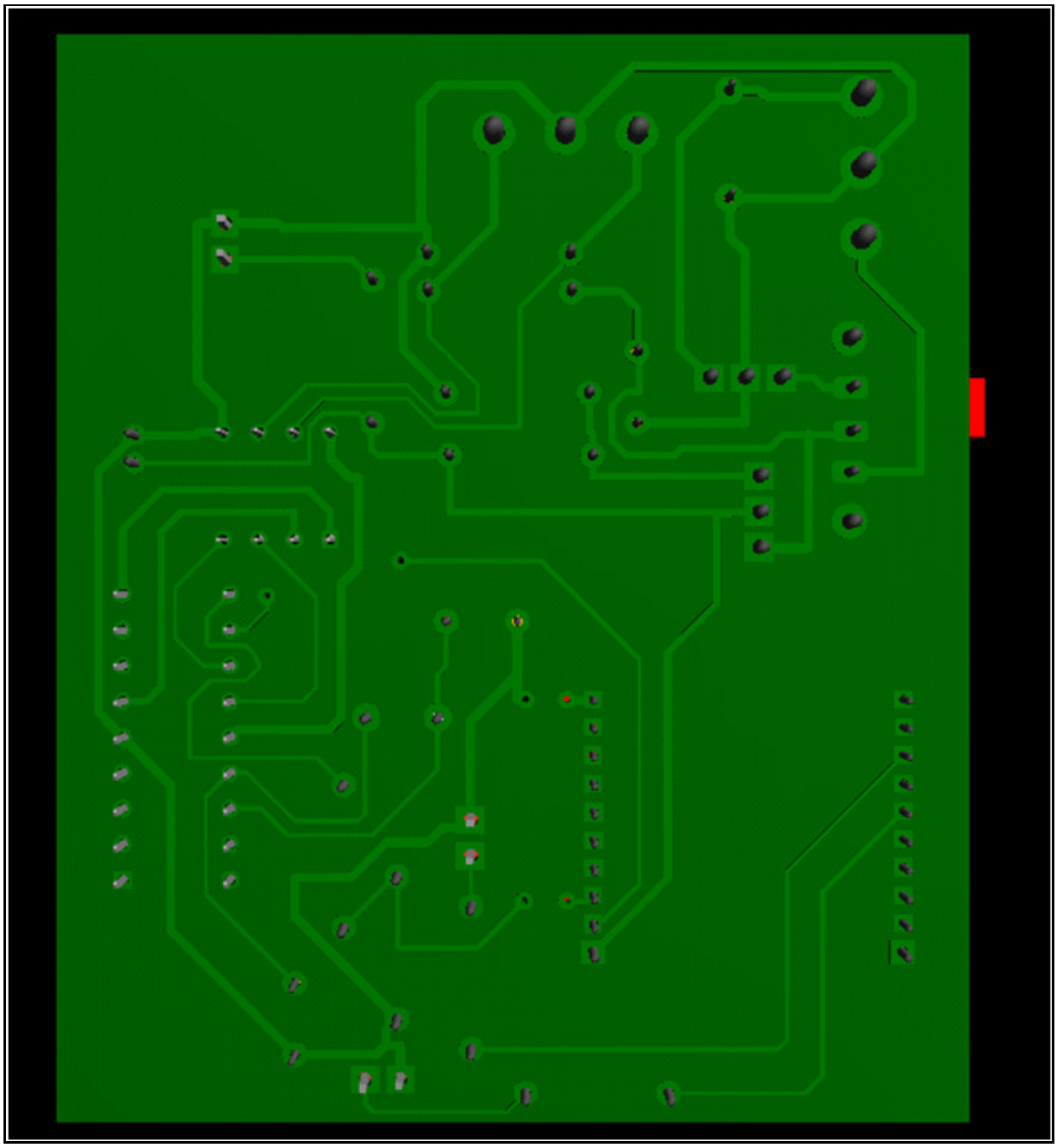


Figura 33 - Vista inferior da placa
Fonte: Autoria própria

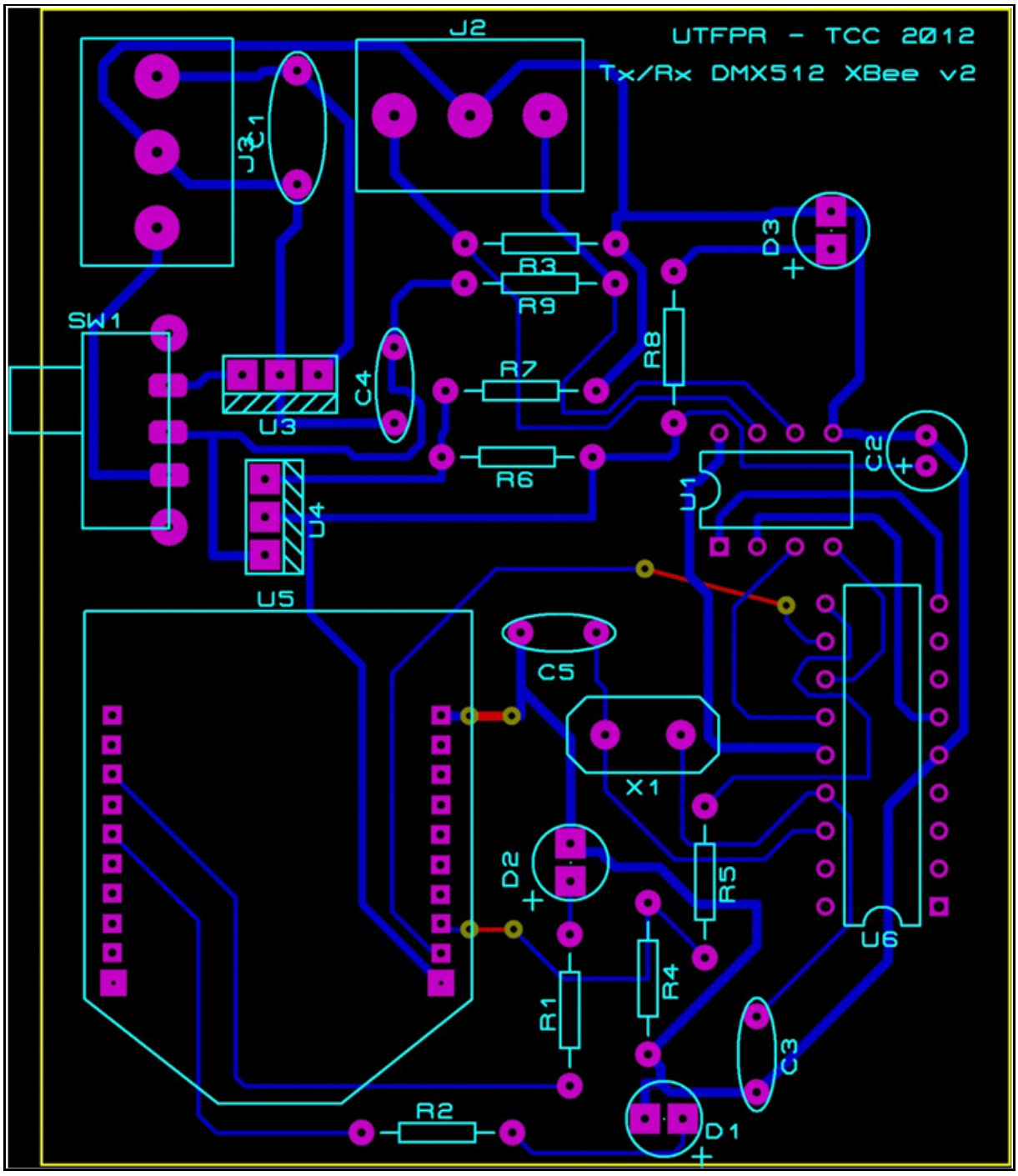


Figura 34 - Vista inferior – Simulação de raios-X
Fonte: Autoria própria

3.3 TESTES

3.3.1 Funcionamento

Os testes foram realizados em duas fases. Na primeira fase, as programações estavam individualizadas no transmissor e receptor, sendo testados individualmente. Na segunda fase a programação e o circuito estavam unificados e configurados para as funções de transmissor e receptor.

Na primeira fase todos os testes foram feitos em *proto-board*. Uma mesa controladora de 128 canais foi colocada na entrada do circuito e na saída do circuito foi colocado um dispositivo controlado. Os parâmetros (forma de onda, tempo de atraso, ruído) foram controlados com um osciloscópio em cada bloco do circuito, garantindo que cada bloco estava enviando corretamente a informação para o seguinte. Durante essa primeira fase, foram observadas algumas necessidades do protótipo, como o cristal externo de 20 MHz no micro-controlador. Esse cristal foi colocado devido aos atrasos gerados pelo código e para melhorar o sincronismo na coleta de dados. Uma vez estabelecido todos os ajustes e o com o protótipo funcionando, foi confeccionada a placa de circuito impresso.

Na segunda fase os testes foram feitos com o circuito montado nas placas de circuito impresso, acertando somente o sincronismo. O sincronismo foi um dos grandes desafios do protótipo. Os tempos entre um bit e outro foram calculados, porém para que não sejam gerados erros no sinal transmitido ou recebido deve-se considerar o tempo de execução da instrução pelo micro-controlador, principalmente se usado o oscilador interno que possui um *clock* menor, influenciando muito no sincronismo.

O tempo de resposta usando os dois protótipos é na ordem de alguns milissegundos, o que não afetou o funcionamento do dispositivo controlado.

3.3.2 Consumo

Foi usada uma fonte externa de 12 Volts que apresentou um consumo de aproximadamente 100 mA. O consumo do circuito ficou menor quando usada uma fonte de 5 V, consumindo uma corrente de 75 mA. O circuito tem um consumo menor em 5 V, pois não alimenta o circuito integrado LM-7805, aumentando assim sua eficiência. O circuito pode reduzir ainda mais seu consumo se for configurado o modo *sleep* dos módulos xbee e do PIC16F648A.

Foi usada uma alimentação de 4,5 V através de 3 pilhas de 1,5 V e o circuito funcionou adequadamente. Com pilhas recarregáveis de 1,25 V, a tensão após o regulador LM-317 ficou em 2,4V, abaixo dos 2,8 V exigidos pelo xbee. Nessa condição o módulo não operou adequadamente.

3.3.3 Alcance

Os módulos xbee possuem um pino que pode ser usado como LED de diagnóstico. Esse LED indica o tipo de associação que possuem. O quadro 7 mostra a relação entre a associação e a frequência de acendimento dos LEDs (XBEE..., 2007).

Frequência de acendimentos	Modo
Acesso	Não associado
4 Hz	Associado a rede Associado a rede como *router* ou *end device*
2 Hz	Associando a rede Associado a rede como um coordenador
10 Hz	*End device* em modo econônico

Quadro 7 - Identificação do modo de associação
Fonte: Xbee... (2007)

Com os LEDs de diagnóstico ligados, aumentou-se gradativamente à distância até que o módulo ficasse em modo não associado (LED acesso). (XBEE..., 2007).

Os módulos permaneceram associados até uma distância de aproximadamente 65 metros em vão livre e 50 metros com objetos para transpor.

4 CONCLUSÃO

A tecnologia de transmissão de dados sem fio é amplamente usada em diversos meios de comunicação. Com o avanço de tecnologias de modulação e demodulação, comunicações que eram dominadas por tecnologias a cabo estão migrando para tecnologias sem fio.

O projeto apresentado conseguiu eliminar o uso do cabo para transmissões DMX-512 entre dois pontos, de forma segura e com baixo consumo de energia, sendo dessa maneira concluído com êxito. Os protótipos xbee, que sofreram algumas alterações durante o projeto, atenderam a proposta inicial e as expectativas.

As pesquisas realizadas para o desenvolvimento do protótipo xbee fizeram com que fossem estudadas novas tecnologias, gerando novos conhecimentos. Esses conhecimentos, mesmo não tendo sido usados no protótipo xbee, podem ser utilizados em novos projetos ou melhorias desse protótipo.

O circuito transmissor e receptor desse protótipo, o xbee, pode ser trocado por outros módulos da mesma série, ou módulos de outras séries. Dessa forma, pode-se utilizar o protótipo para ter um alcance maior ou ser mais imune a ruídos. Pode-se ainda utilizar outras topologias que as redes xbee suportam para atender outras demandas de comunicação da tecnologia DMX-512.

O conceito do projeto inicial pode ser estendido para substituir outros sistemas com fios que usam os padrões RS-232 e o RS-485, como automação residencial, automação industrial, agronegócio, biomedicina, entre outros.

Analisando os custos do protótipo, ele é comercialmente viável, uma vez que o custo por interface é menor que interfaces similares encontradas no mercado. Considerando o mercado de telecomunicações, que é muito competitivo, o custo de um produto é um fator determinante na escolha de um produto.

O produto final consiste da soma dos esforços da equipe em conjunto com os conhecimentos adquiridos durante o Curso Superior de Tecnologia em Comunicações Digitais. O êxito na elaboração do protótipo não ocorreu de forma fácil, consumindo bastante tempo e esforços. A insistência e criatividade dos envolvidos foram essenciais para conseguir o resultado final que atendesse as exigências propostas.

5 REFERÊNCIAS

ANATEL - Agência Nacional de Telecomunicações. **Atribuição de faixas de frequências no Brasil.** Outubro de 2006. Disponível em: <http://www.anatel.gov.br/Portal/exibirPortalRedireciona.do?codigoDocumento=98580&caminhoRel=In%EDcio-Radiofreq%FC%EAncia-Apresenta%E7%E3o>. Acesso em: 10 jan. 2012.

ATMEL Corporation. ***AT86RF211S Datasheet.*** Março de 2005. Disponível em: <http://atmel.icfull.com/datasheet/AT86RF211S-PDF.html>. Acesso em: 28 jul. 2011.

BALDWIN, Tom. ***5 pin XLR.*** Disponível em: <http://business.virgin.net/tom.baldwin/pinout-5xlr.html>. Acesso em: 20 jul. 2011.

BOYLESTAD, Robert L.; NASHELSKY, Louis. **Dispositivos eletrônicos e teoria de circuitos.** 8ª Edição, São Paulo: Pearson Prentice Hall, 2006.

BRITO, Janaína; FONTES, Nena. **Estratégias para eventos: uma ótica do marketing e do turismo.** São Paulo: Aleph, 2002.

CARVER, Dave. ***Quadrature Amplitude Modulation.*** Disponível em : <http://www.physics.udel.edu/~watson/scen103/projects/96s/thosguys/qam.html>. Acesso em: 10 jun. 2012.

CASTRO, Maria Cristina Felippeetto de. **Fundamentos de comunicação de dados.** Porto Alegre. Disponível em: <www.feng.pucrs.br/~decastro/TPI/TPI_Cap3_parte2.pdf>. Acesso em: 10 fev. 2012.

ELETROHOO Projetos. **Amplificador AM/FM.** Disponível em: <http://eletrohoo.paros.uni5.net/projetos/projeto_exibir.asp?Amplificadores_1>. Acesso em: 04 abr. 2012.

ERNST, Matt. ***The DMX Portal.*** *Circuit Cellar Magazine*, Agosto de 2008. Disponível em: <http://www.circuitcellar.com/archives/viewable/217-Ernst/Ernst-217.pdf>. Acesso em: 09 jul. 2011.

EXAR Corporation. ***XR2206 Datasheet.*** Ano de 1972. Disponível em: <www.bucek.name/pdf/xr2206.pdf>. Acesso em: 29 jul. 2011.

EXAR Corporation. ***XR2211 Datasheet.*** Ano de 1995. Disponível em: <http://www.jaycar.com.au/images_uploaded/XR2211V3.PDF>.
Acesso em: 29 jul. 2011.

FAIRCHILD Semiconductor Corporation. ***LM-317 Datasheet.*** Outubro de 2011. Disponível em: <www.fairchildsemi.com/ds/LM/LM317.pdf>. Acesso em: 03 fev. 2012.

