Universidade de Brasília – UnB
Campus Gama – FGA
Engenharia Eletrônica

Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível

Kaio Diego de Araújo Coelho

e

Paulo Afonso Dutra de Alarcão Vieira

Orientador: Dr. Leonardo Aguayo

Kaio Diego de Araújo Coelho
e
Paulo Afonso Dutra de Alarcão Vieira

Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Orientador: Dr. Leonardo Aguayo

Brasília, DF
2017

Brasília/DF, Dezembro de 2017

Ficha Catalográfica

Kaio Diego de Araújo Coelho e Paulo Afonso Dutra de Alarcão Vieira

Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível

123p., 210 × 297 mm (FGA/UnB Gama, Engenharia Eletrônica, 2017)

Orientador: Dr. Leonardo Aguayo

Trabalho de graduação em engenharia eletrônica
Universidade de Brasília, Campus Gama – FGA/UnB

1. Comunicação via Luz Visível
2. Redes
I. Dr. Leonardo Aguayo
II. Universidade de Brasília.
III. Faculdade UnB Gama.
IV. Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível

Referência

Coelho, K. D. A.; Vieira, P. A. D. A. (2017). Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível. Dissertação de graduação em engenharia eletrônica, Universidade de Brasília, Campus Gama, DF, 123p.

Dedicamos esse trabalho aos nossos familiares, amigos e ao nosso Professor Orientador Dr. Leonardo Aguayo.

Agradecimentos

Agradeço primeiramente a Deus pela saúde e força nos momentos de dificuldade. Aos familiares e amigos, por todo o apoio, conselhos, suporte, ensinamentos e experiências que contribuíram para essa conquista. Ao meu orientador Prof. Dr. Leonardo Aguayo pela oportunidade, ensinamentos e conselhos que tornaram possível a realização desse trabalho. A minha dupla de TCC e meu amigo na vida Paulo Afonso, pela parceria no trabalho realizado. Aos demais que de alguma forma contribuíram para minha formação, meu muito obrigado.

Kaio Diego de Araújo Coelho

Agradecimentos

Agradeço minha mãe, meu pai, meu irmão, minha namorada e meus amigos que me apoiaram nessa caminhada. Agradeço também ao meu orientador, Prof. Dr. Leonardo Aguayo pela oportunidade, apoio e conselhos, que foram essenciais para concretização desse trabalho de conclusão de curso. Agradeço ao meu amigo de TCC e da vida Kaio Diego pelo trabalho realizado.

Paulo Afonso Dutra de Alarcão Vieira

FGA/UnB – Universidade de Brasília, Campus Gama

Projeto, Implementação e Caracterização de uma Rede de Comunicação via Luz Visível

Kaio Diego de Araújo Coelho

e

Paulo Afonso Dutra de Alarcão Vieira

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica.

Aprovada por:

Prof. Dr. Leonardo Aguayo
(Orientador)

Prof. Dr. André Noll Barreto
(Examinador externo)

Prof. Me. Renato Coral Sampaio
(Examinador interno)

"Se você não pode explicar algo de forma simples, então você não entendeu muito bem o que tem a dizer."

Albert Einstein

"Quando for desenvolver algo, primeiro faça um LED piscar, depois "Domine o Mundo"."

Magno Batista Correa

Resumo

Em virtude da lotação da faixa do espectro eletromagnético destinada ao WiFi, surgiu na comunidade científica o interesse pela busca de novos meios de transmissão sem fio. Com a invenção das fibras ópticas, a comunicação óptica ganhou seu primeiro impulso como uma alternativa para contornar o problema, porém, em alguns casos, realizar o cabeamento não é tão simples. Uma solução possível foi o chamado *Free Space Optics* (FSO), que utiliza o ar como canal de transmissão. Nesse cenário, o presente projeto traz a elaboração de uma rede digital de comunicação óptica para transmissão de dados, através do uso de microcontroladores e da aplicação dos fundamentos do *Visible Light Communication* (VLC), que tem relação com o conceito de FSO. A elaboração da rede foi feita seguindo três etapas: Projeto, Implementação e Caracterização. A etapa de projeto compreendeu todo o estudo das características de uma rede e a definição dos parâmetros do protótipo. Uma vez determinado o projeto, iniciou-se a fase de implementação, onde todas as ideias foram de fato produzidas. Por último, através do protótipo desenvolvido, realizou-se a caracterização dessa rede com intuito de estabelecer um cenário de aplicação para essa tecnologia. Como resultado, observou-se que tendo por base a lâmpada de LED, é possível a criação de uma alternativa eficiente para a transmissão de dados em ambientes fechados.

Palavras-chave: Comunicação óptica, VLC, Rede de transmissão, Microcontrolador.

Abstract

Due to the capacity of the band of the electromagnetic spectrum destined to WiFi, the interest in the search for new means of wireless transmission has arisen in the scientific community. With the invention of fiber optics, optical communication has gained its first thrust as an alternative to bypass the problem, but in some cases, performing the cabling is not so simple. One possible solution was called *Free Space Optics* (FSO), which uses air as the transmission channel. In this scenario, the present project brings the elaboration of a digital network of optical communication for data transmission, through the use of microcontrollers and the application of the fundamentals of *Visible Light Communication* (VLC), which has relation with the concept of FSO . The elaboration of the network was done following three stages: Design, Implementation and Characterization. The design stage comprised the entire study of the characteristics of a network and the definition of prototype parameters. Once the project was determined, the implementation phase began, where all ideas were actually produced. Finally, through the developed prototype, the characterization of this network was carried out in order to establish an application scenario for this technology. As a result, it has been observed that based on the LED lamp, it is possible to create an efficient alternative for the data transmission indoors.

Keywords: Optical communication, VLC, Transmission network, Microcontroller.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Símbolos

G	Giga (10^9)
K	Kilo (10^3)
Ω	Ohm
β	Ganho do Transistor
bps	Bits por segundo
dB	Decibéis
m	Mili (10^{-3})

Lista de Abreviaturas

ADC	*Analog to Digital Converter*
ASCII	*American Standard Code for Information Interchange*
BER	*Bit Error Rate*
CAD	*Computer-Aided Design*
CI	Circuito Integrado
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
FIFO	*First In, First Out*
FSO	*Free Space Optics*
GND	*Ground*
ID	Identificação
IEEE	*Institute of Electrical and Electronics Engineers*
LED	*Light Emitting Diode*
LOS	*Line of Sight*
MCD	miliCandela
MDF	*Medium Density Fiberboard*
NLOS	*Non Line of Sight*
PCB	*Printed Circuit Board*
RF	Radiofrequência
RSR	Relação Sinal Ruído
SRAM	*Static Random Access Memory*
TBJ	Transistor de Junção Bipolar
TED	*Technology, Entertainment, Design*
USB	*Universal Serial Bus*
UVLC	*Underwater Visible Light Communication*
VLC	*Visible Light Communication*

1 Introdução

A quantidade de dispositivos móveis ao redor do mundo aumentou consideravelmente nos últimos anos onde, *Smartphones*, *Tablets* e *Notebooks* estão cada vez mais comuns no dia a dia das pessoas (Vieira et al., 2017). Consequentemente, houve também um grande aumento na procura por serviços de comunicação sem fio.

Entretanto, há alguns anos, nas mídias internacionais, utilizaram-se diversas vezes o termo "*WiFi Spectrum Crunch*" (De Vries et al., 2014), que refere-se à lotação da faixa do espectro eletromagnético destinada ao WiFi. Essa repetição ocasionou indagações no setor científico, que motivaram as buscas por tecnologias alternativas com objetivo de evitar essa lotação.

Em questão de velocidade, a tecnologia óptica se mostra como uma boa alternativa, pois as conexões podem atingir valores superiores a $10Gbps$, trafegando livre em fibras ópticas (Aleixo et al., 2013). Entretanto, realizar o cabeamento nem sempre é uma tarefa fácil e, dessa forma, acaba tornando-se uma limitação dessa tecnologia.

Nesse contexto, o *Free Space Optics* (FSO) se encaixa como uma forma de adaptação capaz de contornar o problema do cabeamento, uma vez que incorpora em seu conceito a mobilidade característica de um sistema sem fio combinada com a tecnologia das comunicações ópticas (Aleixo et al., 2013), servindo como auxiliar para os modelos de transmissão existentes. Dessa forma, também é visto como uma alternativa econômica em telecomunicações, dentre suas vantagens, pode-se citar (Mahdy and Deogun, 2004) (Arnon, 2003):

- Oferecimento de larguras de banda elevadas, da mesma ordem dos sistemas a fibra, com custos de instalação menores que estes por não necessitar de grandes obras de engenharia civil;
- Não necessita de licenças para operação como as licenças de uso do espectro dos sistemas sem fio, nem para a instalação como as licenças para obras em vias públicas, necessárias para a colocação de fibras;
- Rapidez de operação, com possibilidade de instalação de enlaces temporários, usados, por exemplo, para a cobertura de grandes eventos;
- Imunidade à interferência de outros sistemas de telecomunicações ou sistemas elétricos e mesmo entre enlaces FSO, devido à alta diretividade do feixe;
- Elevada segurança, devido à dificuldade de interceptação dos sinais, sobretudo quando comparado com os sistemas sem fio em rádio frequência (RF).

1.1 Motivação

Apesar de bastante difundida, a comunicação por Radiofrequência (RF) possui limitações. Além do "*WiFi Spectrum Crunch*", existem problemas relacionados à segurança em ambientes restritos. Nesse contexto, o *Visible Light Communication* (VLC) tem destacado-se como uma alternativa para solucionar esses problemas (Pohlmann, 2010).

O VLC é visto como uma aplicação do FSO em ambientes fechados (*indoor*), pois trata-se de um sistema que utiliza a iluminação do ambiente para transmitir dados para os dispositivos, os quais são iluminados pela mesma fonte. É por esse motivo que o VLC possibilita uma segurança superior ao RF, pois tendo-se em vista que a luz não pode atravessar paredes, dispositivos externos a esse ambiente não tem acesso à iluminação da fonte e, dessa forma, não tem acesso às informações que são transmitidas pela luz, garantindo assim, uma proteção contra interceptação de dados.

Essa segurança possibilita que a tecnologia óptica seja levada a lugares onde o RF não pode ser utilizado. O Professor alemão Harald Haas, durante uma apresentação ao *Technology, Entertainment, Design* (TED) em 2011, citou algumas aplicações importantes para esse sistema (Haas, 2011):

> "Intrinsecamente ambientes seguros como esta usina de petroquímicos – Não se pode usar RF, ele pode gerar faíscas na antena, mas eu posso usar luz – vocês veem muita luz lá. Em hospitais, para novos instrumentos médicos; nas ruas, para o controle do tráfego. Carros têm faróis baseados em LED, luzes traseiras baseadas em LED, e os carros podem comunicar-se uns com os outros e prevenir acidentes na maneira como eles transmitem informações. Semáforos podem comunicar-se com o carro, e assim por diante. E então temos milhões de lâmpadas nas ruas instaladas ao redor do mundo. E cada lâmpada de rua seria um ponto de acesso livre. E então nós temos estas cabines de avião. Há centenas de luzes em uma cabine de avião, e cada uma destas luzes poderia ser um potencial transmissor de informações sem fio. Então vocês poderiam usufruir dos seus vídeos favoritos do TED nas suas longas viagens de volta para casa. Vida online. Eu penso que esta visão é possível."

No ano de 2016, o engenheiro eletrônico Vilmey Filho, em seu trabalho de conclusão de curso intitulado como "VLC - Comunicação Óptica por Luz Visível" (Romano Filho, 2016), projetou um sistema de transmissão ponto a ponto que utilizava a luz visível como meio de emissão. Nesse cenário, iniciou-se esse projeto que consiste em uma rede de transmissão de dados via luz visível, que tem como base os resultados obtidos para o sistema ponto a ponto e visa realizar uma futura integração entre o RF e o VLC para aplicações em ambientes restritos.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho consiste em projetar, implementar e caracterizar uma rede digital de transmissão de dados, que utilize a luz visível como meio de emissão.

1.2.2 Objetivos Específicos

A fim de desenvolver e implantar essa rede digital, se faz necessário estabelecer objetivos específicos de modo a estipular pequenas metas, são elas:

- Criar o protocolo de acesso à camada física;
- Criar o protocolo de transmissão;
- Criar o protocolo de recepção;
- Desenvolver os algoritmos dos dispositivos ativos;
- Desenvolver o algoritmo do roteador;
- Caracterizar a rede em termos de: taxa de transmissão, distâncias, angulações e relação sinal ruído;
- Validar o funcionamento da rede através do protótipo de teste.

2 Fundamentação Teórica

Visible Light Communication (VLC) é como são conhecidos os sistemas em que dados são enviados através da modulação das ondas de luz no espectro visível, ou seja, utiliza-se apenas a faixa do espectro eletromagnético que varia entre $390nm$ a $700nm$ (Vieira et al., 2017).

O VLC baseia-se em um sistema capaz de realizar, ao mesmo tempo, a iluminação do ambiente no qual está inserido e a transmissão de dados entre os dispositivos. É composto, essencialmente, por um transmissor responsável por modular a fonte de alimentação da iluminação, criando pulsos de luz de acordo com o dado que deseja enviar e um receptor capaz de distinguir as variações da iluminação e recuperar os dados transmitidos (Souza et al., 2013).

A velocidade de comutação do emissor de luz deve ser elevada e, por esse motivo, não é possível utilizar lâmpadas incandescentes ou fluorescentes nesse sistema. Nesse cenário, as lâmpadas de *Light Emitting Diode* (LED) se tornam as principais candidatas para compor o sistema do VLC, pois, além da sua popularização como fonte de iluminação ambiente, possui algumas vantagens importantes para o sistema, das quais podemos destacar (Ferreira, 2014):

1. O número de vezes e a frequência em que é ligado e desligado não altera sua vida útil;
2. Flexibilidade de usos, formas, tamanhos e design;
3. Acendimento imediato;
4. Possibilidade de dimerização.

2.1 Histórico

Os estudos envolvendo VLC não são um privilégio do século XXI. A luz sempre esteve presente entre os elementos utilizados pelo ser humano para se comunicar (Aleixo et al., 2013). Contudo, durante todo o período histórico das comunicações ópticas, existiram marcos fundamentais que representam a evolução das tecnologias ópticas, que hoje são utilizadas para comunicação. A Figura 2.1 mostra uma linha do tempo com esses marcos, seguido de suas respectivas descrições (Caetano, 2011).

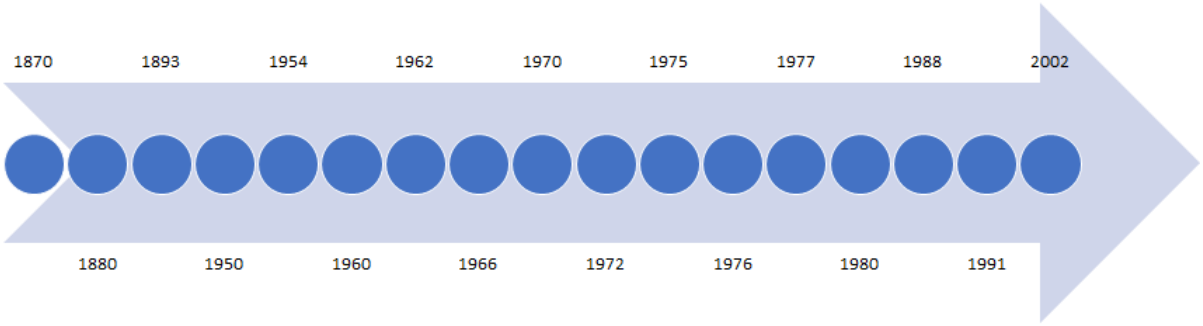


Figura 2.1. Linha do tempo com marcos históricos das comunicações ópticas.

- **1870**: O físico irlandês John Tyndall demonstrou o princípio de guiamento da luz (Tyndall, 1886);
- **1880**: Alexander Graham Bell, ao patentear o fotofone, possibilitou que ocorresse a primeira transmissão de voz, que deu-se através de luz não guiada (Hutt et al., 1993);
- **1893**: O padre Landell de Moura inventa no Brasil o telefone sem fio. Sua base foi a emissão de luz branca originada em um arco voltaico que era modulada pela voz do locutor (Fornari, 1984);
- **1950**: Inicia-se experimentos em busca de um guia de luz;
- **1954**: Surgimento do Guia Óptico Recoberto, o qual consistia em um material dielétrico com índice de refração ligeiramente menor do que o meio no qual se desejava a propagação de luz;
- **1960**: O físico Theodore Maiman criou o primeiro Laser, no *Hughes Research Laboratory* (Maiman, 1967);
- **1962**: Começaram experiências com lasers de semicondutores. Este tipo de laser foi aprimorado e hoje é utilizado nas comunicações ópticas;
- **1966**: Charles Kao e Charles Hockham, na Inglaterra, propuseram a utilização de fibra de vidro (fibras ópticas) para transmissão de luz do laser, (atenuação da ordem de $1000 dB/Km$) (Hecht, 2015);
- **1970**: A *Corning Glass Works* produziu a primeira fibra óptica com atenuação de $20 dB/Km$;
- **1972**: Fibras ópticas com atenuação de $4 dB/Km$ já eram obtidas;
- **1975**: As fibras deixam os laboratórios e entram em fase de produção industrial;

- **1976**: Foi implantado pela Western Electric em Atlanta, um link de Fibra Óptica com extensão de 2,5 km, para voz, dados e imagens, á uma taxa de 44,7 Mb/s (Megabits por segundo);

- **1977**: Foi instalado pela Bell, no centro de Chicago, utilizando um cabo multi fibras, a primeira Rede Óptica de uma Empresa de Telecomunicações. Tinha a capacidade de transportar 54 Mb/s e a distância entre os Centros Telefônicos era de 2,6 km. Neste mesmo ano foi instalado pela General Telephone and Electronics um Sistema Óptico de 6 Mb/s em Long Beach, Califórnia;

- **1980**: O Sistema Bell inaugura a primeira Rede Óptica Nacional, interligando a Capital Washington á cidade de Cambridge, no estado de Massachusetts;

- **1988**: Foi inaugurada a primeira Rede Óptica Internacional, pelo lançamento do Cabo Óptico Submarino TAT – 8, usando Laser de 1,3 micrômetros em Fibra Monomodo;

- **1991**: A NTT (*Nippon Telegraph and Telephone Corporation*) no Japão, demonstrou a transmissão de Solitons, através de um milhão de kilometros de Fibra Óptica;

- **2002**: Mais de trinta anos se passaram, desde a fabricação da primeira Fibra Óptica, e neste ano mais de 80% de todo tráfego do mundo, é escoado através de Fibras Ópticas, sendo a sua utilização cada vez maior.

2.2 VLC vs. Radiofrequência

Buscando definir se o VLC é um substituto do WiFi ou seu complementar, o pesquisador Gordon Povey (Povey, 2011) realizou um estudo comparativo, onde foram levantadas as características de ambas as tecnologias em um ambiente fechado (*indoor*). Seus resultados são apresentados na Tabela 2.1.

Tabela 2.1. Comparação entre o WiFi e o VLC em locais fechados (*indoor*) (Povey, 2011).

Especificações	WiFi	VLC
1. Maturidade	***	*
2. Velocidade	***	***
3. Alcance	**	*
4. Potência	*	***
5. Backhaul	**	**
6. Potência RX/TX	**	***
7. Segurança	**	***
8. Densidade de dados /reutilização de frequência	*	***
9. Performance NLOS	**	*
10. Tabela de materiais	**	***

Afim de justificar as classificações dadas às tecnologias, o autor, dispôs de suas considerações para cada item da Tabela 2.1, as quais são descritas a seguir.

1. A tecnologia WiFi, por ser mais antiga, é melhor fundamentada e, consequentemente, mais estudada e explorada, enquanto o VLC é uma ideia relativamente nova em processo de evolução;

2. No quesito velocidade, as tecnologias se equiparam, atingindo valores notavelmente altos da ordem de *Gbps*;

3. Por se tratar de uma tecnologia baseada em ondas eletromagnéticas, o WiFi é capaz de atravessar paredes, possuindo assim um maior alcance, diferentemente do VLC que não é capaz de atravessar barreiras físicas;

4. Dado que o VLC se utiliza da própria iluminação *indoor*, a energia gasta para emitir os dados é inferior e, com isso, tem uma maior potência quando comparada ao WiFi;

5. O *backhaul*[1] é o mesmo para ambas as tecnologias;

6. O VLC, por se tratar de uma transmissão feita a partir de uma lâmpada de LED, não necessita de antenas para realizar a transmissão dos dados;

7. Como a iluminação está confinada a ambientes fechados, o VLC não transmite dados para locais fora desse ambiente, trazendo mais segurança na transmissão de dados;

8. O WiFi causa muita interferência de co-canal e a implantação de mais nós WiFi não aumenta a capacidade de forma linear. Já no VLC, a interferência é limitada e, portanto, a eficiência da reutilização é muito maior;

[1]É a porção de uma rede hierárquica de telecomunicações responsável por fazer a ligação entre o núcleo da rede, ou *backbone*, e as sub-redes periféricas.

9. O WiFi é uma tecnologia do tipo *Non Line of Sight* (NLOS) e, portanto, não necessita estar na linha de visada do equipamento para realizar a comunicação de boa performance, enquanto que, o VLC, por ser do tipo *Line of Sight* (LOS), necessita estar na linha de visão para que seja realizada a comunicação entre os dispositivos;

10. De acordo com o item 6, o VLC torna-se mais barato, pois se utiliza de equipamentos já existentes no local sem a necessidade de incorporar novos periféricos.

Por fim, a partir de sua análise, concluiu:

> "Em reflexão, vou sugerir que o VLC seja complementar do WiFi. O WiFi pode ser usado para cobertura de área ampla dentro de um edifício. Muitos nós VLC podem ser usados para *links* de taxa de dados de curto alcance. Se um *link* não estiver disponível, por exemplo devido a sombreamento, o WiFi pode fornecer o *backup*."

2.3 Aplicações

Pesquisas demonstram que o conceito do VLC está aplicado nas mais variadas áreas, sendo algumas delas: espaço óptico livre, telecomunicação com sistemas móveis, sinalização de trânsito, sistemas de posicionamento, comunicação entre veículos, comunicação estrada-veículo utilizando câmera de alta velocidade, comunicação por sinalização RGB, etc. Alguns exemplos são citados a seguir.

Serviços de localização global

Nessa aplicação, um servidor conectado à Internet contém as coordenadas de cada luminária, semáforo ou qualquer outra fonte emissora de luz que transmita sua posição usando VLC. Cada um desses objetos possui uma Identificação (ID) única, que permite conhecer a localização de uma pessoa ou objeto com precisão de centímetros. Essa informação de localização é continuamente atualizada, enviada através do sistema VLC e mantida armazenada, permitindo o rastreio do objeto (Haruyama, 2011), como mostra a Figura 2.2.

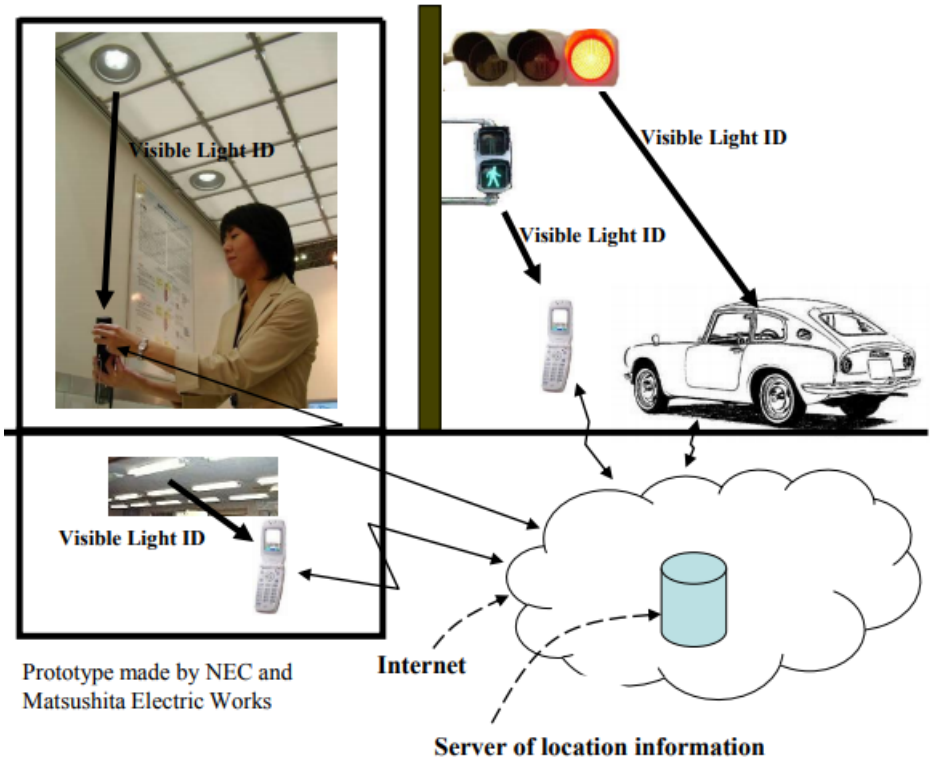


Figura 2.2. Exemplo de aplicação com sistema ID (Haruyama, 2011).

Sistemas de alcance ultra pequenos

Diante da grande complexidade dos circuitos e da necessidade cada vez maior de altas taxas de transmissões em barramentos, cientistas buscam fazer comunicação de barramentos via transmissores ópticos, o que pode reduzir a complexidade das vias de cobre e aumentar a eficiência na troca de dados.

Essa tecnologia trabalha na ordem dos milímetros, e esse tipo de sistema geralmente é encontrada no nível de chip, que é utilizado para comunicar circuitos (Figura 2.3), ou fazer comunicação dentro de um mesmo chip.

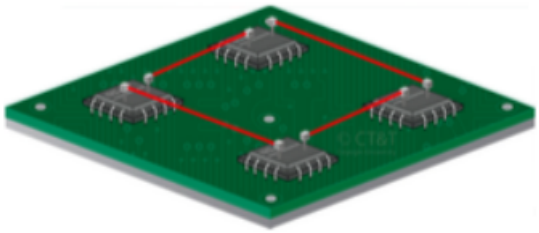


Figura 2.3. Ilustração de um sistema de alcance ultra pequeno (Romano Filho, 2016).

Com o uso desse meio de transmissão, conexões baseadas em cobre poderão ser substituídas. Esse processo oferece benefícios na redução da latência dos dispositivos, ou seja, o tempo de resposta do circuito será reduzido.