FAIRCHILD Semiconductor Corporation. ***LM-7805 Datasheet.*** Fevereiro de 2012. Disponível em: <www.fairchildsemi.com/ds/LM/LM7805.pdf>. Acesso em: 03 fev. 2012.

FARIAS, Irene Silva. **Modulação angular por sinais digitais.** Fevereiro de 2007. Disponível em:
<http://professores.unisanta.br/isfarias/Materia/Comunicacao%20Digital/fsk.pdf>.
Acesso em: 11 jul. 2011.

FILARDI, Vitor Leão. **RS-485 Especificação e Modo de Operação.** Salvador, 2007. Disponível em:
<http://www.e-science.unicamp.br/angra/admin/publicacoes/documentos/
publicacao_616_RS485.pdf>. Acesso em: 13 jul. 2011.

FUTURLEC *Electronic Components*. ***XLR-5 Pin XLR.*** Disponível em:
<http://www.futurlec.com/XLR-5Pin.shtml>. Acesso em: 20 jul. 2011.

GIDO, Jack; CLEMENTS, James P. **Gestão de Projetos.** São Paulo: Thomson Learning, 2007.

GOMES, Romeu. **Iluminação computadorizada II – DMX 512 como funciona.** Disponível em:
<http://www.equipashow.com.br/index.php?link1=not&pgid=lermtc&mtc_id=8&titulo=Ilumina%E7%E3o+Computadorizada+II+-+DMX-512+como+funciona> Acesso em: 23 jul. 2011.

HOCHMAN, Kátia. **Mercado de eventos movimenta R$ 37 Bi.** Diário do Comércio & Indústria, 24/07/2002. Disponível em:
<http://www.sebrae-sc.com.br/ideais/printer.asp?cd_noticia=2341>. Acesso em: 23 jul. 2011.

INFINEON Technologies. ***TDA-7100 Datasheet.*** Maio de 2007. Disponível em:
<www.ti.com/lit/ds/symlink/cc1100.pdf>. Acesso em: 28 jul. 2011.

INTEL Corporation. ***MCS-8051 Datasheet.*** Outubro de 1988. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/107780/INTEL/8051.html>. Acesso em: 29 jul. 2011.

KINNEY, Patrick. ***ZigBee Technology: Wireless Control that Simply Works.*** Disponível em: <http://hometoys.com/emagazine.php?url=/htinews/oct03/articles/kinney/zigbee.htm>. Acesso em: 10 mar. 2012.

KRON Medidores. **Conceitos Básicos de RS485 e RS422.** Disponível em: <http://www.kronweb.com.br/download2.php?id=353>. Acesso em: 10 jul. 2011.

LACERDA, Ana Paula. **Mercado de eventos é dominado por empresas de pequeno porte.** O Estado de São Paulo, página 35, 19/04/2005. Disponível em: <http://acervo.estadao.com.br/pagina/#!/20050419-40726-spo-35-eco-b20-not>. Acesso em: 23 jul. 2011.

LANGTON, Charan. ***Frequency Modulation (FM), FSK, MSK and more.*** Fevereiro de 2002. Disponível em: <http://www.complextoreal.com/chapters/fm.pdf>. Acesso em: 18 jul. 2011.

LANGTON, Charan. ***All About Modulation – Part 1.*** Dezembro de 2005. Disponível em: <http://www.complextoreal.com/chapters/mod1.pdf>. Acesso em: 18 jul. 2011.

LINEAR Equipamentos Eletrônicos. **Processos de modulação – Técnicas de modulação digital.** Ano de 2004. Disponível em: <http://www.scribd.com/doc/58706867/23/MODULACAO-POR-DESVIO-DE-FREQUÊNCIA-FSK>. Acesso em: 08 jul. 2011.

MAXIM Integrated Products. ***Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers.*** Ano de 2003, Sunnyvale, California. Disponível em: <http://www.datasheetcatalog.org/datasheet/maxim/MAX1487-MAX491.pdf>. Acesso em: 21 abr. 2012.

MAXIM Integrated Products. ***DS-1216 Datasheet.*** Sunnyvale, California. Disponível em: <http://pdfserv.maxim-ic.com/en/ds/DS1216-DS1216H.pdf>. Acesso em: 29 jul. 2011.

MESSIAS, Antônio Rogério. **Controle remoto e aquisição de dados via XBee/ZigBee (IEEE 802.15.4).** Disponível em: <http://www.rogercom.com/ZigBee/ZigBee.htm>. Acesso em: 04 fev. 2012.

MICROCHIP *Technology Inc.*. ***PIC16F627A/628A/648A Data Sheet.*** Agosto de 2006. Disponível em:
<http://ww1.microchip.com/downloads/en/devicedoc/40044f.pdf>.
Acesso em: 25 abr. 2012.

NATIONAL Semiconductor Corporation. ***LM-741 Operational Amplifier.*** Novembro de 1994. Disponível em:
<http://www.physics.rutgers.edu/rcem/~ahogan/327/LM741.pdf>. Acesso em: 03 fev. 2012.

NATIONAL Semiconductor Corporation. ***LM158 / LM258 / LM358 / LM2904 Low Power Dual Operational Amplifiers.*** Outubro de 2005 . Disponível em:
<www.national.com/ds/LM/LM158.pdf >. Acesso em: 03 fev. 2012.

ON Semiconductor. ***LM392 Low Power Operational Amplifiers and Comparators.*** Setembro de 2009. Disponível em:
< www.onsemi.com/pub/Collateral/LM392-D.PDF >. Acesso em: 15 mar. 2012.

PERRIN, Bob. ***The art and Science of RS-485.*** *Circuit Cellar Magazine*, Julho de 1999. Disponível em:
<http://www.circuitcellar.com/library/ccofeature/perrin0799/c79bppdf.pdf>. Acesso em: 11 jul. 2011.

PLASA. ***Alternate Start Codes.*** Disponível em:
<http://tsp.plasa.org/tsp/working_groups/CP/DMXAlternateCodes.php>. Acesso em: 17 jul. 2011.

RIBEIRO, Daniel; TORRES, Pedro; PEREIRA, Paula. ***DMX Light Control.*** Fevereiro de 2006. Disponível em:
<http://www.est.ipcb.pt/pessoais/pedrotorres/Documentos/Artigo_DMXLightControl.pdf>. Acesso em: 23 jul. 2011.

SALEIRO, Mario; EY, Emanuel. **ZIG-BEE, uma abordagem prática.** Fevereiro de 2008, Algarve. Disponível em:
<http://lusorobotica.com/ficheiros/Introducao_ao_Zigbee_-_por_msaleiro.pdf>.
Acesso em: 23 jan. 2012.

SPARKFUN Eletronic. ***XBee 1mW Wire Antenna - Series 1 (802.15.4).*** Disponível em: <http://www.sparkfun.com/products/8665>. Acesso em: 02 fev. 2012.

TANENBAUM, Andrew S.. **Redes de computadores.** 3ª ed. Rio de Janeiro: Campus, 1997.

TANENBAUM, Andrew S.. **Redes de computadores.** 4ª ed. Rio de Janeiro: Campus, 2003.

TEXAS Instruments. ***HEX INVERTERS.*** Dezembro de 1983, Dalas, Texas. Disponível em: <www.ti.com/lit/ds/symlink/sn7404.pdf>. Acesso em: 27 jul. 2011.

TEXAS Instruments. ***Dual EIA-232 Drivers/Receivers.*** Fevereiro de 1989, Dallas, Texas. Disponível em: <www.ti.com/lit/ds/symlink/max232.pdf>. Acesso em: 21 abr. 2012.

TEXAS Instruments. ***HEX INVERTER BUFFER/DRIVERS WITH OPEN-COLLECTOR HIGH VOLTAGE OUTPUTS.*** Ano de 2001, Dallas, Texas. Disponível em: < www.ti.com/lit/ds/symlink/sn7416.pdf>. Acesso em: 28 jul. 2011.

TEXAS Instruments. ***CC1101E Datasheet.*** Ano de 2011, Dallas, Texas. Disponível em: <www.ti.com/lit/ds/symlink/cc1100.pdf>. Acesso em: 28 jul. 2011.

UNITED STATES INSTITUTE FOR THEATRE TECHNOLOGY. ***About Us.*** Disponível em: <http://www.usitt.org/AboutUs.aspx>. Acesso em: 15 jul. 2011.

UNITED STATES INSTITUTE FOR THEATRE TECHNOLOGY. ***DMX512 FAQ.*** Disponível em: <http://www.usitt.org/DMX512FAQ.aspx>. Acesso em: 02 jan. 2012.

UNIVERSITY OF CALIFORNIA – IRVINE (UCI). ***Understanding the Transmission Line Theory.*** Versão 2001.2, Junho de 2001. Disponível em: <http://www.ece.uci.edu/docs/hspice/hspice_2001_2-269.html>. Acesso em: 10 jul. 2011.

XBEE™/XBEE-PRO™ OEM RF Modules. Ano de 2007. Disponível em: <http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Manual.pdf>. Acesso em: 02 fev. 2012.

XBEE®/XBee-PRO® RF Modules. Ano de 2009. Disponível em: <http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf>. Acesso em: 02 fev. 2012.

APÊNDICE A - CRONOGRAMA PREVISTO

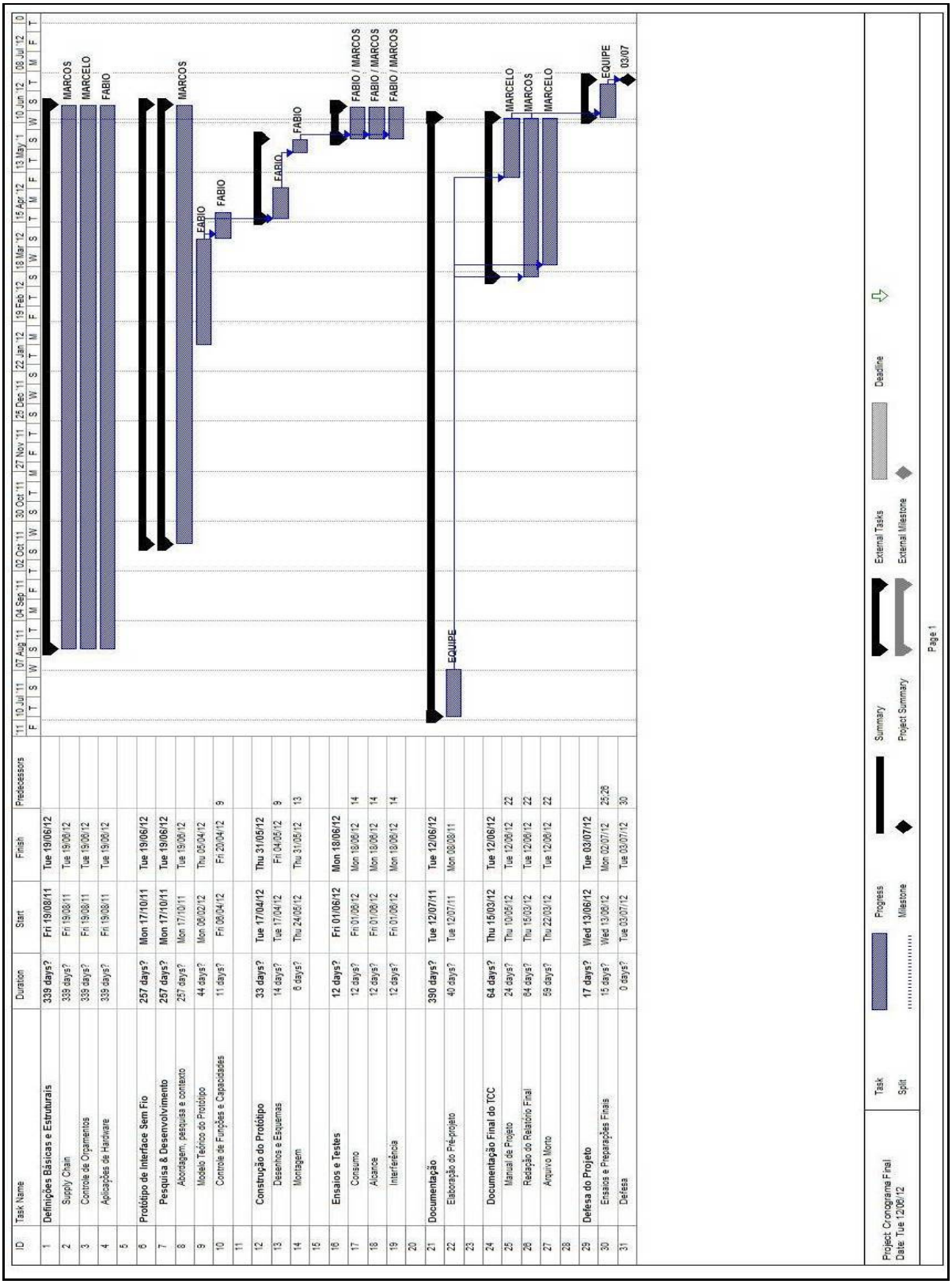


Figura 35 - Gráfico de Gantt do projeto.
Fonte: Autoria própria

APÊNDICE B - WBS DO PROJETO

APÊNDICE B.1 - WBS – Macro tarefas

A figura 36 ilustra as macro-tarefas do WBS:

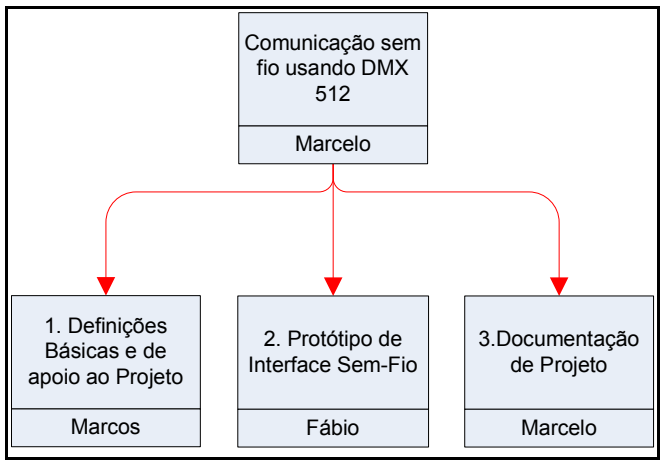


Figura 36 - Macro tarefas do WBS
Fonte: Autoria própria

APÊNDICE B.2 - WBS detalhado

Da figura 37 até a 44 é apresentado o detalhamento das tarefas do WBS:

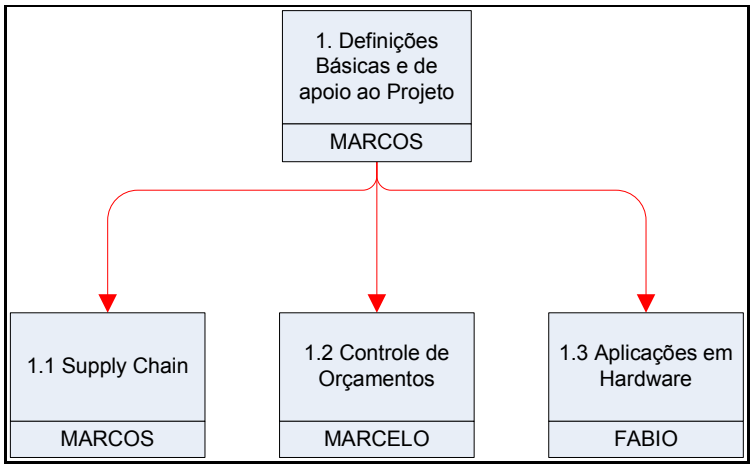


Figura 37 - Definições básicas e de apoio ao projeto
Fonte: Autoria própria

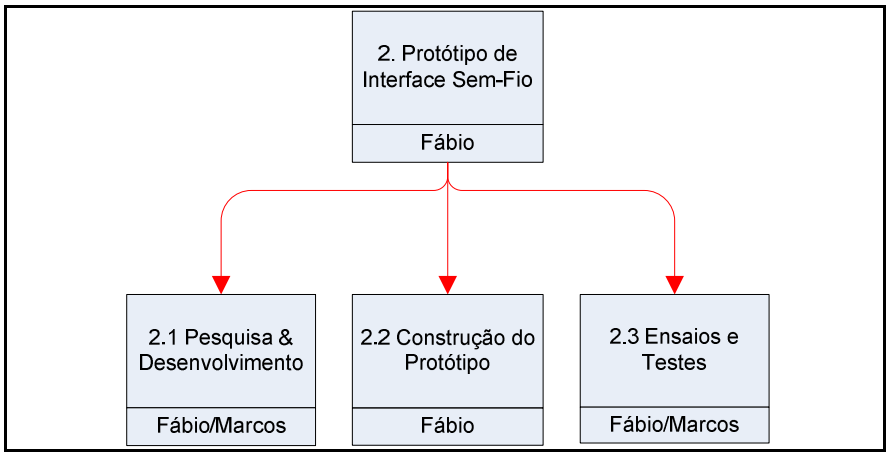


Figura 38 - Protótipo de interface sem fio
Fonte: Autoria própria

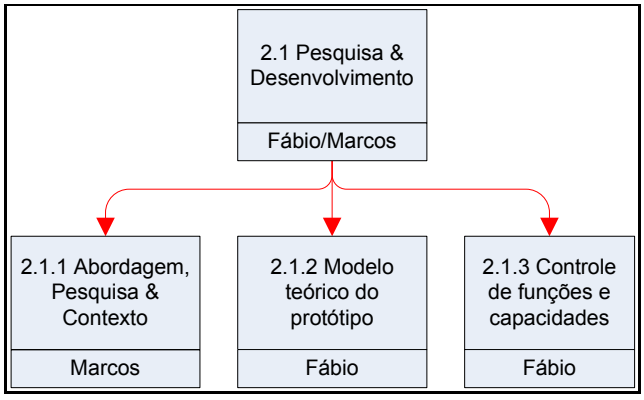


Figura 39 - Detalhamento da "Pesquisa e Desenvolvimento"
Fonte: Autoria própria

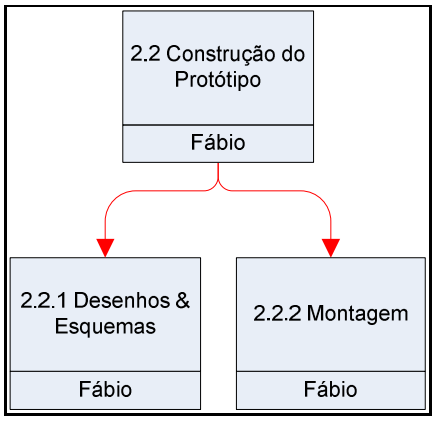


Figura 40 - Detalhamento da "Construção do protótipo"
Fonte: Autoria própria

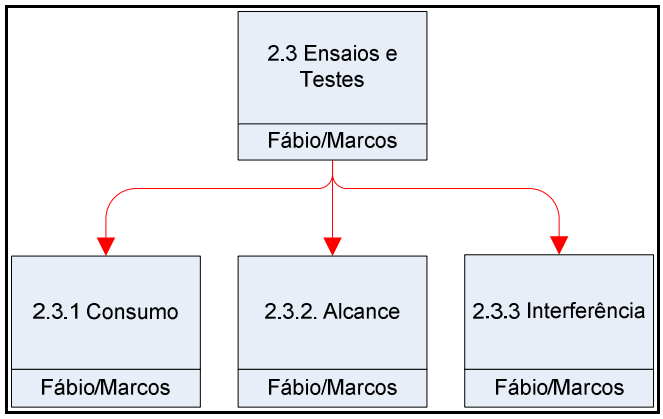


Figura 41 - Detalhamento dos "Ensaios e testes"
Fonte: Autoria própria

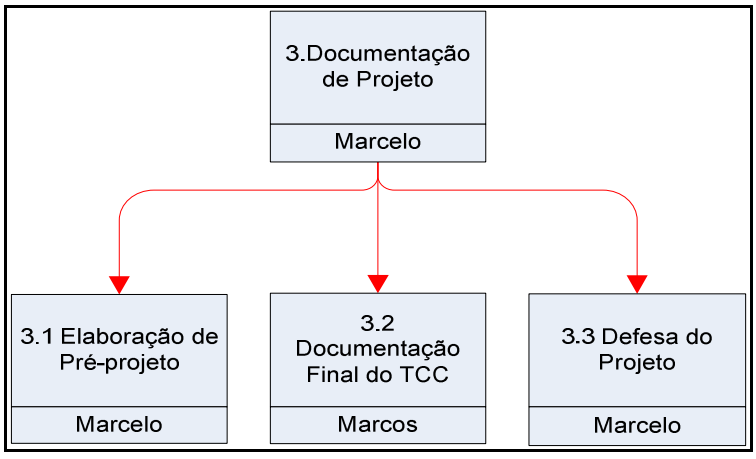


Figura 42 - Detalhamento da "Documentação do projeto"
Fonte: Autoria própria

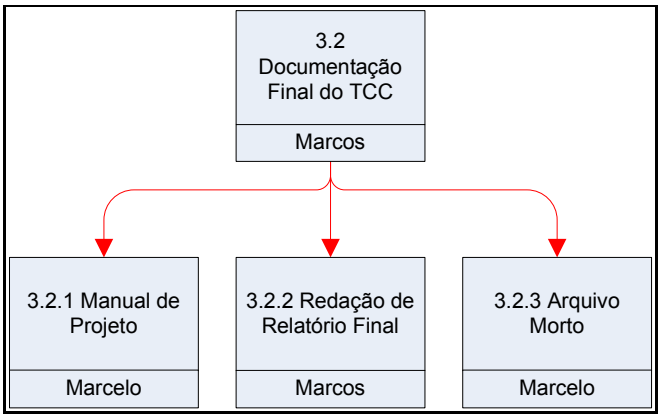


Figura 43 - Detalhamento da "Documentação Final do TCC"
Fonte: Autoria própria

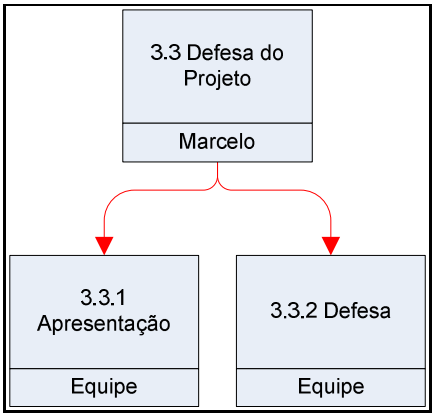


Figura 44 - Detalhamento da "Defesa do projeto"
Fonte: Autoria própria

APÊNDICE C - PROTÓTIPOS

APÊNDICE C.1 - Escolha dos protótipos

Com base nas comparações e premissas descritas no item 3.1, foram pesquisados circuitos e tecnologias que atendessem as necessidades. Foram encontrados inicialmente cinco métodos de executar o protótipo:

- Método 1 - Usar um circuito integrado que executa a maioria das funções necessárias, usando poucos componentes externos, que consiga transmitir dados a 250kbps;
  - Circuitos baseados no CC-1100E (TEXAS, 2011);
- Método 2 – Usar um circuito para conversão dos sinais DMX 512 em FSK e um circuito transmissor que consiga transmitir o sinal FSK a 250kbps;
- Método 3 – Usar um *buffer*, e um circuito integrado que executa a maioria das funções necessárias, usando poucos componentes externos, que consiga transmitir dados a uma taxa entre 10k e 100kbps;
  - Circuitos baseados na memória DS-1216 para *buffer* do sinal e no AT86RF211S (100 kbps) ou TDA-7100 (20 kbps) para transmissão e recepção (ATMEL, 2005; INFINEON, 2007; INTEL, 1988; MAXIM, 2011);
- Método 4 – Usar um *buffer*, um circuito micro-controlado para conversão dos sinais e um circuito transmissor ASK/FSK que consiga transmitir a entre 10k e 50kbps;
  - Circuitos baseados na memória DS-1216 para *buffer* do sinal, no micro-controlador MCS-8051 para conversão do sinal e no AT86RF211S (100 kbps) ou TDA-7100 (20 kbps) para transmissão e recepção (ATMEL, 2005; INFINEON, 2007; INTEL, 1988; MAXIM, 2011);
- Método 5 – Usar um *buffer*, um circuito micro-controlado para conversão e compactação do sinal e um circuito transmissor ASK/FSK que consiga transmitir a entre 10k e 50kbps;

- Circuitos baseados na memória DS-1216 para *buffer* do sinal, no micro-controlador PIC16F648A para tratamento e compactação do sinal e TDA-7100 (20 kbps) para transmissão e recepção (INFINEON, 2007; MAXIM, 2011; MICROCHIP, 2006);

O quadro 8 apresenta uma comparação entre os cinco métodos. Essa comparação foi feita tendo como base folhas de dados (*datasheet*) de circuitos integrados que atendem os métodos (ATMEL, 2005; EXAR, 1972; EXAR, 1995; INFINEON, 2007; INTEL, 1988; MAXIM, 2011; MICROCHIP, 2006; TEXAS, 2011). Os itens do quadro foram comparados e foi atribuído um número inteiro de 1 a 5 para avaliação, sendo que quanto maior o número, melhor a característica, em relação ao outro método. A soma das notas atribuídas foi o critério utilizado para a escolha do método.

	Método 1	**Método 2**	**Método 3**	**Método 4**	**Método 5**
Alcance do sinal	5	4	4	4	4
Facilidade de construção	2	3	3	2	1
Custo	4	5	5	4	4
Tamanho do protótipo	4	4	3	3	2
Taxa de dados	5	4	3	2	2
Soma	20	20	18	15	13

Quadro 8 - Comparação entre os métodos
Fonte: Autoria própria

Como o método um e dois tiveram notas iguais, foi elaborado o quadro 9, que compara as principais características dos circuitos utilizados nos dois métodos (EXAR, 1972; EXAR, 1995; TEXAS, 2011).

	CC1100E Protótipo Alfa	XR-2206 / XR-2211 Protótipo Beta
Alcance do sinal	~100m, depende da configuração do circuito	~100m, depende do amplificador do transmissor
Facilidade de construção	SMD - Necessário componentes periféricos e placa de circuito impresso.	Eletrônica discreta - Componentes periféricos e placa de circuito impresso
Consumo estimado	50mW	100mW
Custo estimado	R$ 100,00 cada transmissor / receptor	R$ 50,00 cada transmissor / receptor
Tamanho do circuito	~5cm² (2,25x2,25)	~5cm² (2,25x2,25)
Taxa de dados	até 500 kbps	até 250 kbps

Quadro 9 - Características resumidas do circuito do método 1 e 2
Fonte: Autoria própria

Os dois métodos apresentam características bastante similares, cada um com vantagens em critérios específicos. Foram executados protótipos com ambos os métodos para comparar qual dos dois seria o mais adequado para atender a necessidade do projeto. O protótipo que usa o CC1100E foi denominado protótipo alfa e o protótipo que usa os integrados XR-2206 e XR-2211 foi denominado protótipo beta para simplificar futuras referências. O protótipo alfa derivou o protótipo xbee, que funcionou adequadamente seguindo os critérios estabelecidos. Os outros dois protótipos não atenderam as necessidades do projeto e estão nesse apêndice para preservar as pesquisas realizadas.

APÊNDICE C.2 - Protótipo alfa

A ideia desse protótipo era usar o transceiver CC1100E como circuito principal, utilizando componentes periféricos para ajuste de suas funções, de acordo com a necessidade (transmissor / receptor). A figura 45 apresenta um diagrama em blocos do circuito:

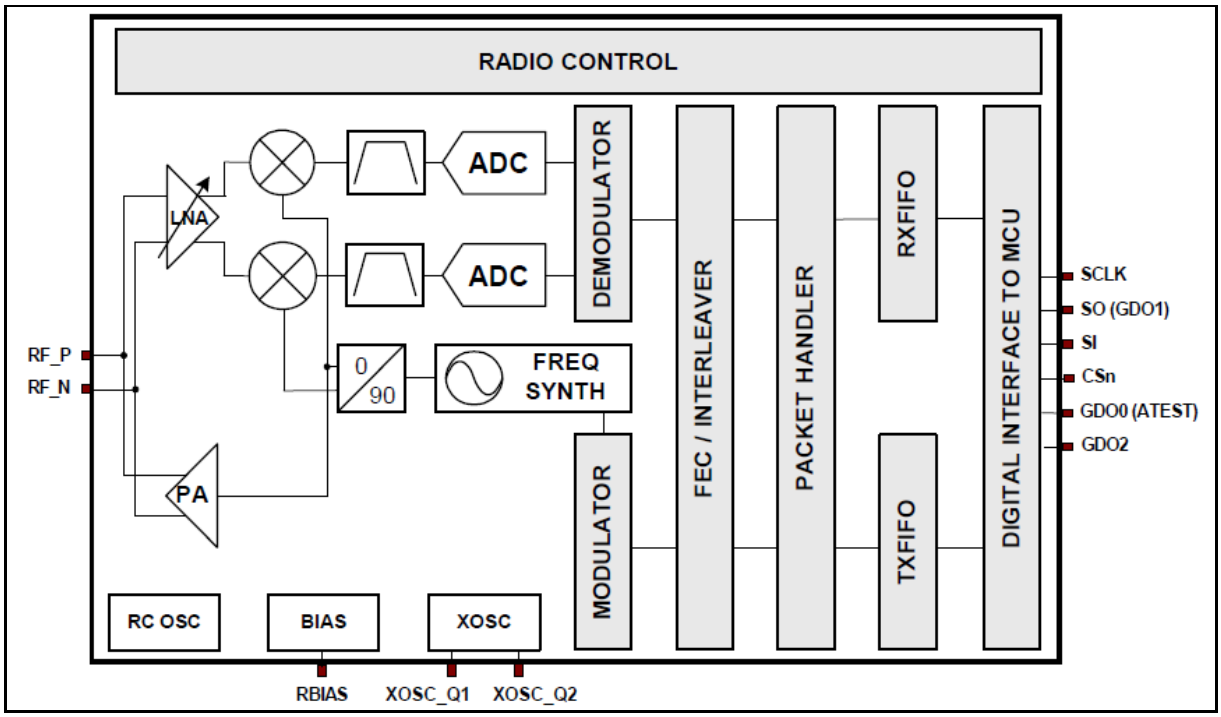


Figura 45 - Diagrama em blocos do CC1100E
Fonte: Texas (2011)

A figura 46 é o diagrama elétrico recomendado do circuito retirado de sua folha de dados, sendo o quadro 10 a relação de componentes recomendados para uso na frequência de 950 MHz:

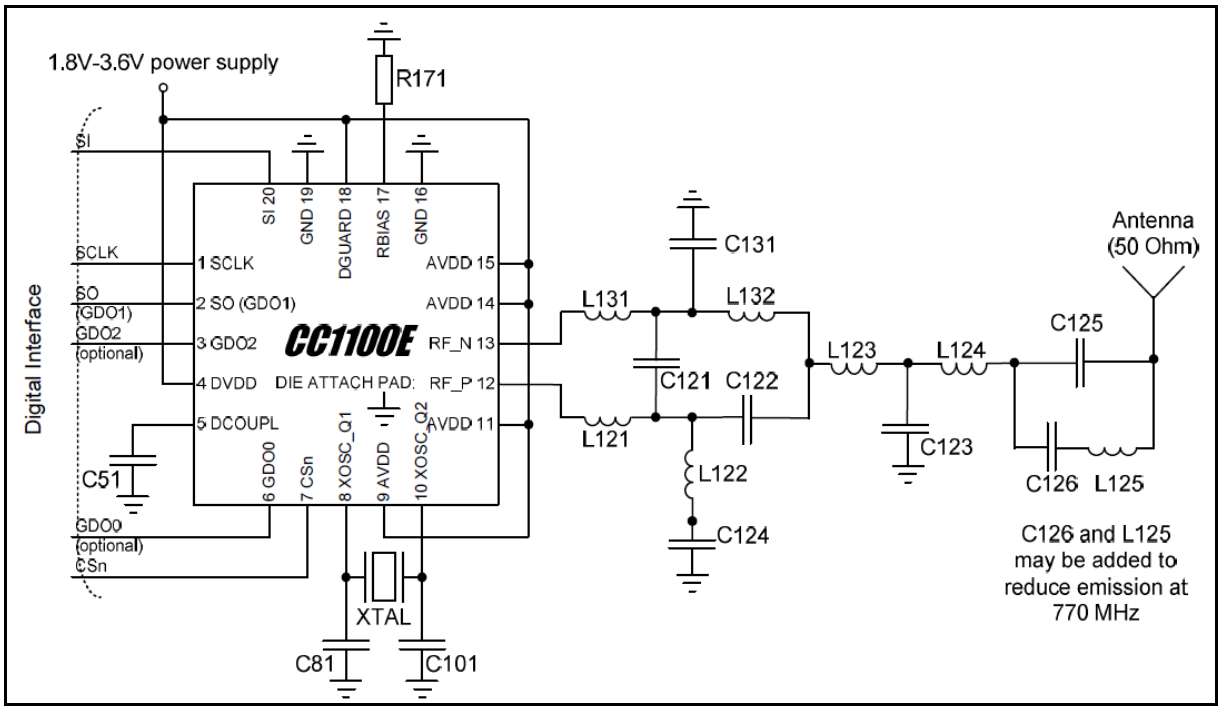


Figura 46 - Circuito típico para aplicação e avaliação (950 MHz)
Fonte: Texas (2011)

Component	Value at 470MHz	Value at 950MHz
C51	100 nF ± 10%, 0402 X5R	
C81	27 pF ± 5%, 0402 NP0	
C101	27 pF ± 5%, 0402 NP0	
C121	3.9 pF ± 0.25 pF, 0402 NP0	1.0 pF ± 0.25 pF, 0402 NP0
C122	6.8 pF ± 5% pF, 0402 NP0	1.5 pF ± 0.25 pF, 0402 NP0
C123	5.6 pF ± 0.5 pF, 0402 NP0	2.7 pF ± 0.25 pF, 0402 NP0
C124	.220 pF pF ± 5%, 0402 NP0	100 pF ± 5%, 0402 NP0
C125	220 pF ± 5%, 0402 NP0	100 pF ± 5%, 0402 NP0 or 11 pF ± 5%, 0402 NP0 when part of optional filter
C126		47 pF ± 5%, 0402 NP0
C131	3.9 pF ± 0.25 pF, 0402 NP0	1.5 pF ± 0.25 pF, 0402 NP0
L121	27 nH ± 5%, 0402 wire wound	12 nH ± 5%, 0402 wire wound
L122	22 nH ± 5%, 0402 wire wound	18 nH ± 5%, 0402 wire wound
L123	27 nH ± 5%, 0402 wire wound	12 nH ± 5%, 0402 wire wound
L124		12 nH ± 5%, 0402 wire wound
L125		2.7 nH ± 0.2nH, 0402 wire wound
L131	27 nH ± 5%, 0402 wire wound	12 nH ± 5%, 0402 wire wound
L132		18 nH ± 5%, 0402 wire wound
R171	56k Ω, 0402, 1%	
XTAL	26.0 MHz surface mount crystal	

Quadro 10 - Lista de componentes – 950 MHz
Fonte: Texas (2011)

A tecnologia do integrado faz necessário o uso de placa de muitas camadas e componentes que não são facilmente encontrados no mercado. Seu tamanho reduzido (área de aproximadamente 0,25 cm^2) inviabiliza a produção de um protótipo em baixa escala, como pode ser observado na fotografia 10, onde é comparado o circuito integrado a um objeto. Os custos para a programação do circuito integrado CC-1100E seriam altos, uma vez que seria necessário adquirir uma interface para sua configuração, placas de múltiplas camadas, etc.

Fotografia 10 - Comparativo – CC1100E *vs* tampa de caneta
Fonte: Autoria própria

Foram pesquisados circuitos integrados semelhantes com outra forma de encapsulamento, e nesse momento foram encontradas as tecnologias *ZigBee e xbee*. Essas tecnologias, em determinados casos, utilizam o circuito integrado CC-1100E para transmissão e recepção de dados. Foi realizada uma análise de custos. Como os custos ficaram similares, foi alterado o protótipo para usar interfaces *xbee series 1*, que atendem as demandas do projeto de forma análoga ao circuito que seria projetado.

O protótipo usando xbee foi denominado protótipo xbee e suas informações encontram-se no item 3.2 e Apêndice C.3.

APÊNDICE C.3 - Protótipo xbee – pesquisas

O uso do protótipo xbee resultou em diversas pesquisas e testes. Os circuitos que funcionam e foram testados estão descritos no item 3.2. Demais circuitos pesquisados foram colocados nesse apêndice.

Circuito transmissor

O circuito transmissor foi dividido em dois circuitos. O primeiro circuito deve converter os sinais elétricos da entrada (modo diferencial) para sinais TTL. O segundo circuito deve converter os sinais TTL em sinais seriais que então seriam colocados na porta de entrada do módulo xbee.

Foi feita uma análise dos sinais e o primeiro circuito elaborado para converter o modo diferencial em TTL consistia em um amplificador operacional na configuração de comparador. O sinal RS-485 é transmitido no modo diferencial, com tensão de aproximadamente 5 V. Para adaptar esse sinal usou-se um amplificador operacional, alimentado com a mesma tensão de alimentação do modulo xbee. Dessa forma, quando a tensão for maior na entrada não inversora, o sinal de saída será aproximadamente o sinal da alimentação +VCC (5,0 V) do amplificador operacional. Quando o sinal na entrada inversora for maior que na entrada inversora, o sinal de saída será o sinal de alimentação –VCC (0 V) do amplificador operacional (BOYLESTAD, NASHELSKY, 2006; FILARDI, 2011).

A figura 47 apresenta o circuito que foi descrito anteriormente:

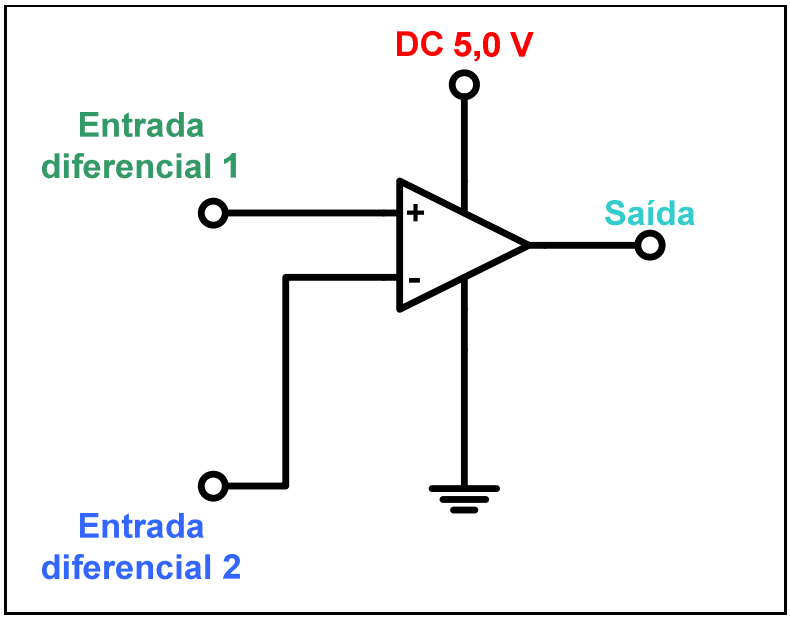


Figura 47 - Circuito de conversão para o transmissor
Fonte: Adaptado de Boylestad, Nashelsky (2006)

Inicialmente foram testados os amplificadores operacionais LM-741 e LM-358, porém eles não possuem um tempo de resposta adequado para a frequência de 250 kHz. Como o tempo de resposta não era adequado, os CI's não identificavam corretamente a mudança de estado de um nível 0 para um nível 1, mantendo constante o nível da saída dos amplificadores operacionais (NATIONAL, 1994; NATIONAL, 2005).

Foram procurados amplificadores operacionais de resposta rápida e foi escolhido o LM-392, que é um comparador de resposta rápida e baixo consumo (ON, 2009). O circuito de entrada foi montado usando o LM-392 e funcionou corretamente em determinadas situações, porém apresentou diversos ruídos quando a diferença entre os sinais estava baixa, devido à sensibilidade do LM-392. Em alguns casos o sinal de saída apresentava uma onda triangular, que não era adequada para a entrada do segundo circuito, pois alterava o período e a forma de onda, fazendo o segundo circuito não entender o *start bit* e o *stop bit* ou os próprios dados transmitidos. O circuito foi descartado por não atender nossas necessidades sem que fosse necessário algum controle adicional.

Foram pesquisados outros circuitos para a conversão de sinais para o xbee. Os mais comuns consistem em conversores que transformavam os níveis de tensão do RS-232 para níveis de tensão TTL e depois divisores de tensão para adaptar os níveis de tensão TTL para os níveis de tensão exigidos pelos módulos xbee no transmissor. Para o circuito receptor os conversores encontrados transformavam os níveis de tensão TTL para RS-232. Esses sinais TTL são adaptados à tensão dos módulos xbee e são colocados ou retirados nas portas de entrada e saída serial. Os módulos foram configurados em modo AT funcionando como uma interface serial sem fio. A ideia inicial era converter o sinal de RS-485 para RS-232 e depois de RS-232 para TTL no transmissor. No receptor, a ideia era converter o sinal de saída para RS-232 e depois fazer a conversão para RS-485. (FILARDI, 2011; XBEE..., 2009). A figura 48 ilustra um diagrama em blocos para conversão do sinal RS-485 para um sinal compatível com o sinal do xbee:

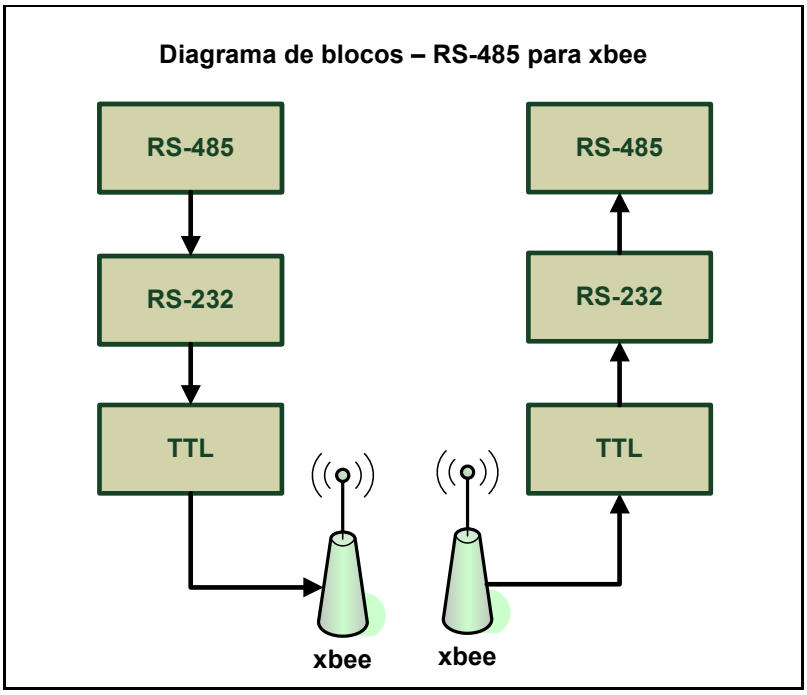


Figura 48 - Diagrama de blocos – RS-485 para xbee
Fonte: Autoria própria

Ao analisar os circuitos de conversão de RS-485 para RS-232, percebeu-se que os circuitos convertiam os sinais para níveis TTL. Dessa forma, é necessário a conversão de RS-485 para RS-232 e sim converter os níveis de tensão RS-485 para níveis TTL (MAXIM, 2003; TEXAS, 1989).

Também foram realizadas tentativas de utilizar um dos sinais diferenciais passando por um *buffer* inversor, o SN-7404 e resistores como divisor de tensão para adequar o sinal para as entradas do xbee. Esse circuito não funcionou corretamente, pois em determinadas situações o sinal ficava atenuado por características da linha e não era corretamente interpretado pelo inversor e consequentemente não era entendido pelo segundo circuito, pois não eram identificados os bits de início e parada.

Ao analisar o circuito de conversão de RS-485 para RS-232, identificou-se que o circuito integrado MAX-485 era usado na entrada do sinal RS-485 e gerava níveis de sinal TTL que então eram convertidos em sinais RS-232. O circuito foi testado e, mesmo com sinais atenuados ele apresenta boa resposta, gerando níveis de tensão TTL. A saída desse circuito foi ajustada para os níveis de tensão do módulo xbee e colocada na porta de recepção desse mesmo módulo.

Alguns dados foram transmitidos, porém a perda de sincronismo e a perda de sinal eram constantes. Analisando-se o quadro DMX-512, observou-se que o *start*

frame poderia gerar um erro no xbee e esse erro poderia estar influenciando na transmissão dos dados. Dessa forma surgiu o segundo circuito, que iria converter os sinais TTL em sinais seriais. Para essa função optou-se por usar um micro-controlador, devido ao tempo de resposta ser baixo e os micro-controladores necessitarem de poucos componentes adicionais para seu funcionamento.

Foram pesquisados dois circuitos micro-controladores. O primeiro era o micro-controlador 8051 (INTEL, 1988) e o segundo o PIC16F628A (MICROCHIP, 2006). Por ter uma tecnologia melhor, possuir melhores bibliotecas e a equipe já possuir as ferramentas de desenvolvimento (hardware e software), foi escolhido o PIC16F648A.

Ao analisar o *datasheet* do PIC16F628A, optou-se pelo PIC16F648A por possuir maior capacidade de ROM e RAM com um custo pouco maior (cerca de R$ 1,00).

O segundo circuito inicialmente tinha como função detectar o *break* e remover esse sinal. Esse sinal deve ser removido, pois não é um sinal serial. O *break* possui uma sequência de “zeros” que pode variar de 88 µs até 1 s. Esse sinal de *break* determina o início do *frame* DMX-512 e se colocado diretamente na porta do xbee, gera erro, pois o dado serial possui um bit de início (*start bit*) que fica em “zero”, a sequência de 8 bits de dados e pelo menos um bit de parada (*stop bit*) que fica em “um”. Com a sequência de *break* o xbee fica aguardando o bit de parada, que nunca chega, gerando erro no xbee e fazendo com que ele não transmita nada até o *start code*.

Após o teste realizado com o circuito removendo o *break*, percebeu-se que o circuito continuava com os problemas de sincronismo e perda de sinal. O circuito micro-controlador foi configurado inicialmente para usar o oscilador interno de 4 MHz. Ao medir o sinal de saída do oscilador, percebeu-se que ele não estava estável e optou-se por colocar um cristal externo de 20 MHz para o micro-controlador pudesse ser utilizado com sua velocidade máxima.

Os problemas com sincronismo e perda de sinal foram reduzidos, porém continuaram mesmo com o cristal externo. O PIC16F648A foi então programado para gerar o *start code* e o sinal do primeiro canal DMX-512 a 250 kbps. Dessa forma testava-se se a mesa controladora estava afetando o funcionamento do circuito e tem-se o controle completo do frame enviado. Essa programação também não teve êxito, persistido os problemas. Foi então reduzida à velocidade de

transmissão de 250 kbps para 115,2 kbps e reprogramados os módulos xbee. Com essa taxa de transmissão os módulos funcionaram corretamente sem perda de sincronismo nem de sinal.