Sistemas nesse nível podem ser guiados ou não. Arquiteturas guiadas promovem um link direto de luz entre o transmissor e o receptor, confinando a luz em uma guia de onda. Enquanto nos sistemas não guiados o feixe de luz é exposto ao ambiente, o que pode causar ruídos e interferências na comunicação (Kachris et al., 2012).

Comunicação embaixo d'água

Em outubro de 2010, na Universidade de Keio, os laboratórios Nakagawa e a empresa Rise Underwater demonstraram um transmissor VLC em forma de lanterna capaz de transmitir áudio a $30m$ de distância sob a água, possibilitando, assim, a comunicação por voz entre os mergulhadores. A luz produzida pelos LEDs da lanterna é modulada por intensidade. Um fotodetetor fica ao lado do LED e funciona como receptor. Uma imagem do equipamento é mostrada na Figura 2.4. Essa transmissão de dados sob a água é chamada pelos laboratórios Nakagawa de *Underwater Visible Light Communication* (UVLC) (Junior, 2011).

Figura 2.4. Equipamento desenvolvido (Haruyama, 2011).

3 Projeto da rede

3.1 Configuração da rede

Nessa seção, serão descritas de forma detalhada as configurações do sistema proposto de acordo com algumas características fundamentais, as quais são apresentadas a seguir.

3.1.1 Validação do sistema

Antes de iniciar a discussão sobre as características do sistema, é necessário estabelecer a estratégia de validação para que todas as configurações escolhidas a seguir sejam coerentes com a validação que deve ser realizada.

Desta forma, a validação deve ocorrer por meio da transmissão de algum tipo de dado, seja ele vídeo, imagem ou texto. Por uma questão prática, nesse projeto será realizada a validação por meio da transmissão de mensagens de texto utilizando como base a tabela *American Standard Code for Information Interchange* (ASCII).

Tabela ASCII

ASCII é uma codificação de caracteres de oito bits baseada no alfabeto inglês. Os códigos ASCII representam texto em computadores, equipamentos de comunicação, entre outros dispositivos que trabalham com texto. Desenvolvida a partir de 1960, grande parte das codificações de caracteres modernas a herdaram como base.

A codificação define 256 caracteres, preenchendo completamente os oito bits disponíveis. Desses, 33 não são imprimíveis, tratam-se de caracteres de controle atualmente não utilizáveis para edição de texto, porém amplamente utilizados em dispositivos de comunicação, que afetam o processamento do texto. Exceto pelo caractere de espaço, o restante é composto por caracteres imprimíveis (Faria, 2011).

3.1.2 Codificação de linha

A codificação de linha ou codificação digital de dados são termos empregados para modulação digital em banda base. Muitas redes de computadores são para aquisição e armazenamento de dados ou são simplesmente redes locais de computadores. Para tais redes, a codificação em banda base é suficiente para transmissão, não havendo necessidade de modular os dados em uma frequência mais alta (Machado, 2013).

Em geral, a codificação de linha é responsável por transformar mensagem binária em pulsos elétricos (formas de onda), para realizar a transmissão no canal (Lathi and Ding, 2012). Assim, para o presente projeto, foi escolhida a codificação *Manchester* para evitar

que ocorra uma longa sequência de zeros, que tornaria visível a transmissão. A seguir é descrito como esse protocolo é implementado.

Codificação *Manchester*

A codificação *Manchester*, também conhecida como bifásica de nível, é a técnica especificada pelo IEEE-802 (Backers, 1988) para uma rede padrão *Ethernet*, onde cada período do bit é dividido em metades complementares (Soares et al., 1995). Assim, uma transição de tensão de negativa para positiva no meio do bit indica um número binário "1", enquanto uma transição de positiva para negativa representa um "0", ou seja, os bits 1 geram uma borda de subida no meio do tempo de bit e os bits 0 uma borda de descida no meio do tempo de bit. A Figura 3.1 exemplifica essa codificação para uma dada sequência de bits.

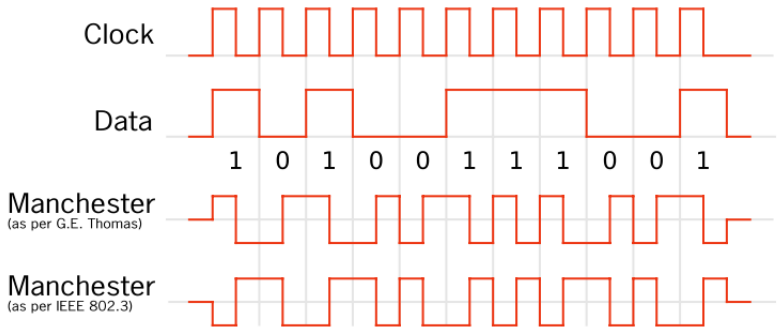


Figura 3.1. Exemplo de codificação *Manchester* (Weller, 2009).

3.1.3 *Design* do sistema

O *design* do sistema define a forma como os dispositivos devem estar dispostos para que seja possível obter êxito nas transmissões. Contudo, é necessário avaliar as escolhas realizadas quanto a eficiência, o que foi feito por meio de uma busca de padrões já utilizados em sistemas semelhantes. As escolhas realizadas são apresentadas a seguir.

Topologia de rede

A topologia de redes descreve como é o *layout* da rede de computadores através da qual há o tráfego de informações e também como os dispositivos estão conectados a ela. Há várias formas de se organizar a interligação entre cada um dos nós (computadores) da rede. Topologias podem ser descritas fisicamente e logicamente, onde a topologia física é a verdadeira aparência (ou *layout*) da rede, enquanto que a lógica descreve o fluxo dos dados através da mesma (Soares et al., 1995).

No presente projeto, será utilizada a topologia estrela, apresentada na Figura 3.2, onde cada nó é interligado a um nó central (mestre), através do qual todas as mensagens devem passar. Assim, tal nó age como centro de controle da rede, interligando os demais nós (escravos). Nada impede que haja comunicações simultâneas, desde que as estações envolvidas sejam diferentes (Soares et al., 1995).

Figura 3.2. Topologia em estrela (Soares et al., 1995).

Regra de acesso ao canal

Buscando utilizar de forma correta a topologia escolhida, se faz necessário estabelecer uma regra de acesso ao canal de transmissão. No presente projeto, é necessário que existam, no mínimo, três nós (escravos) para que as características de uma rede possam ser observadas.

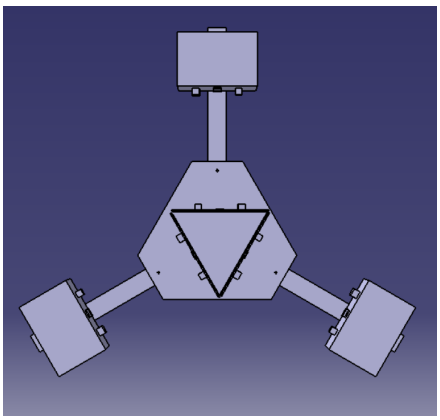
Assim, estabeleceu-se que a regra de acesso é feita de forma que o primeiro nó que solicitar o acesso, desde que o canal não esteja ocupado, terá a permissão para transmitir os dados desejados. Esse modelo é semelhante ao ALOHA (Dantas, 2010), se houver colisões, a transmissão não será devidamente realizada.

3.2 Protótipo

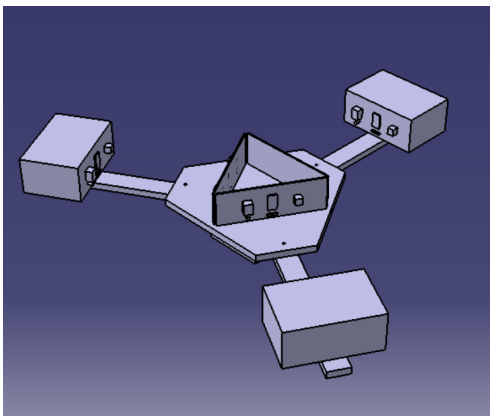
Para a implementação dos algoritmos propostos, desenvolveu-se um protótipo a fim de dispor todo o *hardware* necessário para o funcionamento da rede. O mesmo foi projetado no *software* CATIA® V5R19, atendendo aos seguintes requisitos:

- *Layout* em topologia estrela;
- Hastes com fixação móvel;
- Possibilitar testes de angulação e distância.

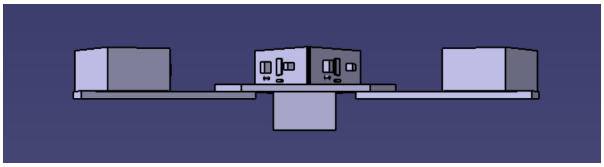
Na Figura 3.3 temos o *Layout* do protótipo.



(a) Vista superior.



(b) Vista isométrica.



(c) Vista frontal.

Figura 3.3. *Computer-Aided Design* (CAD) do *layout* em estrela do protótipo, onde: (a) Vista superior; (b) Vista isométrica; e (c) Vista frontal.

3.2.1 Microcontrolador

O microcontrolador escolhido para a implementação dos algoritmos desenvolvidos foi o Atmega328, o qual apresenta a maioria das características da família AVR[1], e uma memória flash maior se comparada a outros microcontroladores dessa família, mantendo o mesmo número de pinos (Lima, 2012). Por ser utilizado na plataforma *Arduíno*, torna-se de fácil aquisição e por isso foi escolhido para o projeto da rede. Na Tabela 3.1 temos as características técnicas do microcontrolador Atmega328.

Tabela 3.1. Características técnicas do Atmega328 (Lima, 2012).

Microcontrolador	Atmega328
Voltagem Operacional	$5V$
Pinos de I/O Digital	14 (dos quais 6 podem ser saídas PWM)
Pinos Analógicos	6
Corrente CC por pino I/O	$40mA$
Corrente CC pino de 3v3	$50mA$
Flash Memory	$32Kb$, sendo 0,5 utilizada no bootloader
SRAM	$2Kb$
EEPROM	$1Kb$
Clock	$16Mhz$

[1]A empresa diz que o nome AVR não é um acrônimo e não tem nenhum significado em especial. Os criadores nunca deram uma resposta definitiva sobre o assunto.

3.2.2 Transceptor Bidirecional

Essencialmente o transceptor bidirecional é composto por um transmissor e um receptor. Para o projeto desenvolvido, o transceptor deve possuir características ópticas, isto é, a transmissão e recepção devem operar na faixa da luz visível. A seguir, são apresentadas detalhadamente, as partes que compõem esse transceptor.

Transmissor Óptico

Para a montagem do *hardware* do transmissor óptico é utilizada a tecnologia LED para emissão dos dados. Na Tabela 3.2 são apresentadas informações sobre as características técnicas do LED de alto brilho na cor branca, viabilizando assim, sua utilização no projeto do módulo transceptor.

Tabela 3.2. Características do LED de alto brilho

Características Técnicas	Valores
Cor	Branca
Intensidade Luminosa	10.000 MCD
Tensão de Polarização Direta (V_f)	$3V$
Corrente Direta(I_f)	$20mA$

De acordo com as características técnicas apresentadas na Tabela 3.2, foi projetado um arranjo de 5 (cinco) LEDs em paralelo e assim definidos os parâmetros elétricos do emissor, presentes na Tabela 3.3.

Tabela 3.3. Características do emissor

Propriedades	Valores
Tensão Funcionamento	$9V$
Corrente	$100mA$
Potência	$0.9W$

Para que o microcontrolador escolhido seja capaz de enviar a mensagem utilizando esse arranjo, se faz necessário o uso de um *driver* de potência, pois como observado na Tabela 3.1, temos que a corrente máxima por pino do Atmega328 é de $40mA$, ou seja, muito inferior que a do arranjo de LEDs da Tabela 3.3.

O *driver* de potência é muito utilizado para acionamentos de cargas que precisam de uma elevada corrente de funcionamento. Os diversos tipos de transistores existentes no mercado são utilizados na construção desses circuitos de potência. Na Figura 3.4, temos o Transistor de Junção Bipolar (TBJ), que é comumente modelado como uma fonte de corrente controlada por corrente para ser utilizado no circuito *driver* (Lima, 2012).

Analisando a Figura 3.4 para o dimensionamento correto do *driver* de potência, as variáveis V_e (Tensão de saída do microcontrolador), V_{cc} (Tensão de alimentação da carga),

I_c (Corrente de coletor) e o ganho do transistor NPN, são dados fixos do circuito, restando calcular os valores dos resistores da base R_b e do coletor R_c, utilizando a Tabela 3.2 (Lima, 2012).

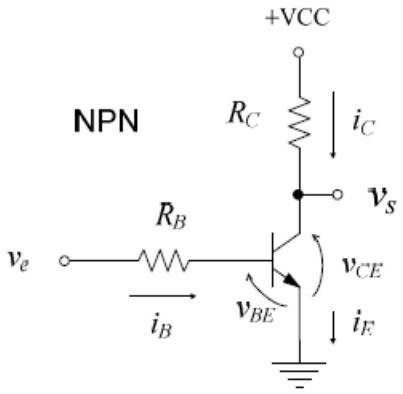


Figura 3.4. Transistor TBJ-NPN

Dessa forma, o dimensionamento do *driver* foi realizado de acordo com os seguintes passos:

- Primeiro, a corrente de carga I_c foi definida como $100mA$ de acordo com a Tabela 3.3;
- Com a corrente I_c definida, optou-se pelo transistor BC337-25, pois o mesmo suporta uma corrente de coletor que oferece uma margem de folga em relação ao BC548. A Tabela 3.4 mostra a comparação realizada entre os transistores;
- O próximo passo é identificar o fator de ganho (β) do transistor escolhido. A Tabela 3.4 mostra o ganho mínimo desse transistor;
- Com esses dados podemos então, calcular o valor do resistor da base utilizando a Equação 3.1.

$$R_b = \frac{(V_{cc} - V_{be}) \times \beta}{C_s \times I_c}, \tag{3.1}$$

  onde R_b é o resistor da base, V_{cc} é a tensão do coletor, V_{be} é a tensão base-emissor em saturação, β é o ganho do transistor, I_c é a corrente de coletor e C_s coeficiente de segurança para garantir a saturação.

- Substituindo $V_e = 5V$, $V_{be} = 0.7V$, $\beta = 160$, $I_c = 100mA$ e $C_s = 2$ na Equação 3.1 obtemos:

$$R_b = \frac{(5 - 0.7) \times 160}{2 \times 100 \times 10^{-3}} = 3440\Omega \tag{3.2}$$

- Para o cálculo do resistor R_c utilizou-se a Equação 3.3;

$$R_c = \frac{(V_{cc} - Vf)}{I_f} \tag{3.3}$$

- Substituindo $V_{cc} = 9V$, $V_f = 3V$, $I_f = 20mA$ na Equação 3.3 obtemos:

$$R_c = \frac{9 - 3}{20 \times 10^{-3}} = 300\Omega \tag{3.4}$$

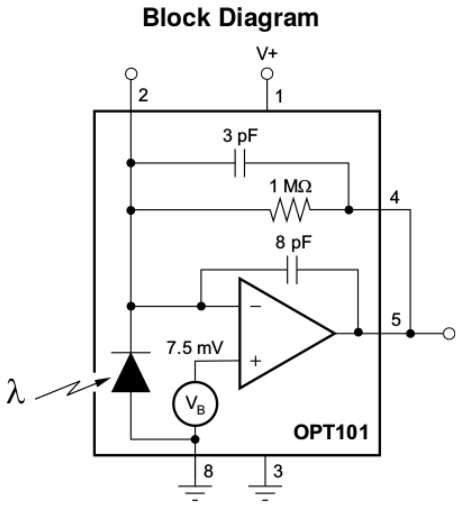
Tabela 3.4. Comparação dos transistores

	BC548	BC337-25	TIP120
V_{ceo}	30[V]	45[V]	60[V]
V_{be}	0.7[V]	1.2[V]	2.5[V]
I_c Max	0.1[A]	0.8[A]	5[A]
β	100	160	1000

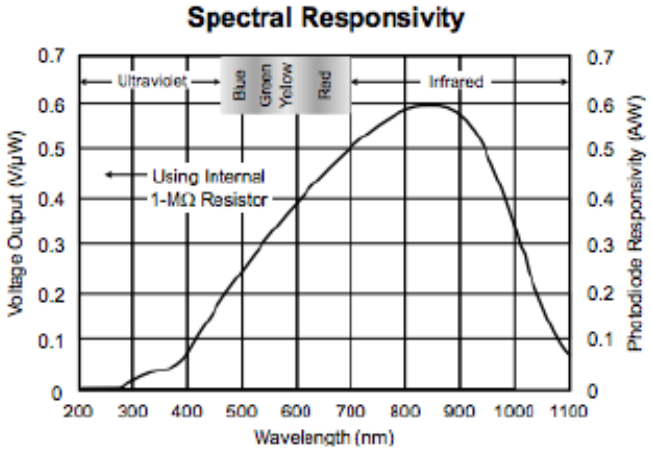
Receptor Óptico

O receptor óptico é responsável por captar a luz emitida e converter o sinal luminoso em elétrico. Para a captação do sinal, foi escolhido um receptor de transimpedância, Figura 3.5, que consiste em um dispositivo óptico que combina um fotodiodo com um amplificador no mesmo Circuito Integrado (CI).

(a) Fotodiodo OPT101 com amplificador de transimpedância.



(b) Circuito interno do CI OPT101.



(c) Sensibilidade do fotodiodo em relação ao espectro de luz.

Figura 3.5. Receptor óptico escolhido , onde: (a) apresenta o encapsulamento; (b) o seu respectivo circuito interno; e (c) o seu espectro de luz (Datasheet, 2003).

Analisando as características do dispositivo, é possível observar por meio da Figura 3.5(c), que o mesmo apresenta alta sensibilidade ao espectro de luz infravermelha e média sensibilidade ao espectro de luz visível.

3.2.3 Montagem do protótipo

A montagem do protótipo foi realizada em 5 etapas, são elas:

Etapa 1 - Simulação e desenho do *layout* do transceptor bidirecional

A montagem do protótipo iniciou-se com a simulação do *driver* de potência utilizando os resultados encontrados na Seção 3.2.2. Na Figura 3.6, observa-se que a corrente encontrada ($97.1mA$) é próxima do valor definido na Tabela 3.3.

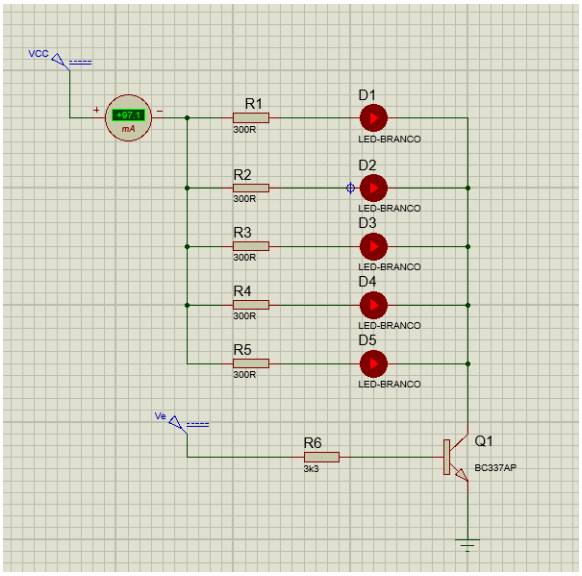
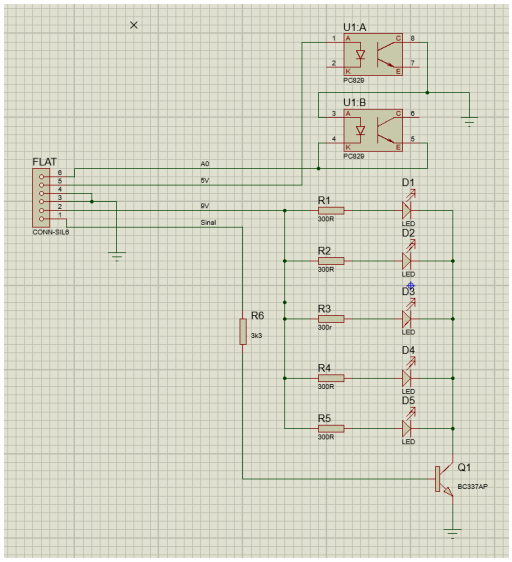
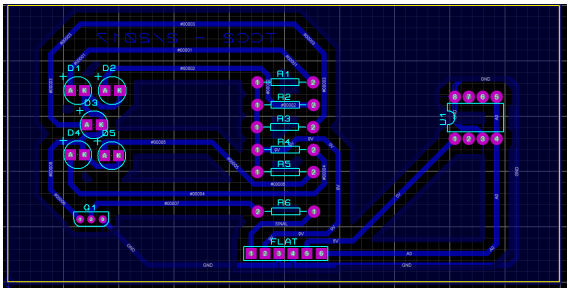


Figura 3.6. Simulação no *Software* Proteus Design Suite do circuito *driver* de potência

Após a validação do circuito de *driver* de potência, o transceptor bidirecional foi projetado no mesmo *software* de simulação, onde temos o esquemático, Figura 3.7(a), e o *Layout*, Figura 3.7(b), da *Printed Circuit Board* (PCB).



(a) Esquemático do Transceptor Bidirecional.



(b) *Layout* do Transceptor Bidirecional.

Figura 3.7. Projeto da PCB: (a) apresenta o esquemático da PCB desenvolvida; e (b) o seu respectivo *layout*;

Etapa 2 - Confecção das placas

A placa projetada na etapa anterior foi confeccionada e o resultado final pode ser visto na Figura 3.8.

Figura 3.8. Placa final do Transceptor Bidirecional

Etapa 3 - Corte do MDF em escala real e pintura

Para construção do protótipo foi utilizada madeira de *Medium Density Fiberboard* (MDF) seguindo o CAD projetado na Seção 3.2. O mesmo foi pintado com tinta preta para acabamento final.

Etapa 4 - Ligações elétricas

As ligações elétricas realizadas para as placas 1, 2 e 3 são representadas na Figura 3.9. A placa central apresenta algumas diferenças em suas ligações relativas aos pinos de *driver* e *Analog to Digital Converter* (ADC), como descrito na Tabela 3.5.

Como observado na Figura 3.9, os pinos $9V$ e *Ground* (GND) do *driver*, que não estão ligados na placa *Arduíno*, são ligados no regulador de tensão DC-DC LM2596 (ligado em uma fonte de $12V/1A$), que é responsável por fornecer a tensão estabilizada para todas as placas do protótipo. A alimentação do *Arduíno* é feita por meio de cabo *Universal Serial Bus* (USB) ligado ao computador.

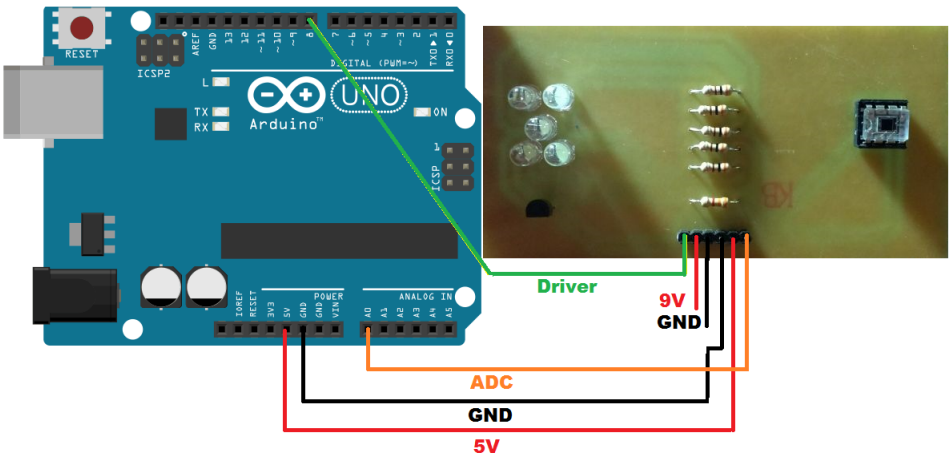


Figura 3.9. Ligações elétricas das placas 1, 2 e 3

Tabela 3.5. Pinagem

	Placa 1	Placa 2	Placa 3	Placa Central
Pino *Driver*	8	8	8	5,6,7
Pino ADC	A0	A0	A0	A0,A1,A2

Etapa 5 - Testes elétricos

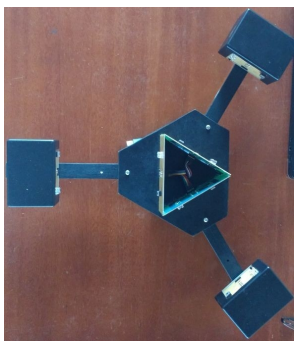
Os testes elétricos foram realizados para a verificação do funcionamento adequado de todo o protótipo e também para comparar os valores projetados na Seção 3.2.2 com os valores reais. Utilizando um multímetro Minipa ET-1002 para medir os parâmetros de uma placa do transceptor bidirecional, obtivemos resultados bem satisfatórios quando comparado com os valores encontrados na etapa de projeto. Esses são apresentados na Tabela 3.6.

Tabela 3.6. Resultados dos testes de medição.

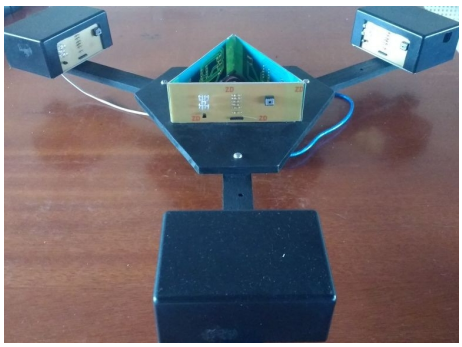
Parâmetro	Resultados
Alimentação	$9,05V$
Corrente total	$90,5mA$
Tensão no LED	$2,97V$
Corrente no LED	$18,64mA$

Etapa 6 - Resultados

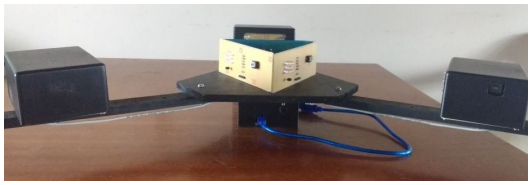
Por fim, na Figura 3.10, temos o resultado da montagem final de todo o *hardware* no protótipo.



(a) Vista superior.



(b) Vista isométrica.



(c) Vista frontal.

Figura 3.10. Protótipo finalizado, onde: (a) Vista superior; (b) Vista isométrica; e (c) Vista frontal.

3.3 Protocolos

Protocolo é definido como um conjunto de regras que governam a conversa entre duas ou mais partes (Tannenbaum and Zucchi, 2009). Em outras palavras, um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento (Kurose et al., 2007).