Com base nessas informações, optou-se por colocar o micro-controlador no circuito para que ele fizesse a conversão dos quadros DMX-512 em sinais seriais e a redução de velocidade do sinal transmitido de 250 kbps para 115,2 kbps.

Os circuitos do transmissor e receptor foram integrados, e para isso foi necessário utilizar os pinos de controle do MAX-485. O quadro 11 são as tabelas verdade do MAX-485:

Table 1. Transmitting

INPUTS			OUTPUTS	
$\overline{RE}$	DE	DI	Z	Y
X	1	1	0	1
X	1	0	1	0
0	0	X	High-Z	High-Z
1	0	X	High-Z*	High-Z*

Table 2. Receiving

INPUTS			OUTPUT
$\overline{RE}$	DE	A-B	RO
0	0	≥ +0.2V	1
0	0	≤ -0.2V	0
0	0	Inputs open	1
1	0	X	High-Z*

Quadro 11 - Tabelas verdade do MAX485
Fonte: Maxim (2003)

Com as tabelas verdade dos pinos de controle, foram usadas rotinas de controle no programa do micro-controlador, integrando assim os módulos de recepção e transmissão usando somente um MAX-485 por protótipo.

Circuito receptor

Analogamente ao circuito transmissor, o circuito receptor foi dividido em duas partes. A primeira parte consiste no micro-controlador, usado para converter o sinal serial em um sinal DMX-512.

A segunda parte do circuito receptor consiste em converter o sinal elétrico de saída do PIC16F648A para os sinais RS-485.

O primeiro circuito usado para converter o sinal elétrico de saída do PIC nos sinais diferenciais, foi um circuito com dois amplificadores operacionais. O sinal de saída foi colocado na entrada não inversora do primeiro amplificador operacional e foi aterrada a entrada inversora. Assim, quando o sinal for maior na entrada não

inversora, o sinal de saída será +VCC (5 V). Quando o sinal na entrada não inversora for menor que o da entrada inversora, o sinal de saída será –VCC (0 V), gerando assim os dois níveis TTL (BOYLESTAD, NASHELSKY, 2006; NATIONAL, 2005).

O segundo amplificador operacional foi ligado de forma análoga ao primeiro, porém ligando o sinal de saída do PIC na entrada inversora, de forma que o sinal de saída será análogo ao sinal acima descrito, porém invertido (BOYLESTAD, NASHELSKY, 2006; NATIONAL, 2005).

A figura 49 apresenta o circuito que foi projetado inicialmente para gerar os sinais diferencias RS-485:

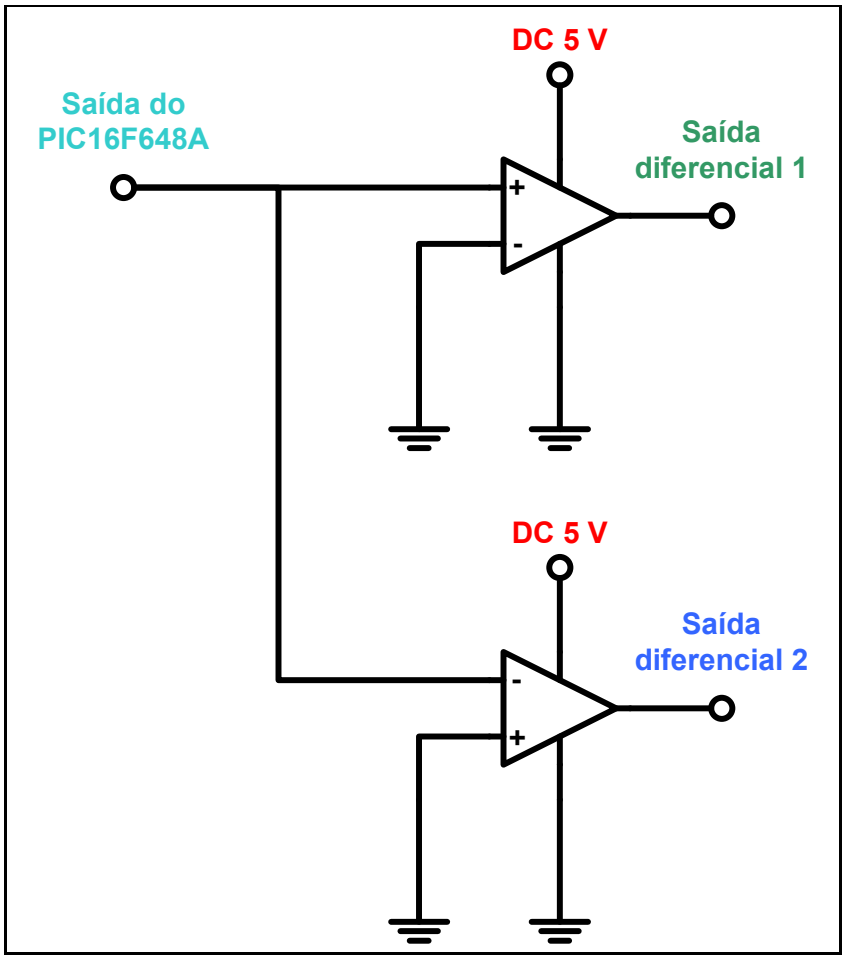


Figura 49 - Circuito de geração dos sinais diferenciais RS-485
Fonte: Adaptado de Boylestad, Nashelsky (2006) e National (2005)

O circuito descrito acima foi montado e não apresentou o resultado esperado, pois como uma das entradas estava aterrada, o sinal de “0” era comparado com 0 V. Como o sinal de zero possui uma tensão de cerca de 0,1 V e o

LM-392 possui boa sensibilidade, o sinal de saída ficava 5 V na saída do primeiro operacional e 0 V na saída do segundo operacional. Uma segunda tentativa foi feita utilizando um diodo diretamente polarizado com um resistor em série alimentado pelos 5 V. Com isso, obteve-se 0,45 V nas entradas que estavam aterradas, gerando os sinais, porém apareceu um ruído nas transições para o zero. Esse ruído, em determinados casos foi interpretado como erro, pois os dois sinais diferenciais apareciam como nível lógico “1” ou “0”.

Foi então analisada a folha de dados do integrado SN-7404. Esse integrado possui tempo de resposta que atende as necessidades do protótipo e interpreta o estado lógico “1” a partir de 2 V (TEXAS, 1983).

O sinal da saída do PIC foi colocado na entrada do SN-7404, que foi alimentado com 5 V, que está ilustrado na figura 71. O sinal de saída do SN-7404 foi ligado em outra entrada do SN-7404. As duas saídas do SN-7404 foram medidas e apresentaram 4,85 V, atendendo as especificações do RS-485. O atraso gerado pelo SN-7404 é muito pequeno e não interferiu no reconhecimento dos sinais. O circuito apresentava os dados corretamente, mas não era corretamente interpretado pelo dispositivo controlado. Em determinados casos o dispositivo controlado apresentava respostas a comando que não haviam sido dados. No osciloscópio os dados apareciam corretamente, porém o dispositivo não respondia corretamente. A figura 50 é o esquema elétrico usado para gerar os sinais diferencias através do SN-7404:

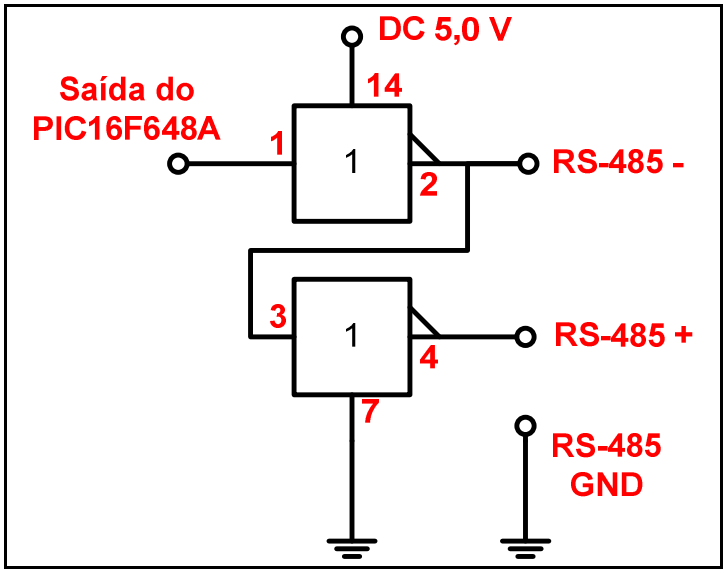


Figura 50 - SN7404 usado para as saídas diferenciais
Fonte: Adaptado de Texas (1983)

Analisou-se o *datasheet* do MAX-485 e segundo o manual esse circuito integrado também poderia ser usado para converter os sinais gerados pelo PIC em sinais RS-485. O circuito foi testado e o dispositivo controlado apresentou uma resposta correta com esse circuito.

APÊNDICE C.4 - Protótipo beta

Esse protótipo deveria inicialmente tratar o sinal DMX 512, deixando ele adequado para a entrada do CI XR-2206. O C.I. XR-2206 converteria os sinais de entrada em sinais FSK. A saída do C.I. XR-2206 iria para um circuito transmissor, que faria a transmissão do sinal de saída. A figura 51 ilustra o diagrama em blocos do circuito transmissor protótipo beta:

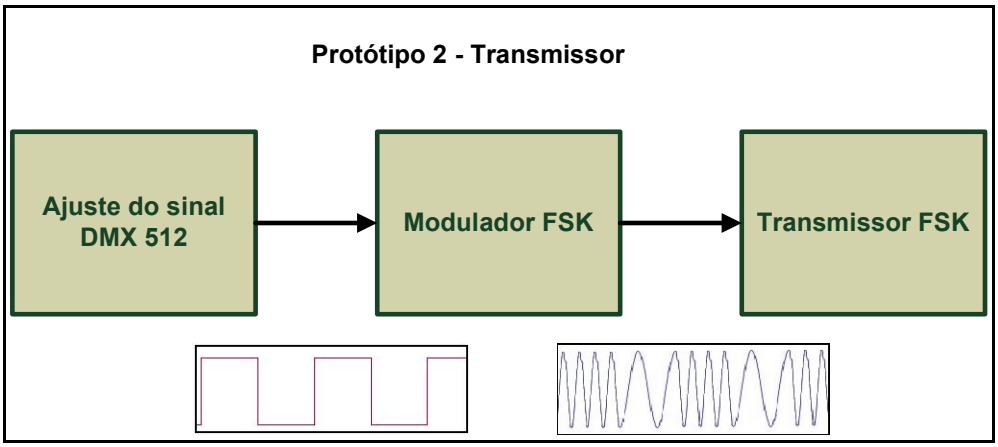


Figura 51 - Protótipo 2 – Transmissor FSK
Fonte: Autoria própria

Na recepção dividiu-se o protótipo em 3 blocos , conforme pode ser visto na figura 52. O primeiro bloco consiste em um circuito receptor, que faria a recepção e amplificação do sinal transmitido. No segundo bloco faria a filtragem do sinal e o último bloco é um circuito de conversão que iria tratar os dados e gerar os dados RS-485 necessários.

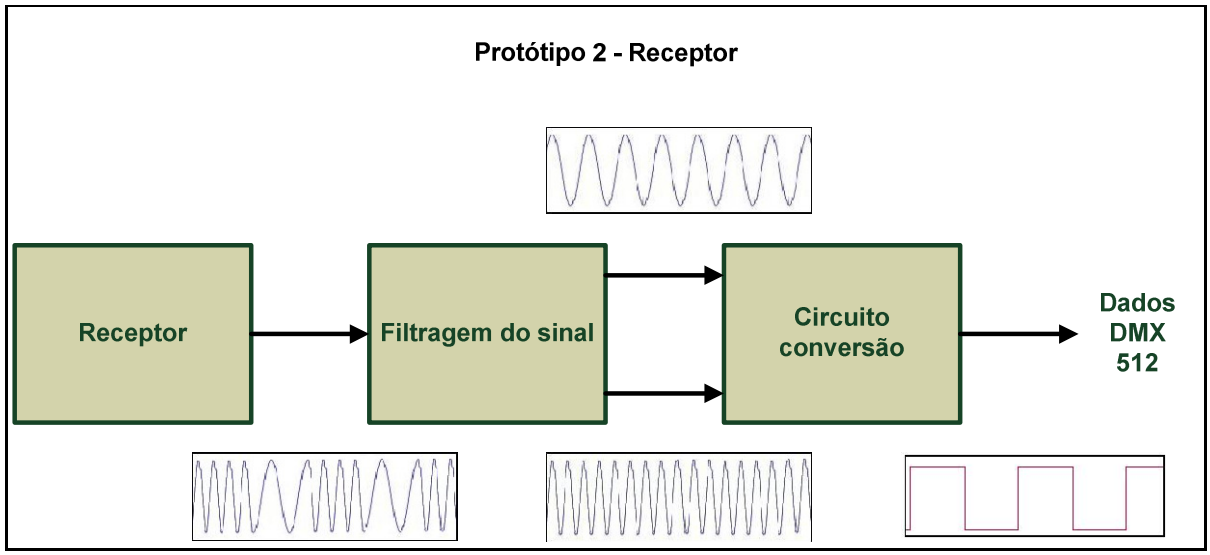


Figura 52 - Protótipo 2 – Receptor FSK
Fonte: Autoria própria

Circuito transmissor

O sinal DMX-512 de entrada foi colocado em um *buffer* inversor e seu sinal de saída foi injetado no CI XR-2206 usando sua configuração de modulador FSK. As frequências de trabalho foram definidas como 3 vezes a frequência do sinal e 5 vezes a frequência do sinal, respeitando o critério de Nyquist. Foi definido o valor do capacitor C_1 e foram calculados os demais componentes periféricos conforme exemplo encontrado na folha de dados do XR- 2206 (CASTRO, 2012; EXAR, 1972).

As figuras 53 e 54 são as equações usadas para o cálculo dos componentes, sendo a figura o circuito gerador de sinais FSK, usando o XR-2206:

$$f_1 = \frac{1}{R_1.C} \longrightarrow 1{,}25MHz = \frac{1}{R_1.1nF} \longrightarrow R_1 = \frac{1}{1{,}25.10^{6}.1.10^{-9}} \longrightarrow$$

$$R_1 = \frac{1}{1{,}25.10^{-3}} = 800\Omega$$

Recalculando f_1 para $R_1 = 820\Omega$

$$f_1 = \frac{1}{R_2.C} \longrightarrow f_1 = \frac{1}{820\Omega.1nF} \longrightarrow f_1 = \frac{1}{820.1.10^{-9}} \longrightarrow f_1 = 1{,}22MHz$$

Figura 53 - Cálculos de R1 e F1, usados no gerador FSK – 1,25 MHz
Fonte: Autoria própria

$$f_2 = \frac{1}{R_2.C} \longrightarrow 750kHz = \frac{1}{R_2.1nF} \longrightarrow R_2 = \frac{1}{750.10^3.1.10^{-9}} \longrightarrow$$

$$R_2 = \frac{1}{750.10^{-6}} = 1333{,}33\Omega$$

Recalculando f_2 para $R_2 = 1{,}2k\Omega$

$$f_2 = \frac{1}{R_2.C} \longrightarrow f_2 = \frac{1}{1{,}2k\Omega.1nF} \longrightarrow f_2 = \frac{1}{1200.1.10^{-9}} \longrightarrow f_1 = 833{,}33kHz$$

Figura 54 - Cálculos de R2 e F2, usados no gerador FSK – 750 kHz
Fonte: Autoria própria

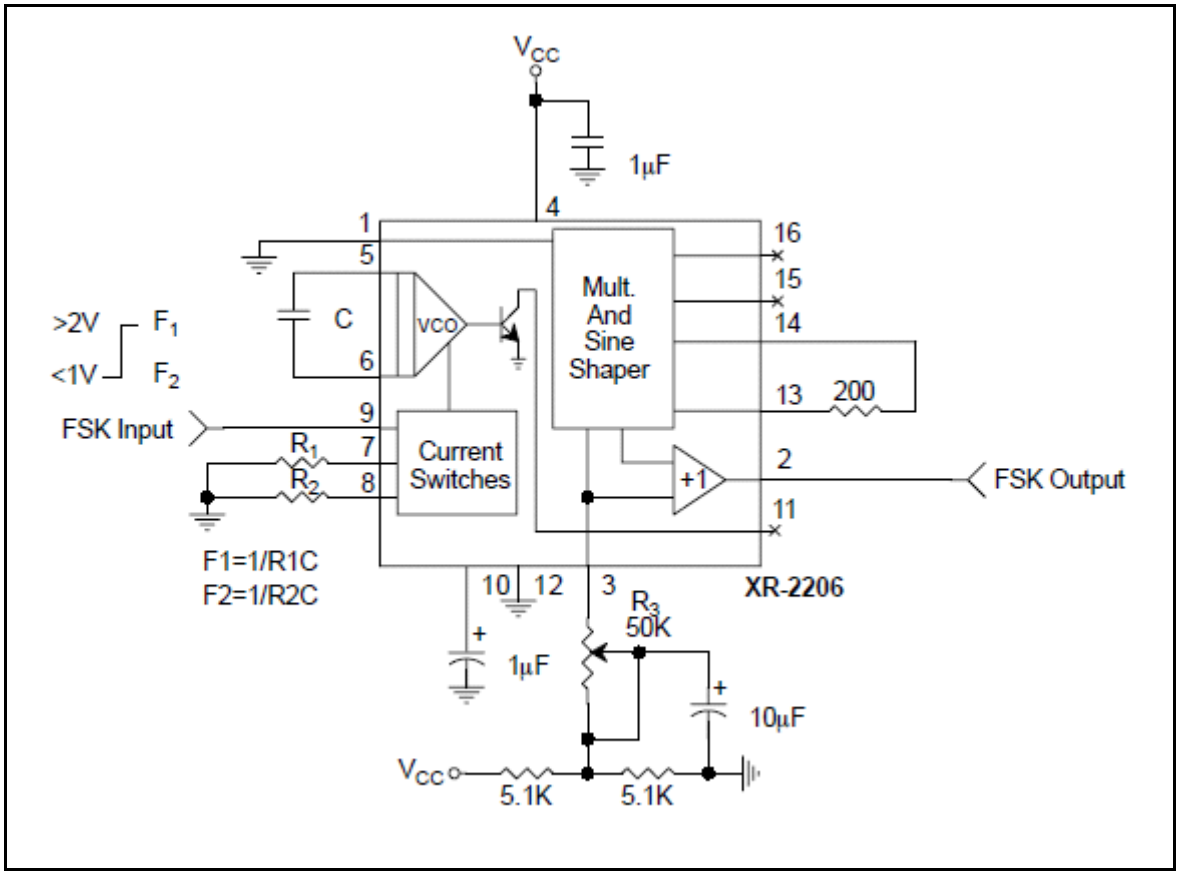


Figura 55 - Circuito gerador FSK usando XR-2206
Fonte: Exar (1972)

Os resistores de 5,1 kΩ foram substituídos por valores comerciais de 4,7 kΩ, o resistor de 200 Ω foi trocado pelo valor comercial de 220 Ω. Os resistores calculados foram substituídos por valores comerciais e as frequências foram

recalculadas. Os demais componentes foram mantidos. O resumo dos componentes está no quadro 12:

	Frequência desejada	Resistores calculados	Resistores comerciais	Frequência calculada
R1	1,25 MHz	800 Ω	820 Ω	1,22 MHz
R2	750 kHz	1333,33 Ω	1200 Ω	833,33 kHz
	Valor sugerido		**Valor usado**	
R3	50 kΩ		47 kΩ	
R4	5,1 kΩ		4,7 kΩ	
R5	5,1 kΩ		4,7 kΩ	
R6	200 Ω		220 Ω	
	Capacitor escolhido			
C	1nF			
	Valor sugerido		**Valor usado**	
C1	1 µF		1 µF	
C2	1 µF		1 µF	
C3	10 µF		10 µF	

Quadro 12 - Componentes do modulador FSK – 1250 e 750 kHz
Fonte: Autoria própria

Os circuitos foram primeiramente montados em *proto-board*, pois a frequência que trabalham não é muito alta (< 2 MHz). Ao injetar o sinal do circuito conversor (250 kHz) no circuito gerador FSK, foram capturados os seguintes sinais:

Os gráficos 12, 13, 14 e 15 apresentam, respectivamente, um sinal de entrada de 250 kHz, o sinal FSK quando a entrada está em “0”, o sinal FSK quando a entrada está em “1” e o sinal de entrada e o sinal modulado em FSK.

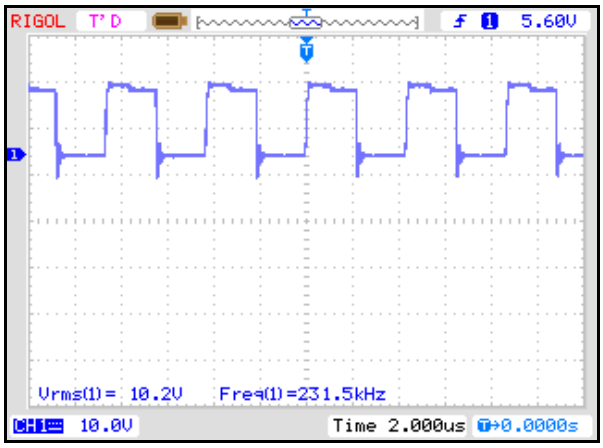


Gráfico 12 - Sinal de 250 kHz
Fonte: Autoria própria

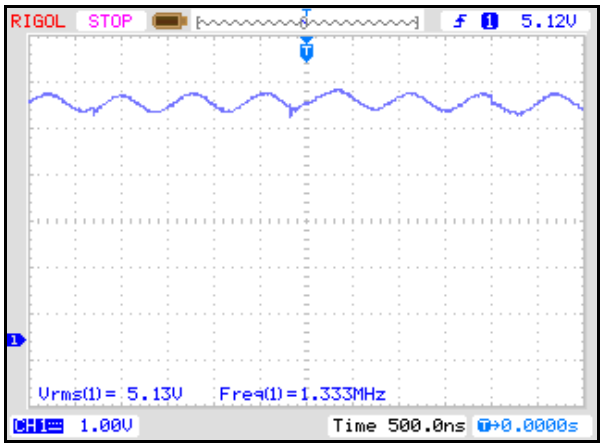


Gráfico 13 - Sinal de saída (entrada 0)
Fonte: Autoria própria

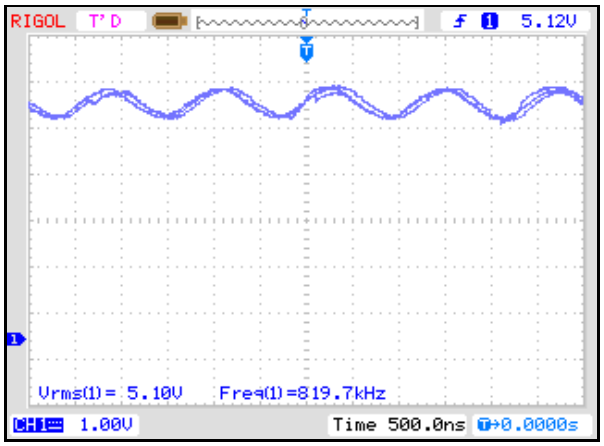


Gráfico 14 - Sinal de saída (entrada 1)
Fonte: Autoria própria

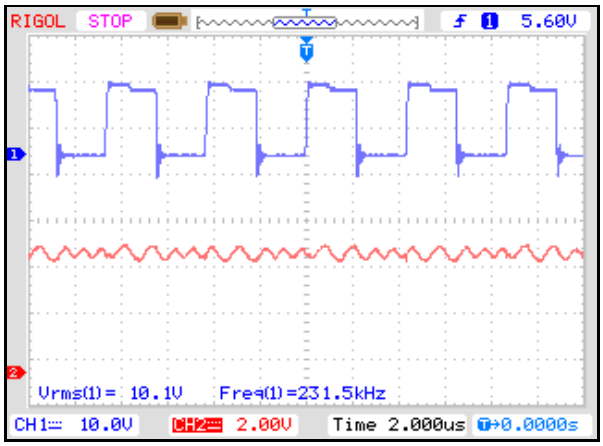


Gráfico 15 - Sinal modulado em FSK
Fonte: Autoria própria

A modulação ocorreu corretamente, porém era necessário um amplificador para melhorar o alcance, uma vez que o circuito não possuía alcance maior que 2 metros. Outro problema encontrado é que essas frequências se mostraram boas para a modulação do sinal, porém não estão adequadas com as normas de transmissão de sinais definidas pela ANATEL, pois interferem em outorgas para sinais de amplitude modulada – AM (ANATEL, 2006).

Foram recalculadas, analogamente ao descrito anteriormente frequências de trabalho abaixo de 500 kHz, com o primeiro sinal em 260 kHz e o segundo sinal em 300 kHz, conforme figuras 56 e 57:

$$f_1 = \frac{1}{R_1.C} \longrightarrow 260kHz = \frac{1}{R_1.1nF} \longrightarrow R_1 = \frac{1}{260.10^3.1.10^{-9}} \longrightarrow$$

$$R_1 = \frac{1}{260.10^{-6}} = 3846{,}15\Omega$$

Recalculando f_1 para $R_1 = 3900\Omega$

$$f_1 = \frac{1}{R_2.C} \longrightarrow f_1 = \frac{1}{820\Omega.1nF} \longrightarrow f_1 = \frac{1}{3{,}9.1.10^{-6}} \longrightarrow f_1 = 256{,}4kHz$$

Figura 56 - Cálculos de R1 e F1, usados no gerador FSK – 260 kHz
Fonte: Autoria própria

$$f_2 = \frac{1}{R_2.C} \longrightarrow 300kHz = \frac{1}{R_2.1nF} \longrightarrow R_2 = \frac{1}{300.10^3.1.10^{-9}} \longrightarrow$$

$$R_2 = \frac{1}{750.10^{-6}} = 3333{,}33\Omega$$

Recalculando f_2 para $R_2 = 3{,}3\,k\Omega$

$$f_2 = \frac{1}{R_2.C} \longrightarrow f_2 = \frac{1}{3{,}3k\Omega.1nF} \longrightarrow f_2 = \frac{1}{3{,}3.1.10^{-6}} \longrightarrow f_1 = 303{,}03kHz$$

Figura 57 - Cálculos de R2 e F2, usados no gerador FSK – 300 kHz
Fonte: Autoria própria

Os demais resistores e capacitores ficaram com o mesmo valor. Os demais componentes foram mantidos. O resumo dos componentes está no quadro 13:

	Frequência desejada	Resistores calculados	Resistores comerciais	Frequência calculada
R1	260 kHz	3846,15 ohm	3900 ohm	256410,26 Hz
R2	300 kHz	3333,33 ohm	3300 ohm	303030,30 Hz
	Valor sugerido		**Valor usado**	
R3	50 kΩ		50 kΩ	
R4	5,1 kΩ		4,7 kΩ	
R5	5,1 kΩ		4,7 kΩ	
R6	200 Ω		220 Ω	
	Capacitor escolhido			
C	1nF			
	Valor sugerido		**Valor usado**	
C1	1 µF		1 µF	
C2	1 µF		1 µF	
C3	10 µF		10 µF	

Quadro 13 - Componentes do modulador FSK – 300 e 260 kHz
Fonte: Autoria própria

Essas frequências foram escolhidas por atenderem as especificações da Agência Nacional de Telecomunicações – ANATEL – e por serem compatíveis com demoduladores FSK do receptor. (ANATEL, 2006)

O primeiro problema encontrado foi o de não respeitar o critério de Nyquist para uma correta taxa de amostras. Contudo, foram feitos os testes em uma frequência simulada de 125 kHz, para respeitar o critério de Nyquist (2,08 para 260 kHz e 2,4 para 300 kHz). Caso o circuito apresentasse boas características o sinal

de entrada RS-485 seria tratado com técnicas de compactação, envio de mais de um símbolo por bit ou um *buffer* de entrada, uma vez que existe um intervalo entre as informações enviadas, conforme o item 3.1.

Como o alcance do sinal diminuiu para 1 metro, foram pesquisados e testados amplificadores transistorizados e antenas, usados em sinais de AM, para melhorar o alcance, como o da figura 58. Além de o alcance ficar abaixo do esperado, cerca de 20 metros o sinal interferia com harmônicos em sinais AM próximos da faixa de 500 kHz. Mesmo melhorando a antena na saída e aumentando a tensão de alimentação do transistor para 18 V, que permitiu um alcance de 30 metros em vão livre, o sinal aumentou sua interferência com harmônicas na faixa dos 500 kHz, praticamente impossibilitando a recepção de rádios (ELETROHOO, 2012).

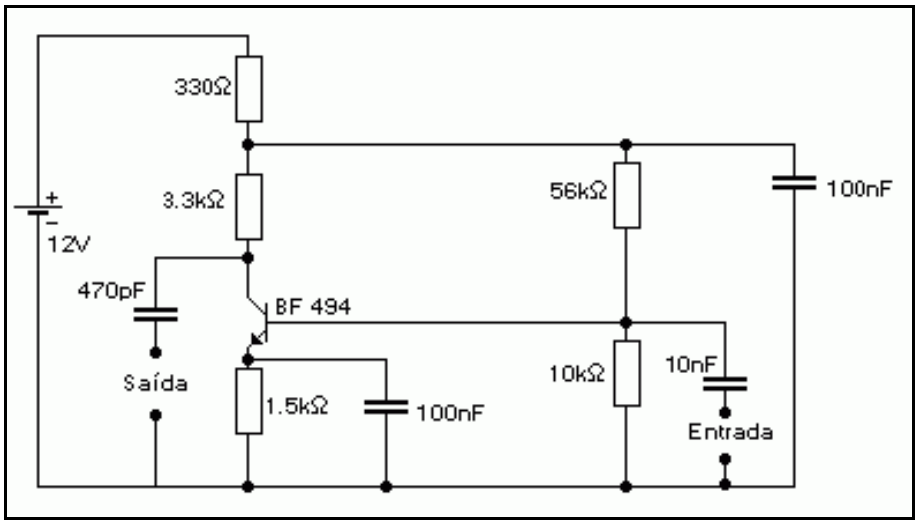


Figura 58 - Circuito amplificador AM
Fonte: Eletrohoo (2012)

Devido ao comprimento de onda e a potência, obstáculos entre o receptor e o transmissor geraram atenuações que praticamente impossibilitavam sua recepção. O sinal também apresentava conflitos com sinais outorgados pela ANATEL, ou seja, interferia e sofria interferências de sinais na faixa de frequência escolhida.

De forma resumida, o módulo transmissor apresentou as seguintes dificuldades para seu pleno funcionamento:

- Conflito com faixas de frequências outorgadas pela ANATEL;
- Frequências de sinais muito próximas;

- Sinais obedecendo ao critério de Nyquist, porém sem uma boa taxa de amostras, gerando em alguns casos interpretação errada do sinal;
- Interferência de harmônicas dentro de faixas outorgadas pela ANATEL;
- Necessidade de circuitos adicionais para transmitir dados a 250 kbps (padrão DMX-512);
- Alcance menor que o esperado;
- Dificuldade de transmissão em espaços que não possuem vãos livres;

Devido essas dificuldades encontradas para permitir o funcionamento do protótipo beta, a equipe decidiu descartar este protótipo. Foram discutidas outras possibilidades, como um modulador QAM e outras técnicas de modulação, porém elas se mostraram inviáveis devido aos custos envolvidos e não atenderem as premissas previamente estabelecidas no início do projeto.