Dessa forma, se faz necessário a criação dos protocolos fundamentais para o correto funcionamento da rede. O detalhamento presente em cada um dos protocolos criados, bem como suas funções, são descritos nas seções 3.3.1, 3.3.2 e 3.3.3.

3.3.1 Protocolo de mensagem

As mensagens, assim como as transmissões e os pedidos de acesso, necessitam seguir um protocolo bem definido para o correto funcionamento do sistema. Criou-se então, o protocolo apresentado na Tabela 3.7, onde adotamos como tamanho limite máximo de entrada 10 caracteres. Entretanto, dos 10 caracteres coletados somente 8 referem-se a mensagem a ser enviada, os quais posteriormente serão convertidos em ASCII. Os dois caracteres restantes referem-se ao endereço de destino da mensagem (1 caractere), e ao separador entre endereço e mensagem (1 caractere) que pode ser qualquer um, visto que será descartado. Esse protocolo, tem como função principal estabelecer o padrão de entrada de dados, que deverá ser respeitado pelo usuário para que a transmissão seja bem sucedida.

Tabela 3.7. Protocolo de mensagem.

Endereço de destino 1 byte	Separador 1 byte	Mensagem a ser enviada 8 bytes

O presente projeto possui apenas 3 destinos possíveis e, dessa forma, apesar de obter um valor de 8 bits da entrada de dados, utilizaram-se apenas 2 bits para endereça-los, sendo que o endereço “00” não representa nenhum dispositivo. O campo de mensagem possui 8 bytes de espaço e, dessa forma, o tamanho máximo de uma palavra deve ser de 8 caracteres de 8 bits cada (modelo ASCII). Se, por exemplo, o usuário 1 (“01”) deseja enviar uma mensagem para o usuário 3 (“11”), sua mensagem, de acordo com a Tabela 3.7, deve ser escrita da seguinte maneira:

$$\textit{Entrada de dados} = \textit{3-Olá},$$

onde o “3” sera convertido em (“11”), “-” será descartado e “Olá” será convertido em ASCII e enviado.

3.3.2 Protocolo de acesso

Responsável por controlar toda tentativa de acesso ao canal de transmissão, o protocolo de acesso é composto por duas tarefas, chamadas de requisição e concessão, as quais são descritas a seguir.

Requisição

A requisição tem como função solicitar acesso ao canal de transmissão. O protocolo de requisição desenvolvido é apresentado na Tabela 3.8.

Tabela 3.8. Protocolo de requisição de acesso ao meio.

Sincronismo	Origem	Destino	Tamanho
4 bits	2 bits	2 bits	8 bits

Nesse protocolo, são utilizados 4 bits para realizar o sincronismo entre dispositivos, seguidos de 2 bits que representam o endereço de origem, 2 bits contendo o endereço de destino e 8 bits contendo a quantidade de pacotes que deseja-se enviar pelo canal. Os bits de sincronismo foram escolhidos de forma que o receptor seja acionado em uma transição de nível. Considerando a utilização do código *Manchester*, a sequência escolhida para o sincronismo é "1010", pois a transmissão do primeiro bit "1" causa imediatamente uma transição de nível lógico que deve ser percebida pelo receptor para realizar, assim, o sincronismo entre dispositivos. Os bits de endereço são fundamentais para que o roteador seja capaz de definir o tráfego dos dados de maneira correta. Por último, os bits responsáveis por indicar o tamanho da mensagem são obtidos por meio do protocolo de mensagem que informa ao sistema quantos bytes foram lidos na porta de I/O.

Concessão

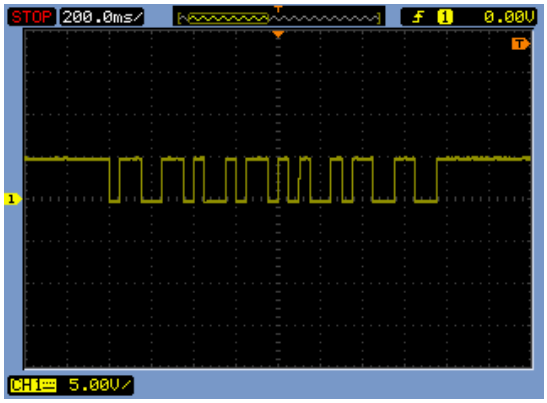
A concessão tem como função apresentar uma resposta ao pedido de requisição do usuário. O protocolo de concessão é apresentado na Tabela 3.9. Esse protocolo possui os mesmos 4 bits de sincronismo que são padrão para todo o sistema, além disso, conta com 1 bit de resposta, que, em caso de resposta afirmativa, é colocado em nível lógico alto ("1").

Tabela 3.9. Protocolo de concessão de acesso ao meio.

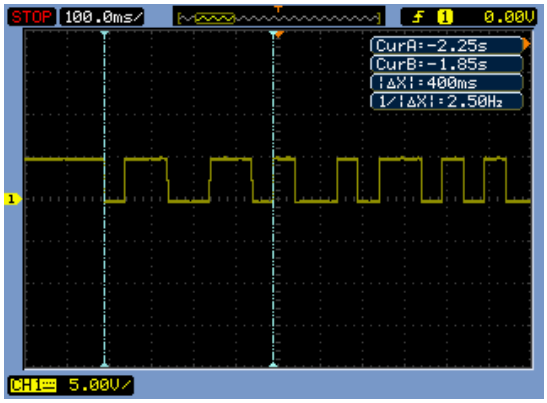
Sincronismo	Resposta
4 bits	1 bit

Para exemplificar o protocolo desenvolvido, foi realizada uma transmissão de um pacote de requisição que solicita acesso ao meio para envio de seis caracteres. A forma

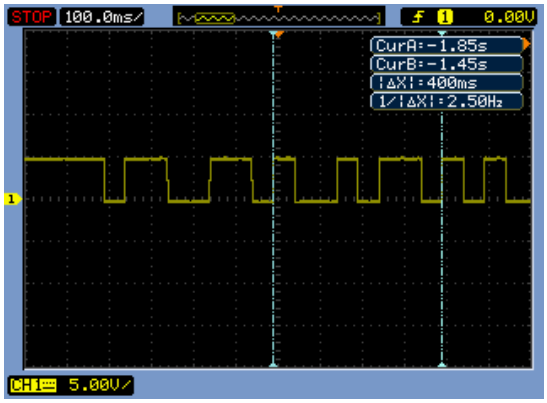
de onda gerada pelo sistema é apresentado na Figura 3.11. É válido ressaltar que essa forma de onda gerada possui codificação *Manchester*.



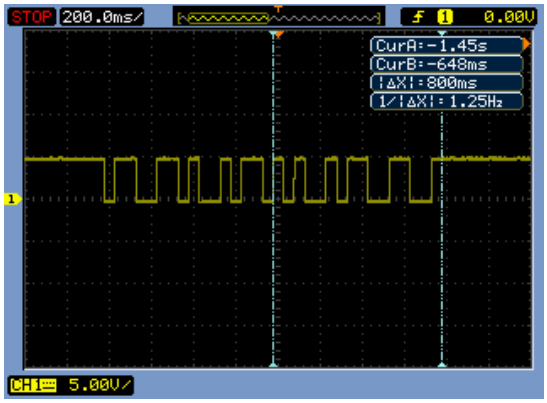
(a) Protocolo de requisição de acesso.



(b) Bits de sincronismo.



(c) Endereços de origem e destino.



(d) ASCII do número 6.

Figura 3.11. Exemplo do protocolo de requisição, onde: (a) representa todos os elementos descritos na Tabela 3.8; (b) mostra a forma de onda correspondente aos bits de sincronismo ("1010"); (c) apresenta os endereços de origem ("01") e destino ("10"); e, (d) exemplifica a forma de onda gerada para a representação binária do número 6 em ASCII ("00110110").

3.3.3 Protocolo de transmissão

O protocolo de transmissão é o responsável por enviar as mensagens do usuário ao destino desejado. Dessa forma, esse protocolo é composto por 4 bits de sincronismo que possuem a mesma sequência apresentada anteriormente ("1010"), 2 bits que representam a origem da mensagem, 2 bits que informam o destino da mensagem e 8 bits de mensagem, que representam um dos caracteres coletados pelo protocolo de mensagem. A Tabela 3.10 apresenta o pacote de transmissão criado para o sistema em questão.

Tabela 3.10. Protocolo de transmissão.

Sincronismo 4 bits	Origem 2 bits	Destino 2 bits	Informação 8 bits

A partir da análise desse protocolo, nota-se que, para transmitir os 8 bytes coletados no protocolo de mensagem, são necessárias 8 transmissões consecutivas desses pacotes, sendo que em cada pacote só é possível armazenar um caractere por vez. É válido ressaltar que os bits de origem e destino, nesse protocolo, servem como redundâncias necessárias ao sistema, pois o campo "origem" permite ao destino identificar quem enviou a mensagem e o campo "destino" serve para validar o caminho da mensagem, garantindo, assim, que a mensagem chegou corretamente ao destino desejado.

Nesse cenário, por exemplo, se o usuário 1 (endereço "01") deseja enviar mensagens para o usuário 2 (endereço "10"), os pacotes enviados conterão, respectivamente, "01" e "10" nos campos "origem" e "destino". Dessa forma, o usuário 2 sabe quem lhe enviou mensagem (usuário 1), interpretando o campo "origem", e verifica se a mensagem recebida é de fato para ele (usuário 2), verificando o campo "destino".

3.4 Algoritmos

Os algoritmos desenvolvidos para o sistema em questão dividem-se em dois grandes grupos que são separados pelas funções que desempenham na rede. O primeiro grupo, chamado de "Dispositivos Ativos", compreende os nós da topologia em estrela (Seção 3.1.3). O segundo grupo, chamado de "Roteador", representa o nó central da topologia em estrela.

As funções de cada grupo são listadas a seguir:

- Dispositivos Ativos
  - Coletar dados externos;
  - Requisitar acesso ao canal de transmissão;
  - Realizar transmissão dos dados coletados;
  - Receber dados transmitidos por outro dispositivo na rede.

- Roteador
  - Identificar pedido de acesso;
  - Solicitar permissão para transmissão ao destino;
  - Enviar resposta ao solicitante;
  - Realizar a retransmissão dos dados recebidos da origem.

Com as funções bem definidas e com os protocolos do sistemas estabelecidos na Seção 3.3, é possível implementar os algoritmos responsáveis por garantir o correto tráfego dos dados pela rede de acordo com o *hardware* escolhido na Seção 3.2.1. A fim de facilitar o entendimento dos algoritmos desenvolvidos, nas seções a seguir são dispostos os seus respectivos fluxogramas.

3.4.1 Dispositivos Ativos

O funcionamento dos dispositivos ativos é dado, essencialmente, pela execução de três rotinas, são elas: rotina geral, rotina de transmissão e rotina de recepção. A seguir, são descritas em detalhes as funções de cada rotina.

Rotina Geral

A rotina geral dos dispositivos ativos é composta por dois processos que são responsáveis por desempenhar tarefas distintas, os quais são descritos a seguir:

- O primeiro processo, representado pela Figura 3.12(a), é chamado de "modo passivo" e funciona da seguinte maneira:

  - Realiza a leitura do ADC, buscando transições de nível lógico que podem vir a indicar o início de uma transmissão;

  - Se houve transição, inicia-se a coleta de bits para verificar se os mesmo são bits de sincronismo;

  - Em caso afirmativo, o processo inicia a rotina de recepção;

  - Em caso negativo, retorna para o inicio do processo e permanece em *loop* até que haja dados para receber ou aconteça uma interrupção.

- O segundo processo, representado pela Figura 3.12(b), é chamado de "SerialEvent" e funciona da seguinte maneira:

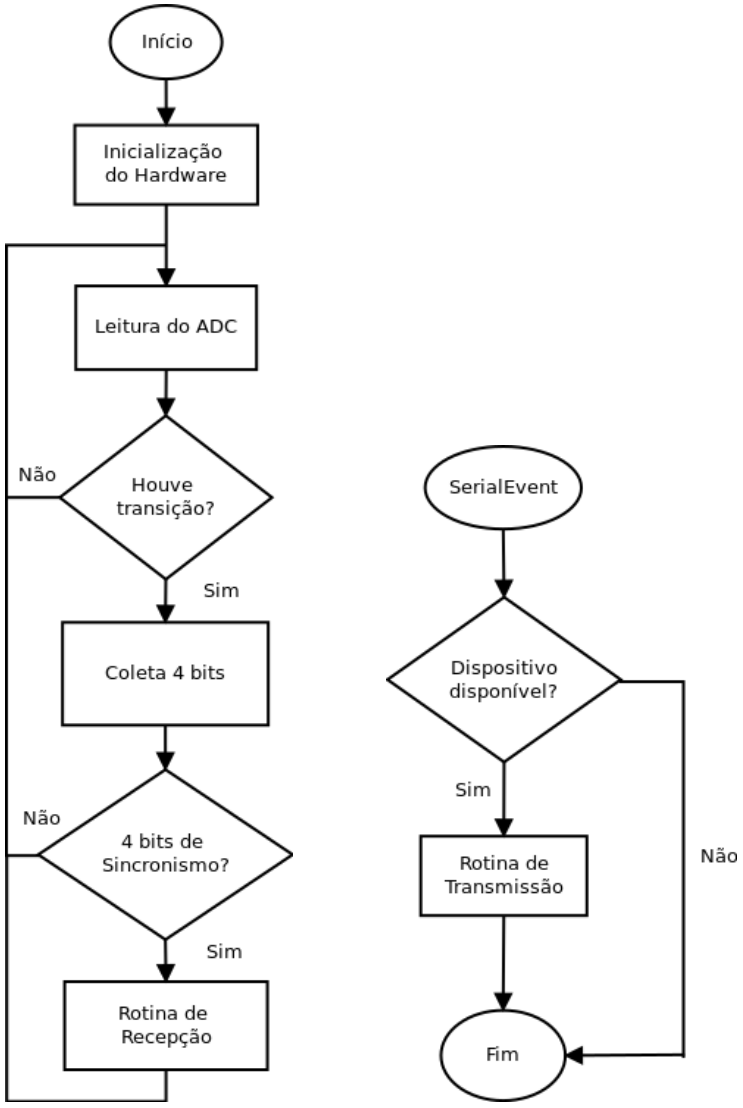
  - Caso haja entrada de dados pelo terminal serial do Atmega328, uma interrupção é ativada, suspendendo momentaneamente o *loop* do primeiro processo;

  - Em seguida é feita uma verificação para confirmar se o microprocessador não está ocupado em uma rotina de recepção;

  - Em caso negativo, o processo inicia a rotina de transmissão;

  - Em caso positivo, o processo apenas finaliza e retorna para o *loop* do primeiro processo. Esse passo funciona apenas como uma medida de segurança no caso de uma tentativa de interromper a rotina de recepção;

  - Ao fim da rotina de transmissão, o processo finaliza e retorna para o *loop* do primeiro processo.



(a) Processo principal. (b) Interrupção SerialEvent.

Figura 3.12. Fluxograma da rotina principal dos dispositivos ativos, onde: (a) Processo principal realizado pelo dispositivo colocando-o em estado "passivo"; (b) Interrupção acionada por entrada de dados pela porta serial.

Rotina de Transmissão

A rotina de transmissão, que é parte integrante do processo "SerialEvent", é representada pelo fluxograma da Figura 3.13, onde estão presentes todas as rotinas secundárias da transmissão.

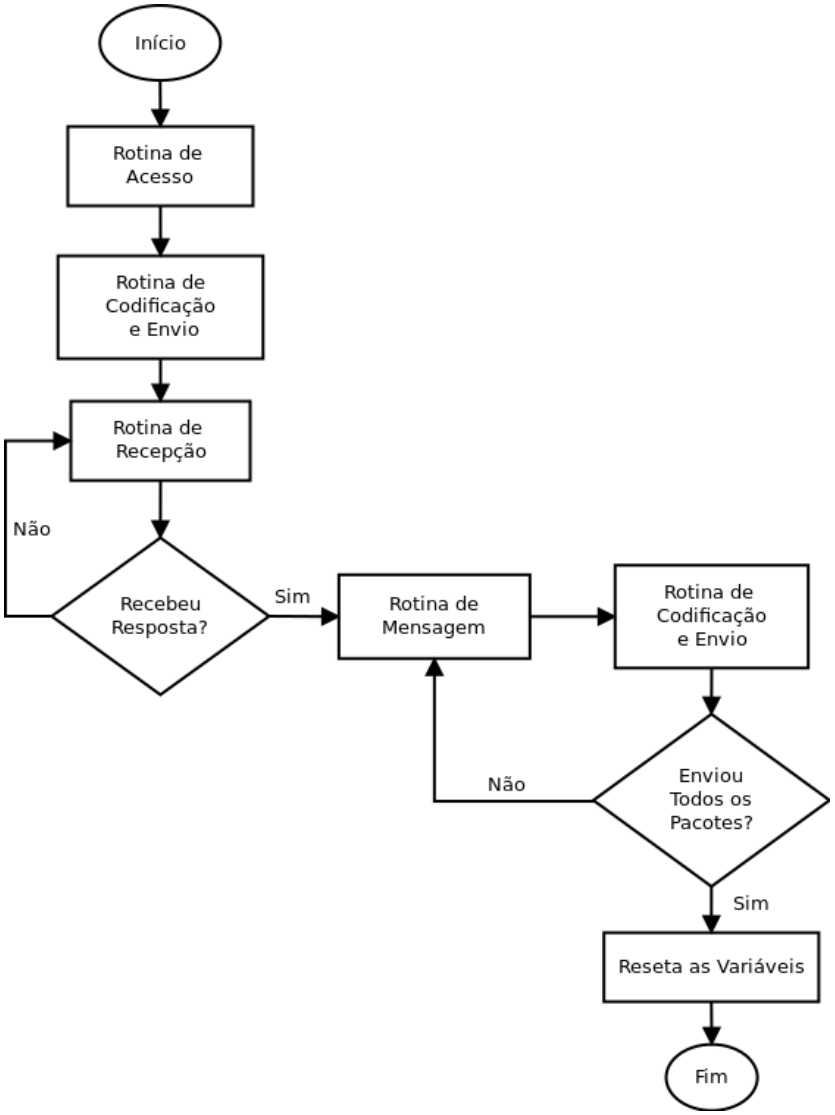


Figura 3.13. Fluxograma da rotina de transmissão dos dispositivos ativos.

A função de cada rotina secundaria é descrita a seguir:

- **Rotina de Acesso:** É responsável por identificar a quantidade de pacotes a serem enviados e montar o pedido de acesso segundo o protocolo da Seção 3.3.2;
- **Rotina de Codificação e Envio:** Como o próprio nome sugere, é responsável por aplicar a codificação adotada no sistema e transmiti-la através da modulação da fonte de emissão;
- **Rotina de Recepção:** Em específico, sua função é aguardar pela resposta do roteador;
- **Rotina de Mensagem:** É responsável por montar os pacotes a serem enviados, segundo o protocolo da Seção 3.3.3;

- **Reseta as Variáveis:** Limpa todas as variáveis do sistema destinadas a transmissão para aguardar uma nova entrada de dados.

Rotina de Recepção

No "modo passivo", quando os bits de sincronismo são identificados, inicia-se a rotina de transmissão, que é representada pelo fluxograma da Figura 3.14.

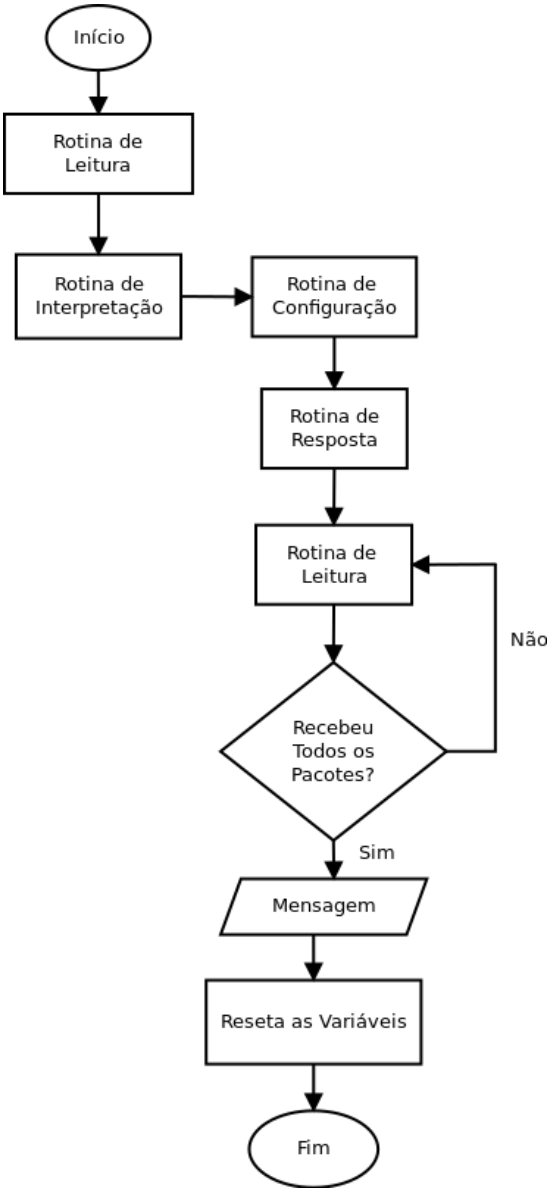


Figura 3.14. Fluxograma da rotina de recepção dos dispositivos ativos.

A função de cada rotina secundária é descrita a seguir:

- **Rotina de Leitura:** É responsável por coletar os pacotes recebidos através do ADC;

- **Rotina de Interpretação:** É responsável por interpretar toda a informação contida no pacote recebido;
- **Rotina de Configuração:** Tem como função principal setar as variáveis do algoritmo para receber a quantidade de pacotes que foram solicitados e interpretados pela rotina anterior;
- **Rotina de Resposta:** Elabora e envia uma resposta ao roteador indicando que está preparado para receber os pacotes;
- **Reseta as Variáveis:** Limpa todas as variáveis do sistema destinadas à recepção para aguardar os pacotes.

3.4.2 Roteador

Assim como nos dispositivos ativos, Seção 3.4.1, o roteador funciona por meio da execução de duas rotinas, são elas: rotina principal e rotina de roteamento. O detalhamento dessas rotinas são apresentadas a seguir.

Rotina Principal

Ao contrário dos dispositivos ativos, o roteador possui um único processo capaz de realizar todas as funções atribuídas a ele, as quais são descritas a seguir:

- O *loop* principal desse processo consiste na verificação das três portas que representam cada um dos dispositivos ativos. Essa verificação segue os seguintes passos:

  - Na inicialização do *hardware* a variável responsável por indicar o endereço da porta recebe o valor da primeira porta;

  - Em seguida, é feita a verificação do ADC da primeira porta;

  - Se houve transição de nível lógico, diferente dos dispositivos ativos, ele já inicia a rotina de roteamento e "bloqueia" a recepção para essa porta;

  - Após a realização da rotina de roteamento, o algoritmo retorna para o *loop* onde a verificação é feita novamente;

  - Se não há transição de nível lógico, o algoritmo seleciona a próxima porta a ser verificada e volta ao início do *loop*;

  - Repete esses passos até haver uma transição de nível lógico em alguma das portas.

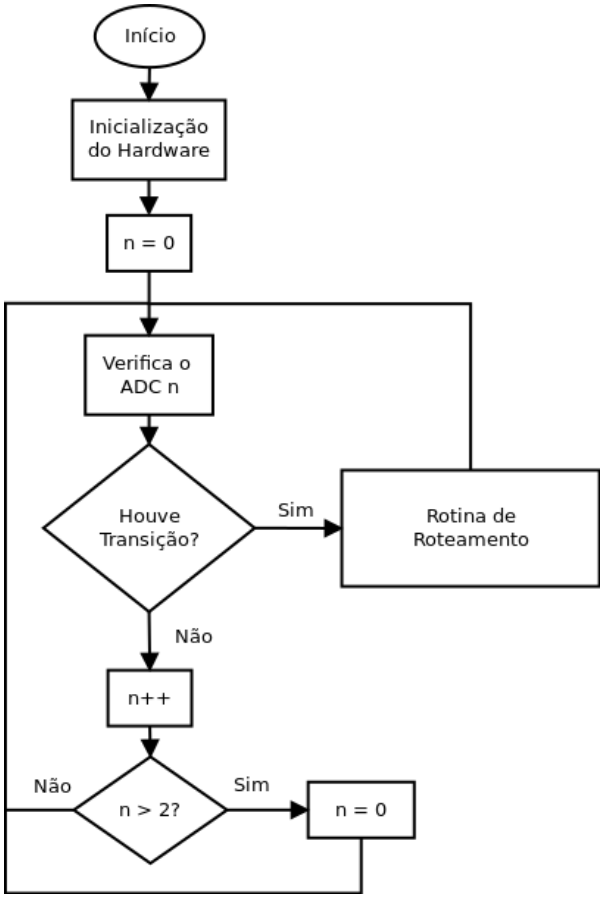


Figura 3.15. Fluxograma da rotina principal do roteador.

Rotina de Roteamento

A rotina de roteamento desempenha um papel fundamental no sistema em questão, pois é responsável por controlar todo o fluxo de informação que passa pela rede. Seu fluxograma é apresentado na Figura 3.16.