Circuito receptor

O protótipo do receptor utilizou diversos circuitos, que foram montados e desmontados, pois não atendiam as necessidades do projeto. Foram pesquisados circuitos integrados demoduladores FSK, dentre os quais foi usado o XR-2211. Ele foi desconsiderado inicialmente devido às frequências de trabalho escolhidas (1250 e 750 kHz), porém como a frequência foi alterada para ser menor que 300 kHz, esse circuito integrado foi testado para a demodulação. O esquema desse circuito é apresentado na figura 59:

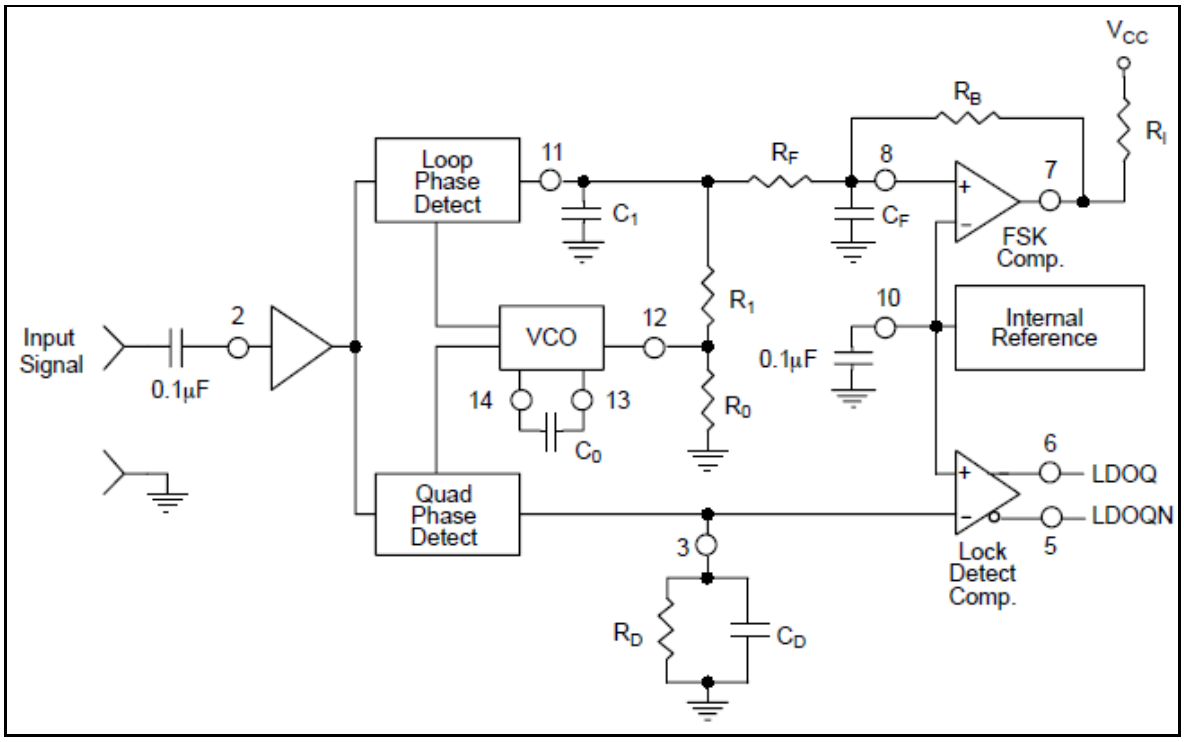


Figura 59 - Diagrama simplificado do XR-2211
Fonte: Exar (1995)

O cálculo dos componentes foi feito conforme os cálculos da figura 60 e 61 (EXAR, 1995):

$$1) f_1 = 260kHz; f_2 = 300kHz; \rightarrow f_0 = \sqrt{f_2 - f_1} \rightarrow \sqrt{40k} = 200Hz$$

$$2) R_0 = 100k\Omega$$

$$3) C_0 = \frac{1}{R_0.f_0} = \frac{1}{100k.200} = 50nF \rightarrow 47nF$$

$$4) R_1 = \left(\frac{R_0.f_0}{f_2 - f_1}\right).2 = \left(\frac{100k.200}{300k - 260k}\right).2 = 1k\Omega$$

$$5) \varsigma = 0,5 \rightarrow \text{Recomendado}; \rightarrow$$
$$C_1 = \frac{1250.C_0}{R_1.\varsigma^2} = \frac{1250.50n}{1k.0,5^2} = 250nF \rightarrow 100nF + 150nF$$

Figura 60 - Cálculos dos componentes usados no circuito receptor
Fonte: Exar (1995)

$$6) R_F = 5.R_0 = 5.100k = 500k\Omega \rightarrow 470k\Omega$$

$$7) R_B = 5.R_F = 5.500k = 2500k\Omega \rightarrow 1M\Omega + 1{,}5M\Omega$$

$$8) R_{SUM} = \frac{(R_F + R_1).R_B}{(R_F + R_1 + R_B)} = 417{,}4k\Omega$$

$$9) C_F = \frac{0{,}25}{R_{SUM}.BR} = \frac{0{,}25}{417{,}4k.125k} = 4{,}79pF \rightarrow 4{,}7pF$$

Figura 61 - Cálculos dos componentes usados no circuito receptor - 2
Fonte: Exar (1995)

O quadro 14 apresenta os componentes utilizados no circuito receptor:

	Calculado	Usado
R0	100k	Pot. 100k
C0	50nF	47nF
R1	1k	1k
C1	250nF	150nF+100nF
RF	500k	470k
RB	2500k	1M+1,5M
CF	4,79pF	4,7pF

Quadro 14 - Componentes utilizados no circuito receptor
Fonte: Autoria própria

O protótipo do receptor foi montado e usado para a recepção dos sinais FSK gerados pelo transmissor, porém o circuito que converteria a saída do demodulador FSK em um sinal DMX não foi elaborado devido à inviabilidade do circuito transmissor.

APÊNDICE D - CIRCUITO FINAL

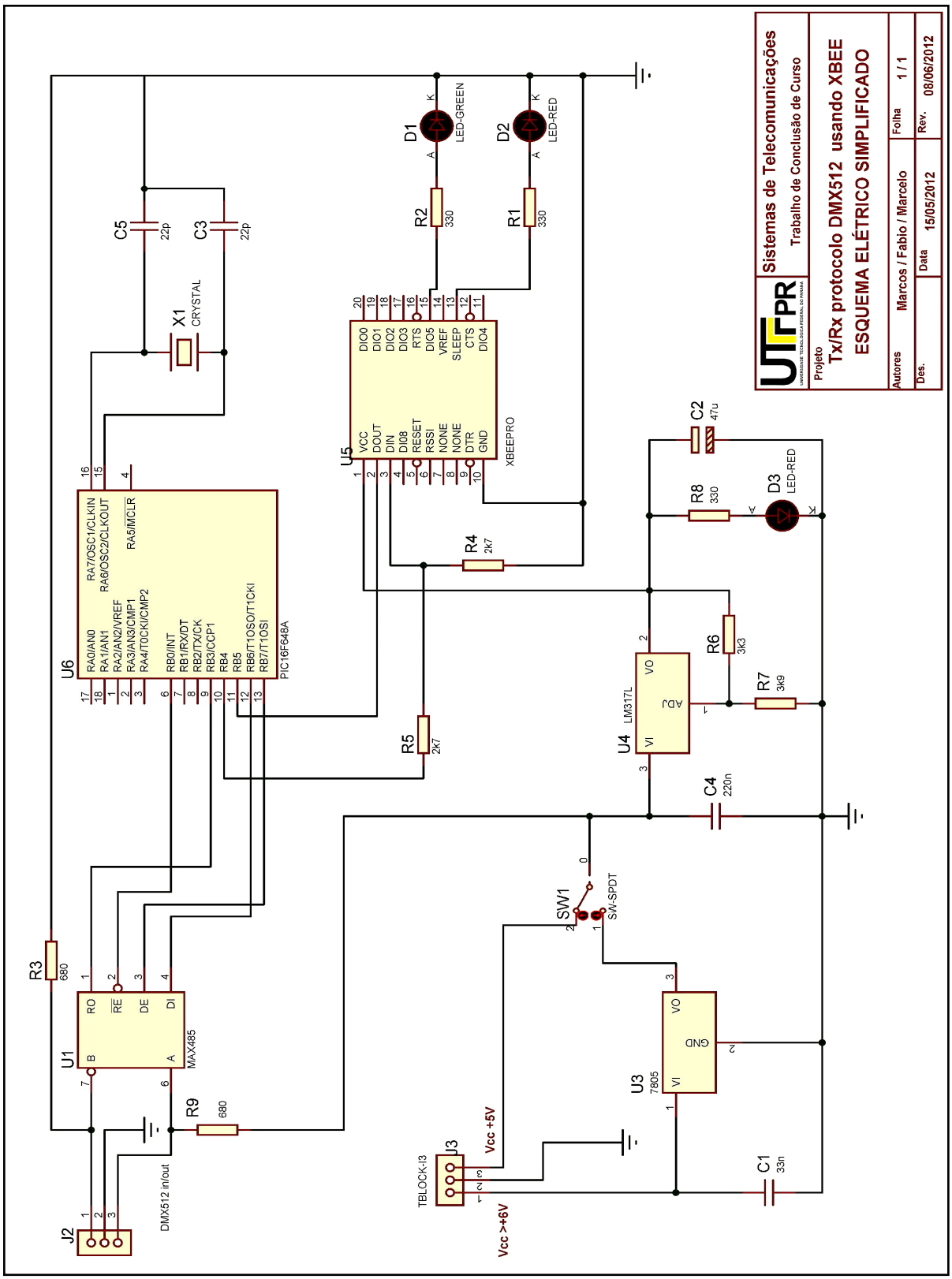


Figura 62 - Esquema elétrico simplificado – Circuito final
Fonte: Autoria própria

APÊNDICE E - PROGRAMA USADO NO PIC16F648A

```
#include <16F648A.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES HS                       //High speed Osc (> 4 MHz)
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOPROTECT                //Code not protected from reading
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES NOMCLR                   //Master Clear pin used for I/O
#FUSES NOLVP   /*Low Voltage Programming on B3 (PIC16) or B5 (PIC18)*/
#FUSES NOCPD                    //No EE protection

/* Instruções deixadas inalteradas após usar o wizard. O wizard
Perguntou sobre a necessidade de interrupções, watch dog, timer,
comparadores, etc.
Todos foram desabilitados para não interferirem no programa


O programa foi elaborado não utilizando timers nem a porta serial
presente no PIC, devido ao tempo de resposta e problemas com as
interrupções e sincronismo do sinal, porém essas funcionalidades
podem ser utilizadas para aprimorar o código */

#use delay(clock=20000000)      //Define o clock externo como 20 MHz
#use fast_io(a)                 //Define a porta A no modo fast IO
#use fast_io(b)                 //Define a porta A no modo fast IO

#define out_en pin_b0           //pino físico 6
//#define s0rx pin_b1           //pino físico 7, rx porta serial
//#define s0tx pin_b2           //pino físico 8, tx porta serial
#define s1rx pin_b3             //pino físico 9
#define s1tx pin_b4             //pino físico 10
#define s2rx pin_b5             //pino físico 11
#define s2tx pin_b6             //pino físico 12
#define in_en pin_b7            //pino físico 13

int16 i;                         //Contador de 16 bits
/*A variável acima foi descrita como 16 bits para que o contador do
laço for que reconstrói o quadro DMX-512 pudesse ser executado
corretamente. Se fosse usada uma variável inteira comum o laço só poderia
ir até 256. Com 16 bits, pode-se ir até 65536*/

/*As variáveis abaixo são de 1 bit e foram declaradas de forma discreta
devido ao tempo usado para ler e retornar os dados do vetor. O código foi
testado com diversos tipos de variáveis sendo essas as que apresentaram
```

```
melhor resultado. A variável d0x armazena o start code, e as demais
armazenam as informações dos 12 canais */

int1 d00, d01, d02, d03, d04, d05, d06, d07;
int1 d10, d11, d12, d13, d14, d15, d16, d17;
int1 d20, d21, d22, d23, d24, d25, d26, d27;
int1 d30, d31, d32, d33, d34, d35, d36, d37;
int1 d40, d41, d42, d43, d44, d45, d46, d47;
int1 d50, d51, d52, d53, d54, d55, d56, d57;
int1 d60, d61, d62, d63, d64, d65, d66, d67;
int1 d70, d71, d72, d73, d74, d75, d76, d77;
int1 d80, d81, d82, d83, d84, d85, d86, d87;
int1 d90, d91, d92, d93, d94, d95, d96, d97;
int1 da0, da1, da2, da3, da4, da5, da6, da7;    // a = 10 em hexadecimal
int1 db0, db1, db2, db3, db4, db5, db6, db7;    // b = 11 em hexadecimal
int1 dc0, dc1, dc2, dc3, dc4, dc5, dc6, dc7;    // c = 12 em hexadecimal

void receptor(porta, delay){

   /* Essa função pega a porta especificada e o delay enviado e recebe os
   dados do DMX512 ou do Xbee */
   loop0:
   /*Foram testados todos os laços e tentou-se usar funções recursivas.
   Os laços apresentavam problemas por serem lentos demais (16 micros-
   segundos para execução completa), o que afetava a recepção e
   transmissão de dados. Dessa forma foi usado um laço com goto, que
   é mais próxima da linguagem nativa (assembly) do micro-controlador.
   O laço goto demora menos de 1 micro-segundo para sua execução*/
   if (!input(porta)){          delay_us(delay);

   /*Aguarda que o pino 11 fique na condição 0, retornando para o loop0
   até que a condição seja satisfeita. Apos a condição ser satisfeita,
   coleta os dados do start code*/
      d00=input(porta);         delay_us(delay);

      /*A linha acima pegar um dado da entrada e aguarda o delay de
      micro-segundos garantindo que a próxima instrução pegue o bit na
      posição correta.
      O tempo para 250 kbps é de 4 micro-segundos, porem com o tempo de
      de execução mais o delay de 3 esse tempo é respeitado. A mesma
      instrução é repetida para os demais bits dos canais
      O tempo para 115.2 kbps é de 8.65 micro-segundos, porem com o tempo
      de execução mais o delay de 8 esse tempo é respeitado. A mesma
      instrução é repetida para os demais bits dos canais*/
      d01=input(porta);         delay_us(delay);
      /* As instruções foram colocadas na mesma linha para simplificar
      a visualização de cada bit sendo coletado */
      d02=input(porta);         delay_us(delay);
      d03=input(porta);         delay_us(delay);
```

```
   d04=input(porta);          delay_us(delay);
   d05=input(porta);          delay_us(delay);
   d06=input(porta);          delay_us(delay);
   d07=input(porta);          delay_us(delay);          delay_us(delay);
   /* Esse último bloco de instruções possui dois delay, um para cada
   stop bit */
}
else
   goto loop0;

loop1:
if (!input(porta)){           delay_us(delay);
   d10=input(porta);          delay_us(delay);
   d11=input(porta);          delay_us(delay);
   d12=input(porta);          delay_us(delay);
   d13=input(porta);          delay_us(delay);
   d14=input(porta);          delay_us(delay);
   d15=input(porta);          delay_us(delay);
   d16=input(porta);          delay_us(delay);
   d17=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop1;

loop2:
if (!input(porta)){           delay_us(delay);
   d20=input(porta);          delay_us(delay);
   d21=input(porta);          delay_us(delay);
   d22=input(porta);          delay_us(delay);
   d23=input(porta);          delay_us(delay);
   d24=input(porta);          delay_us(delay);
   d25=input(porta);          delay_us(delay);
   d26=input(porta);          delay_us(delay);
   d27=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop2;

loop3:
if (!input(porta)){           delay_us(delay);
   d30=input(porta);          delay_us(delay);
   d31=input(porta);          delay_us(delay);
   d32=input(porta);          delay_us(delay);
   d33=input(porta);          delay_us(delay);
   d34=input(porta);          delay_us(delay);
   d35=input(porta);          delay_us(delay);
   d36=input(porta);          delay_us(delay);
   d37=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop3;

loop4:
if (!input(porta)){           delay_us(delay);
   d40=input(porta);          delay_us(delay);
   d41=input(porta);          delay_us(delay);
   d42=input(porta);          delay_us(delay);
   d43=input(porta);          delay_us(delay);
   d44=input(porta);          delay_us(delay);
   d45=input(porta);          delay_us(delay);
```

```
   d46=input(porta);          delay_us(delay);
   d47=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop4;

loop5:
if (!input(porta)){           delay_us(delay);
   d50=input(porta);          delay_us(delay);
   d51=input(porta);          delay_us(delay);
   d52=input(porta);          delay_us(delay);
   d53=input(porta);          delay_us(delay);
   d54=input(porta);          delay_us(delay);
   d55=input(porta);          delay_us(delay);
   d56=input(porta);          delay_us(delay);
   d57=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop5;

loop6:
if (!input(porta)){           delay_us(delay);
   d60=input(porta);          delay_us(delay);
   d61=input(porta);          delay_us(delay);
   d62=input(porta);          delay_us(delay);
   d63=input(porta);          delay_us(delay);
   d64=input(porta);          delay_us(delay);
   d65=input(porta);          delay_us(delay);
   d66=input(porta);          delay_us(delay);
   d67=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop6;

loop7:
if (!input(porta)){           delay_us(delay);
   d70=input(porta);          delay_us(delay);
   d71=input(porta);          delay_us(delay);
   d72=input(porta);          delay_us(delay);
   d73=input(porta);          delay_us(delay);
   d74=input(porta);          delay_us(delay);
   d75=input(porta);          delay_us(delay);
   d76=input(porta);          delay_us(delay);
   d77=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop7;

loop8:
if (!input(porta)){           delay_us(delay);
   d80=input(porta);          delay_us(delay);
   d81=input(porta);          delay_us(delay);
   d82=input(porta);          delay_us(delay);
   d83=input(porta);          delay_us(delay);
   d84=input(porta);          delay_us(delay);
   d85=input(porta);          delay_us(delay);
   d86=input(porta);          delay_us(delay);
   d87=input(porta);          delay_us(delay);          delay_us(delay);
}
else
   goto loop8;
```

```
    loop9:
    if (!input(porta)){          delay_us(delay);
       d90=input(porta);         delay_us(delay);
       d91=input(porta);         delay_us(delay);
       d92=input(porta);         delay_us(delay);
       d93=input(porta);         delay_us(delay);
       d94=input(porta);         delay_us(delay);
       d95=input(porta);         delay_us(delay);
       d96=input(porta);         delay_us(delay);
       d97=input(porta);         delay_us(delay);         delay_us(delay);
    }
    else
       goto loop9;

    loop10:
    if (!input(porta)){          delay_us(delay);
       da0=input(porta);         delay_us(delay);
       da1=input(porta);         delay_us(delay);
       da2=input(porta);         delay_us(delay);
       da3=input(porta);         delay_us(delay);
       da4=input(porta);         delay_us(delay);
       da5=input(porta);         delay_us(delay);
       da6=input(porta);         delay_us(delay);
       da7=input(porta);         delay_us(delay);         delay_us(delay);
    }
    else
       goto loop10;

    loop11:
    if (!input(porta)){          delay_us(delay);
       db0=input(porta);         delay_us(delay);
       db1=input(porta);         delay_us(delay);
       db2=input(porta);         delay_us(delay);
       db3=input(porta);         delay_us(delay);
       db4=input(porta);         delay_us(delay);
       db5=input(porta);         delay_us(delay);
       db6=input(porta);         delay_us(delay);
       db7=input(porta);         delay_us(delay);         delay_us(delay);
    }
    else
       goto loop11;

    loop12:
    if (!input(porta)){          delay_us(delay);
       dc0=input(porta);         delay_us(delay);
       dc1=input(porta);         delay_us(delay);
       dc2=input(porta);         delay_us(delay);
       dc3=input(porta);         delay_us(delay);
       dc4=input(porta);         delay_us(delay);
       dc5=input(porta);         delay_us(delay);
       dc6=input(porta);         delay_us(delay);
       dc7=input(porta);         delay_us(delay);         delay_us(delay);
    }
    else
       goto loop12;

}

void transmissor(porta, delay){
    //start code
```

```
output_low(porta);          delay_us(delay);
output_bit(porta, d00);     delay_us(delay);
output_bit(porta, d01);     delay_us(delay);
output_bit(porta, d02);     delay_us(delay);
output_bit(porta, d03);     delay_us(delay);
output_bit(porta, d04);     delay_us(delay);
output_bit(porta, d05);     delay_us(delay);
output_bit(porta, d06);     delay_us(delay);
output_bit(porta, d07);     delay_us(delay);
output_high(porta);         delay_us(50);
/*stop bits e um tempo adicional, mínimo de 8 micro-segundos, máximo de
1 segundo*/
//1 byte
output_low(porta);          delay_us(delay);
output_bit(porta, d10);     delay_us(delay);
output_bit(porta, d11);     delay_us(delay);
output_bit(porta, d12);     delay_us(delay);
output_bit(porta, d13);     delay_us(delay);
output_bit(porta, d14);     delay_us(delay);
output_bit(porta, d15);     delay_us(delay);
output_bit(porta, d16);     delay_us(delay);
output_bit(porta, d17);     delay_us(delay);
output_high(porta);         delay_us(50);

//2 byte
output_low(porta);          delay_us(delay);
output_bit(porta, d20);     delay_us(delay);
output_bit(porta, d21);     delay_us(delay);
output_bit(porta, d22);     delay_us(delay);
output_bit(porta, d23);     delay_us(delay);
output_bit(porta, d24);     delay_us(delay);
output_bit(porta, d25);     delay_us(delay);
output_bit(porta, d26);     delay_us(delay);
output_bit(porta, d27);     delay_us(delay);
output_high(porta);         delay_us(50);

//3 byte
output_low(porta);          delay_us(delay);
output_bit(porta, d30);     delay_us(delay);
output_bit(porta, d31);     delay_us(delay);
output_bit(porta, d32);     delay_us(delay);
output_bit(porta, d33);     delay_us(delay);
output_bit(porta, d34);     delay_us(delay);
output_bit(porta, d35);     delay_us(delay);
output_bit(porta, d36);     delay_us(delay);
output_bit(porta, d37);     delay_us(delay);
output_high(porta);         delay_us(50);

//4 byte
output_low(porta);          delay_us(delay);
output_bit(porta, d40);     delay_us(delay);
output_bit(porta, d41);     delay_us(delay);
output_bit(porta, d42);     delay_us(delay);
output_bit(porta, d43);     delay_us(delay);
output_bit(porta, d44);     delay_us(delay);
output_bit(porta, d45);     delay_us(delay);
output_bit(porta, d46);     delay_us(delay);
output_bit(porta, d47);     delay_us(delay);
```

```
output_high(porta);          delay_us(50);

//5 byte
output_low(porta);           delay_us(delay);
output_bit(porta, d50);      delay_us(delay);
output_bit(porta, d51);      delay_us(delay);
output_bit(porta, d52);      delay_us(delay);
output_bit(porta, d53);      delay_us(delay);
output_bit(porta, d54);      delay_us(delay);
output_bit(porta, d55);      delay_us(delay);
output_bit(porta, d56);      delay_us(delay);
output_bit(porta, d57);      delay_us(delay);
output_high(porta);          delay_us(50);

//6 byte
output_low(porta);           delay_us(delay);
output_bit(porta, d60);      delay_us(delay);
output_bit(porta, d61);      delay_us(delay);
output_bit(porta, d62);      delay_us(delay);
output_bit(porta, d63);      delay_us(delay);
output_bit(porta, d64);      delay_us(delay);
output_bit(porta, d65);      delay_us(delay);
output_bit(porta, d66);      delay_us(delay);
output_bit(porta, d67);      delay_us(delay);
output_high(porta);          delay_us(50);

//7 byte
output_low(porta);           delay_us(delay);
output_bit(porta, d70);      delay_us(delay);
output_bit(porta, d71);      delay_us(delay);
output_bit(porta, d72);      delay_us(delay);
output_bit(porta, d73);      delay_us(delay);
output_bit(porta, d74);      delay_us(delay);
output_bit(porta, d75);      delay_us(delay);
output_bit(porta, d76);      delay_us(delay);
output_bit(porta, d77);      delay_us(delay);
output_high(porta);          delay_us(50);

//8 byte
output_low(porta);           delay_us(delay);
output_bit(porta, d80);      delay_us(delay);
output_bit(porta, d81);      delay_us(delay);
output_bit(porta, d82);      delay_us(delay);
output_bit(porta, d83);      delay_us(delay);
output_bit(porta, d84);      delay_us(delay);
output_bit(porta, d85);      delay_us(delay);
output_bit(porta, d86);      delay_us(delay);
output_bit(porta, d87);      delay_us(delay);
output_high(porta);          delay_us(50);

//9 byte
output_low(porta);           delay_us(delay);
output_bit(porta, d90);      delay_us(delay);
output_bit(porta, d91);      delay_us(delay);
output_bit(porta, d92);      delay_us(delay);
output_bit(porta, d93);      delay_us(delay);
output_bit(porta, d94);      delay_us(delay);
output_bit(porta, d95);      delay_us(delay);
output_bit(porta, d96);      delay_us(delay);
```

```
   output_bit(porta, d97);      delay_us(delay);
   output_high(porta);          delay_us(50);

   //10 byte
   output_low(porta);           delay_us(delay);
   output_bit(porta, da0);      delay_us(delay);
   output_bit(porta, da1);      delay_us(delay);
   output_bit(porta, da2);      delay_us(delay);
   output_bit(porta, da3);      delay_us(delay);
   output_bit(porta, da4);      delay_us(delay);
   output_bit(porta, da5);      delay_us(delay);
   output_bit(porta, da6);      delay_us(delay);
   output_bit(porta, da7);      delay_us(delay);
   output_high(porta);          delay_us(50);

   //11 byte
   output_low(porta);           delay_us(delay);
   output_bit(porta, db0);      delay_us(delay);
   output_bit(porta, db1);      delay_us(delay);
   output_bit(porta, db2);      delay_us(delay);
   output_bit(porta, db3);      delay_us(delay);
   output_bit(porta, db4);      delay_us(delay);
   output_bit(porta, db5);      delay_us(delay);
   output_bit(porta, db6);      delay_us(delay);
   output_bit(porta, db7);      delay_us(delay);
   output_high(porta);          delay_us(50);

}

 void transmissor2(porta, delay){
/* Ocorreu um erro de compilação devido ao número de instruções na função
transmissor ter ficado muito alta. Foi criada a função transmissor2 para
que o número de instruções ficasse adequado na função transmissor*/
   //12 byte
   output_low(porta);           delay_us(delay);
   output_bit(porta, dc0);      delay_us(delay);
   output_bit(porta, dc1);      delay_us(delay);
   output_bit(porta, dc2);      delay_us(delay);
   output_bit(porta, dc3);      delay_us(delay);
   output_bit(porta, dc4);      delay_us(delay);
   output_bit(porta, dc5);      delay_us(delay);
   output_bit(porta, dc6);      delay_us(delay);
   output_bit(porta, dc7);      delay_us(delay);
   output_high(porta);          delay_us(50);

}

void main()
{
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);
   setup_comparator(NC_NC_NC_NC);
   setup_vref(FALSE);
   setup_oscillator(False);
   /*Opções padrão do wizard do compilador*/
   set_tris_a(0b00000000);
   //           76543210
```

```
set_tris_b(0b00101000);
/*Como o PIC está em modo Fast IO é necessário informar ao registrador
o modo das portas. No caso, a porta A está com todos os pinos como saída
(0=Output=Saída e 1=Input=Entrada) a porta B está com os pinos 9 e 11
como entradas */

output_high (s1tx);       //Envia "1" para a entrada do xbee
output_high (s2tx);       //Envia "1" para a entrada do MAX485
output_low (out_en);
output_low (in_en);

/* As duas instruções acima colocam o MAX-485 em modo receptor, ou seja,
as portas A e B ficaram para receber dados, que é a opção padrão. Quando
o PIC precisar transmitir dados, ele irá colocar o in_en em 1, fazendo
com que o MAX-485 saia do modo de receber dados e entre no modo de
transmissão.
No modo receptor, o MAX-485 irá converter o que receber pelas portas
A/B, que são RS-485 em dados TTL. No modo transmissor irá converter o
que receber em sua porta TTL em dados RS-485 pelas portas A/B. */
   for(;;){
   /* Laço infinito que alterna entre o modo xbee receptor e transmissor
   Quando o xbee está em modo transmissor, o MAX-485 deve receber os
   dados para enviar para o PIC e o PIC enviará os dados recebidos para
   o xbee, que fará a transmissão de dados.
   Quando o xbee está em modo receptor, o MAX-485 deve receber os dados
   TTL do PIC e enviar para as portas RS-485 A/B como sinais
   diferenciais. */
   // Modo xbee transmissor
   if(!input(s1rx)){              delay_us(50);
      if (!input(s1rx)){          delay_us(50);
      /* As instruções acima garantem que o PIC está pegando um dado TTL
      correspondente ao start frame. Após pegar o start frame o PIC irá
      descartá-lo, aproveitando somente a informação. */
         loopa:
         if(input(s1rx)){
            receptor(s1rx, 3);
            //Chama a função receptor(porta, delay)
            /* Essa função recebe os dados a 250 kbps e armazena os
            Dados para serem usados no transmissor. Ela memoriza o start
            code e mais 12 canais, identificando quando inicia o start
            code*/
            transmissor(s1tx, 8);
            transmissor2(s1tx, 8);
            //Chama a função transmissor(porta, delay)
            /* Essa função transmite os dados a 115,2 kbps. Ela irá
            tirar o start frame, transmitir o start code e os 12 canais
```

```
            memorizados*/
         }
         else
            goto loopa;
      }
   }

   /*Modo receptor*/
   if (!input(s2rx)){            delay_ms(1);         delay_us(200);

      /*Se o pino 11 for para a condição de 0, o programa irá entrar no
      Modo receptor e aguardar 1,2 milissegundos garantindo que o
      programa pegue o start code corretamente, uma vez que o modulo
      transmissor só irá transmitir o start code e 12 canais*/

      receptor(s2rx, 8);      //Chama a função receptor(porta, delay)

      /* Essa função recebe os dados a 115,2 kbps e armazena os dados
      para serem usados no transmissor. Ela memoriza o start code e mais
      12 canais, identificando quando inicia o start code*/
      /*Término da coleta das informações dos canais, iniciando a
      transmissão. A instrução abaixo gera o start frame (sequência de
      zeros*/
      output_high (in_en);
      /*habilita a recepção de dados na entrada do MAX 485*/
      output_low(s2tx);          delay_ms(1);
      //Break
      output_high(s2tx);         delay_us(12);
      //MAB
      transmissor(s2tx, 3);    //Chama a função transmissor(porta, delay)
      transmissor2(s2tx, 3);   //Chama a função transmissor(porta, delay)

      /* Essa função transmite os dados a 250kbps em frames DMX512. Ela
      irá gerar o start frame, transmitir o start code e os 12 canais
      memorizados e transmitir mais 500 canais vazios para completar o
      frame dmx512*/
      for (i=13; i<=512; ++i){
         output_low(s2tx);          delay_us(35);
         output_high(s2tx);         delay_us(35);
      }

      output_low(in_en);
      /*desabilita a recepção de dados na entrada do MAX 485*/

   }
  }
}
```