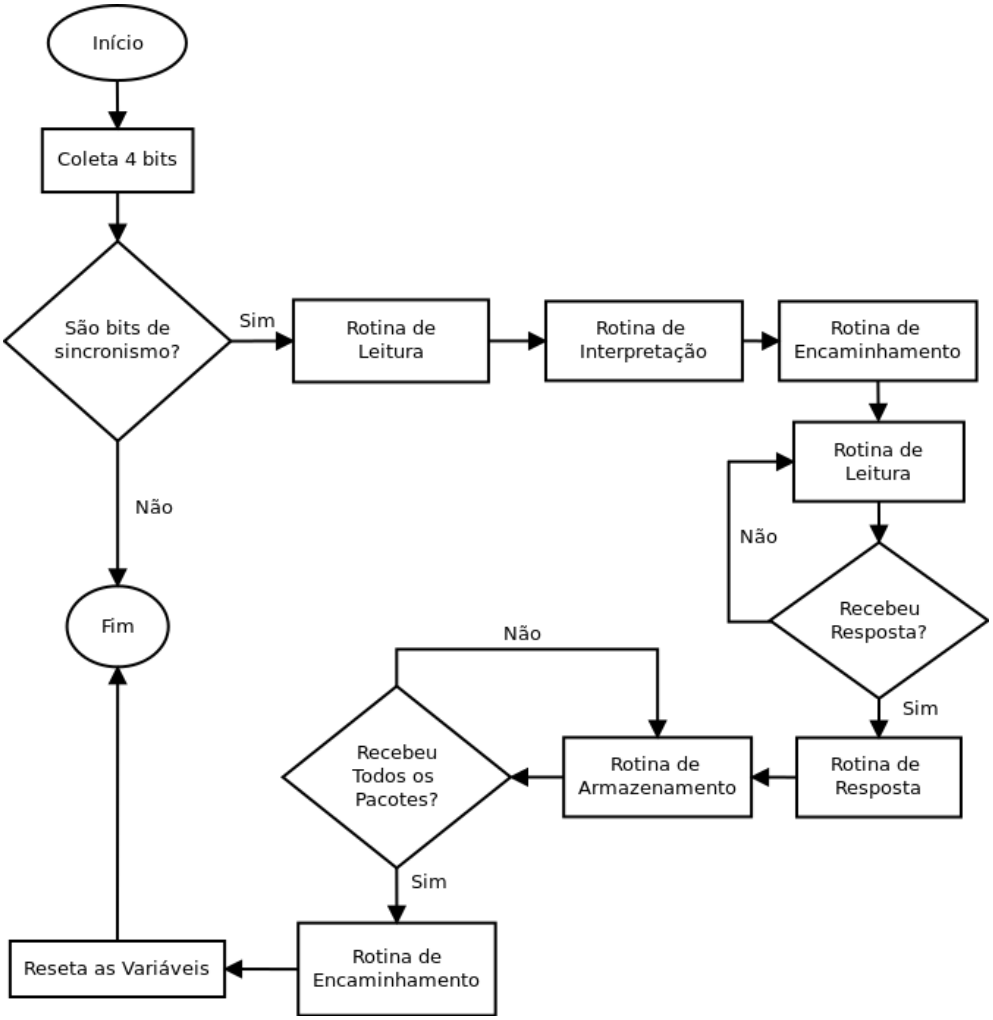


Figura 3.16. Fluxograma da rotina de roteamento do roteador.

É válido notar que algumas rotinas secundárias desse fluxograma já foram apresentadas anteriormente, assim, as novas rotinas são descritas a seguir:

- **Rotina de Encaminhamento:** É responsável por encaminhar os pacotes recebidos ao destino determinado pela rotina de interpretação;
- **Rotina de Armazenamento:** Sua função é armazenar todos os pacotes que foram solicitados para depois encaminhar ao destino;
- **Reseta as Variáveis:** Limpa todas as variáveis do sistema destinadas ao roteamento para aguardar uma nova solicitação/transmissão.

4 Caracterização do sistema

4.1 Experimentos realizados

Para realizar a caracterização do sistema desenvolvido, foram realizados alguns experimentos onde o objetivo principal era obter os valores correspondentes à potência do sinal recebido. Nesse contexto, foram realizados os seguintes experimentos:

- Variação do nível de ruído fazendo medições em momentos distintos do dia, em geral, manhã, tarde e noite. Nesse teste a distância e a angulação permaneceram fixas, respectivamente, em $20cm$ e 90°;
- Variação da distância ao passo de $5cm$, iniciando na marca de $20cm$ e finalizando na marca de $165cm$, o que representa um total de 30 pontos de interesse. Nesse teste, a angulação das placas foram fixadas em 90° (frente a frente, perfeitamente alinhadas);
- Variação da angulação ao passo de 10°, iniciando na marca de 0° e finalizando na marca de 180°, o que representa um total de 19 pontos de interesse. Nesse experimento a distância entre as placas foi fixada em $20cm$.

Nas seções 4.1.1, 4.1.2 e 4.1.3 são descritos, detalhadamente, os procedimentos realizados para obtenção das potências do sinal, e apresentados os resultados obtidos em cada teste.

4.1.1 Variação do nível de ruído do ambiente

Buscando encontrar a melhor hora para utilização do sistema, foi realizado este experimento inicial, onde o ruído do ambiente foi medido a partir das $10h$ até às $22h$ em local fechado com a presença de iluminação externa (luz do sol), de forma que foram obtidas 13 medições. É valido ressaltar que o protótipo utilizado no teste permaneceu imóvel até o fim do experimento, permanecendo sob iluminação ambiente até as $18h$, quando uma lâmpada fluorescente foi acesa. Na Tabela 4.1 estão contidos os resultados do experimento e, na Figura 4.1 é apresentado o gráfico que representa a curva característica do protótipo quanto ao ruído externo.

Tabela 4.1. Valores experimentais obtidos.

Horário	Medida do ADC	Potência (em V^2)
10:05	865	17,839
11:05	865	17,839
12:05	866	17,880
13:05	866	17,880
14:05	866	17,880
15:05	866	17,880
16:05	692	11,417
17:05	594	8,412
18:05	255	1,550
19:05	31	0,023
20:05	13	0,004
21:05	13	0,004
22:05	13	0,004

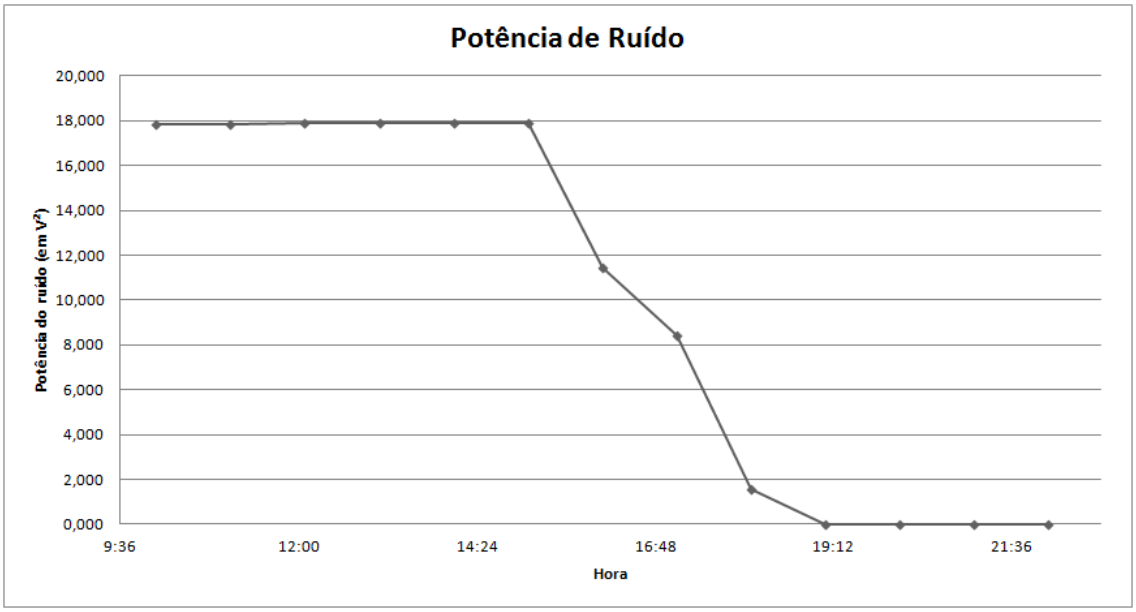


Figura 4.1. Gráfico da potência do ruído ao longo do dia.

Como visto na Seção 3.2.2, o CI OPT101 é bastante sensível à luz infravermelha, o que é provado, de forma empírica, por esse experimento. Analisando a curva característica, nota-se que, das $10h$ às $15h$, nesse ambiente, não é possível realizar uma transmissão, pois nesse horário a potência do ruído tende a ser maior ou igual à potência do sinal, impossibilitando, assim, o uso desse protótipo nesse cenário.

Esse gráfico é de suma importância para auxiliar na escolha do valor de limiar de decisão, que consiste no valor que determina a interpretação do bit, ou seja, é o valor máximo para que a medida seja interpretada como "bit 0", e o valor mínimo para que seja interpretada como "bit 1".

4.1.2 Variação de distância

Nesse teste, primeiramente foi emitido um sinal constante de $5V$ (nível lógico alto) durante todo o processo de aquisição das amostras, representando a potência máxima do sinal. Em seguida foi emitido um sinal com período de $100ms$, que permanece constante em $5V$ (nível lógico alto) durante metade do período e constante em $0V$ (nível lógico baixo) durante a outra metade desse período. O período foi repetido durante todo o processo de aquisição das amostras, representando a potência média do sinal. O procedimento experimental foi realizado em duas etapas, são elas:

1. Coletar os valores lidos pelo OPT101;
2. Converter os valores obtidos para V^2.

Dessa forma, para obter os valores de ADC, medidos pelo OPT101, em cada uma das distâncias, foram coletadas 1000 medições em cada ponto e retirada a média desses valores. Com os valores do ADC correspondentes a cada distância, e uma tensão V_{ref} de $5V$, foi possível calcular a potência do sinal recebido por meio da Equação 4.1.

$$P = \left(\frac{ADC \times V_{ref}}{1024} \right)^2, \tag{4.1}$$

onde ADC é o valor lido pelo sensor OPT101, V_{ref} é a tensão de referência e P é a potência do sinal recebido, medido em V^2.

Os resultados desse teste são apresentados nas Tabelas A.1 e A.4 (Apêndice A), onde estão presentes todos os valores obtidos. De posse desses resultados, utilizando o *Microsoft Excel*®, podemos então, realizar o levantamento da curva que caracteriza o sistema em relação a potência do sinal e a distância entre as placas. A Figura 4.2 apresenta as curvas obtidas.

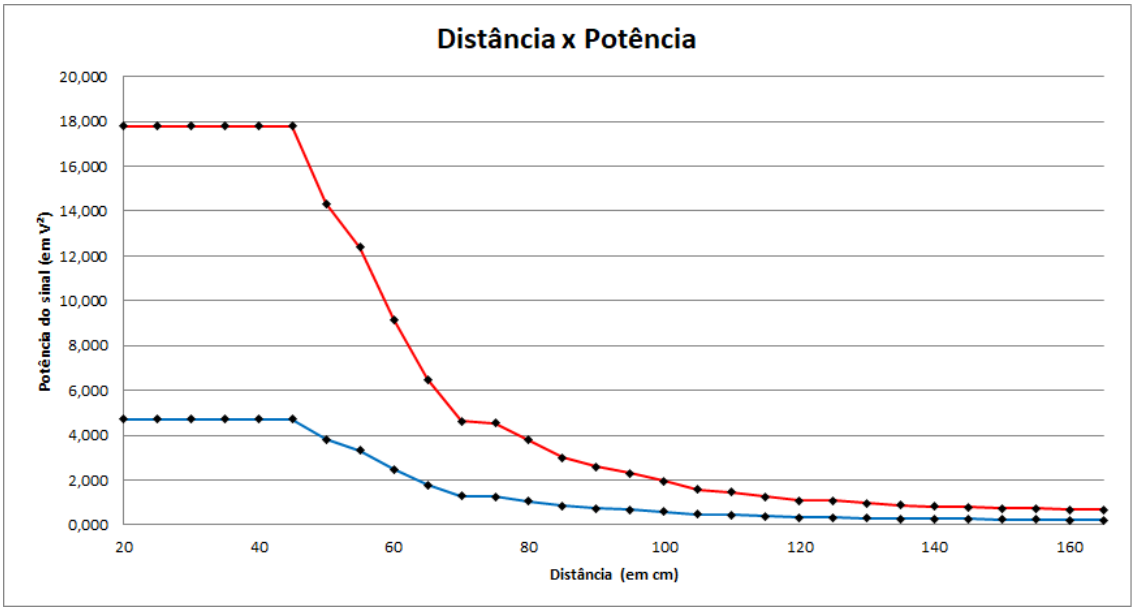


Figura 4.2. Curva característica da relação Distância x Potência. A curva em vermelho representa a potência máxima do sinal, a curva em azul representa a potência média do sinal.

É importante ressaltar que esse experimento foi realizado no período noturno, contando apenas com a iluminação de uma lâmpada fluorescente, que registrava no ADC o valor de 96. Com a Equação 4.1 e adotando $ADC = 96$, encontramos o valor de $0,212V^2$. Dessa forma, é válido notar que a uma distância de $165cm$, com potência de $0,665V^2$, o sistema é capaz de captar a luz emitida, sendo necessário somente ajustar o valor do limiar de decisão de bit.

Relação Sinal Ruído

Sabendo que a potência do ruído nesse experimento foi de $0,212V^2$, podemos calcular a relação entre a potência do sinal e a potência do ruído, que é dada pela Equação 4.2 (Lathi and Ding, 2012).

$$RSR = 10 \times log_{10}\left(\frac{P_{sinal}}{P_{ruido}}\right), \tag{4.2}$$

onde RSR é a relação sinal-ruído, dado em dB, P_{sinal} é a potência do sinal medida no experimento, P_{ruido} é a potência do ruído presente no experimento, dado as condições nas quais o teste foi realizado.

Substituindo os valores de P_{sinal}, presentes nas Tabelas A.1 e A.4 (Apêndice A), na Equação 4.2 obtemos os valores da RSR. As Tabelas A.2 e A.5 (Apêndice A) mostra os resultados obtidos e seu gráfico é apresentado na Figura 4.3.

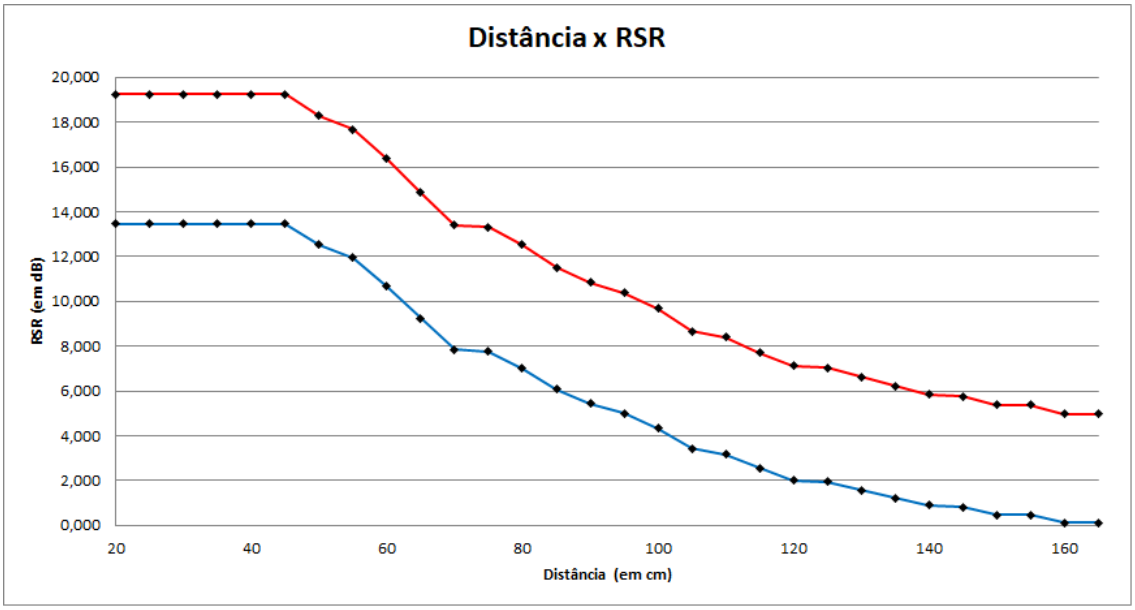


Figura 4.3. Curva característica da relação Distância x RSR. A curva em vermelho representa a potência máxima do sinal, a curva em azul representa a potência média do sinal.

Obtenção da equação do gráfico

Analisando a curva característica (Figura 4.2), observa-se que no inicio de ambas as curvas ocorre uma linearidade devido a saturação do receptor, que da forma como foi projetado só consegue detectar valores até 866. As medidas posteriores apresentam um comportamento semelhante a uma curva exponencial. Dessa forma, pegando apenas os pontos com distância superior a $45cm$, foi elaborado o gráfico da parte não linear (Figura 4.4), onde por meio de uma regressão não linear do tipo potencial foram obtidas as Equações 4.3 e 4.4.

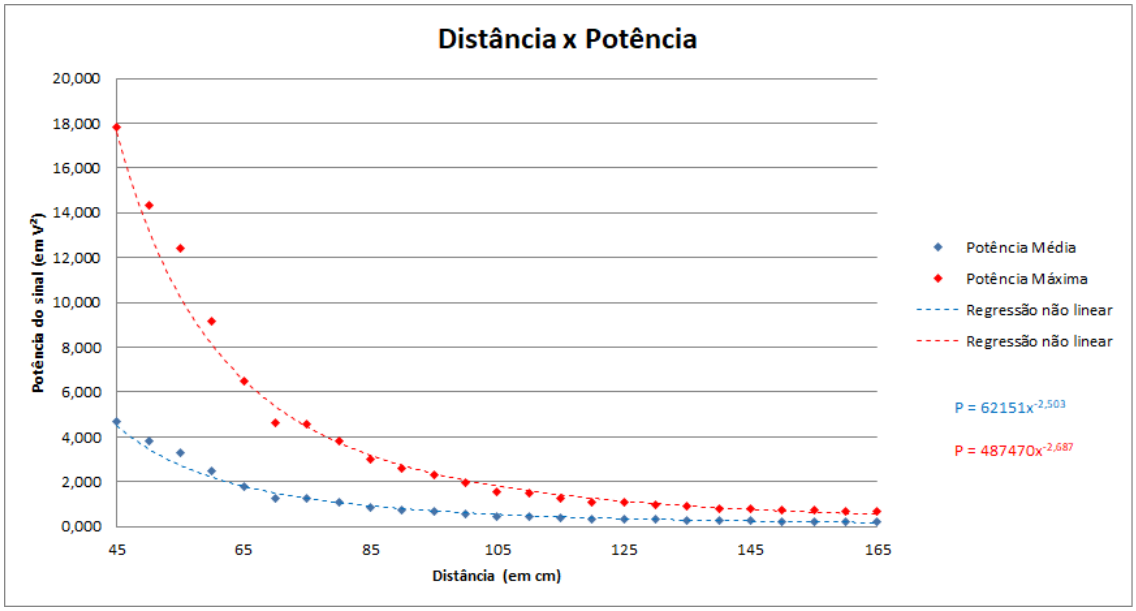


Figura 4.4. Regressão não linear da seção não saturada do gráfico de Distância x Potência. A curva em vermelho representa a potência máxima do sinal, a curva em azul representa a potência média do sinal.

$$P = \left(497987 \times X^{-2,691}\right), \tag{4.3}$$

onde P é a potência calculada e X é a distância entre as placas.

$$P = \left(62151 \times X^{-2,503}\right), \tag{4.4}$$

onde P é a potência calculada e X é a distância entre as placas.

Os valores de potência do sinal foram recalculados utilizando as distâncias presentes nas Tabelas A.1 e A.4 (Apêndice A) e a Equação 4.3, obtendo assim, as Tabelas A.3 e A.6 (Apêndice A), que apresentam os valores calculados. O erro associado foi calculado segundo a Equação 4.5 (Navidi, 2012).

$$e = (P_{experimental} - P_{calculada}), \tag{4.5}$$

onde e é o erro associado, $P_{experimental}$ são os valores reais obtidos no experimento e $P_{calculado}$ são os valores obtidos pela Equação 4.3.

Analisando a Figura 4.4, nota-se que existem pontos que estão acima ou abaixo da curva de regressão. Dessa forma, se faz necessário obter o desvio padrão associado a Equação 4.3. De posse dos valores do erro associado, é possível calcular o desvio padrão do erro, utilizando a Equação 4.6 (Devore, 2010).

$$DP = \sqrt{\frac{(e_1 - \overline{e})^2 + (e_2 - \overline{e})^2 + ... + (e_n - \overline{e})^2}{n}}, \tag{4.6}$$

onde DP é desvio padrão, e_n é o erro associado à medida, n é a quantidade total de amostras e $\overline{e}$ é o erro médio associado.

A partir dos valores presentes nas Tabelas A.3 e A.6 (Apêndice A), obtemos, respectivamente, $\overline{e}_{max} = 0,282$ e $\overline{e}_{med} = 0,087$ e, substituindo na Equação 4.6, temos que:

$$DP_{max} = 0,470$$

$$DP_{med} = 0,234$$

Com isso, as Equações 4.3 e 4.4 podem ser reescritas como mostra as Equações 4.7 e 4.8.

$$P_{max} = \left(497987 \times X^{-2,691}\right) \pm 0,470 \tag{4.7}$$

$$P_{med} = \left(62151 \times X^{-2,503}\right) \pm 0,130 \tag{4.8}$$

4.1.3 Variação de angulação

Nesse experimento, foi emitido novamente um sinal de potência máxima, seguido de um sinal de potência média. De forma semelhante ao que foi realizado no experimento de distância (Seção 4.1.2), foram obtidas 1000 amostras pelo CI OPT101 e foi tirada a média desses valores. A potência do sinal foi calculada utilizando a Equação 4.1 com o mesmo valor para V_{ref}. Os resultados obtidos no experimento estão dispostos nas Tabelas A.7 e A.9 (Apêndice A). A Figura 4.5 apresenta as curvas características do sistema em relação a potência do sinal e a angulação entre as placas.

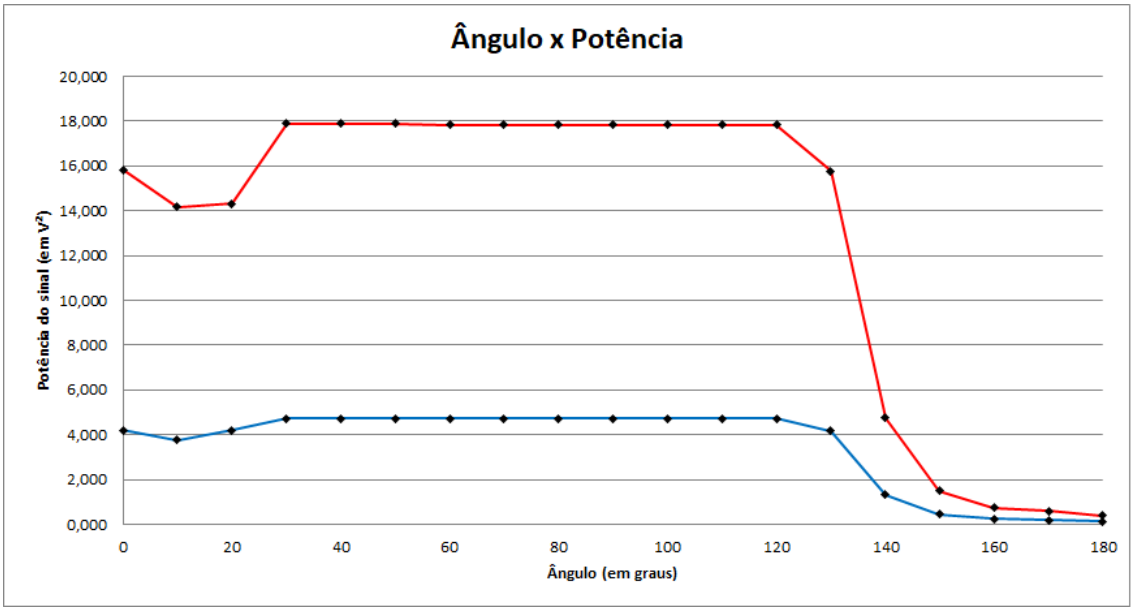


Figura 4.5. Curva característica da relação Ângulo x Potência. A curva em vermelho representa a potência máxima do sinal, a curva em azul representa a potência média do sinal.

Durante a realização desse experimento, que ocorreu no início da noite, com a iluminação de uma lâmpada fluorescente, mediu-se o ruído do ambiente que registrou um valor $ADC = 167$. Assim, utilizando a Equação 4.1, foi encontrado o valor de $0,665V^2$ para a potência do ruído. Analisando as Tabelas A.7 e A.9 (Apêndice A), nota-se que, nesse cenário, o sistema funciona até o ângulo de 140°, pois acima disso P_{ruido} é maior que P_{sinal}, o que pode resultar em uma falha de transmissão. O referencial de angulação é apresentado na Figura 4.6 que mostra a posição considerada como 10°.

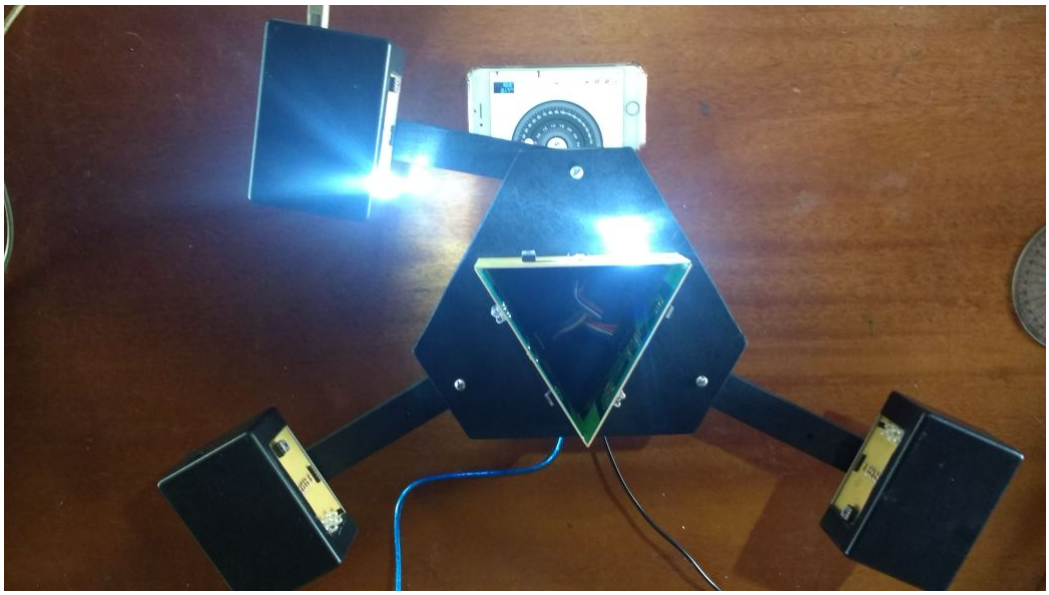


Figura 4.6. Referencial para teste de angulação.

Relação Sinal Ruído

Sabendo-se que $P_{ruido} = 0,665V^2$ e que os valores de P_{sinal} estão listados nas Tabelas A.7 e A.9 (Apêndice A), utilizando a Equação 4.2, foram levantadas as Tabelas A.8 e A.10 (Apêndice A), onde estão dispostos os resultados da RSR. O gráfico que relaciona a angulação entre as placas e a RSR é apresentado na Figura 4.7.

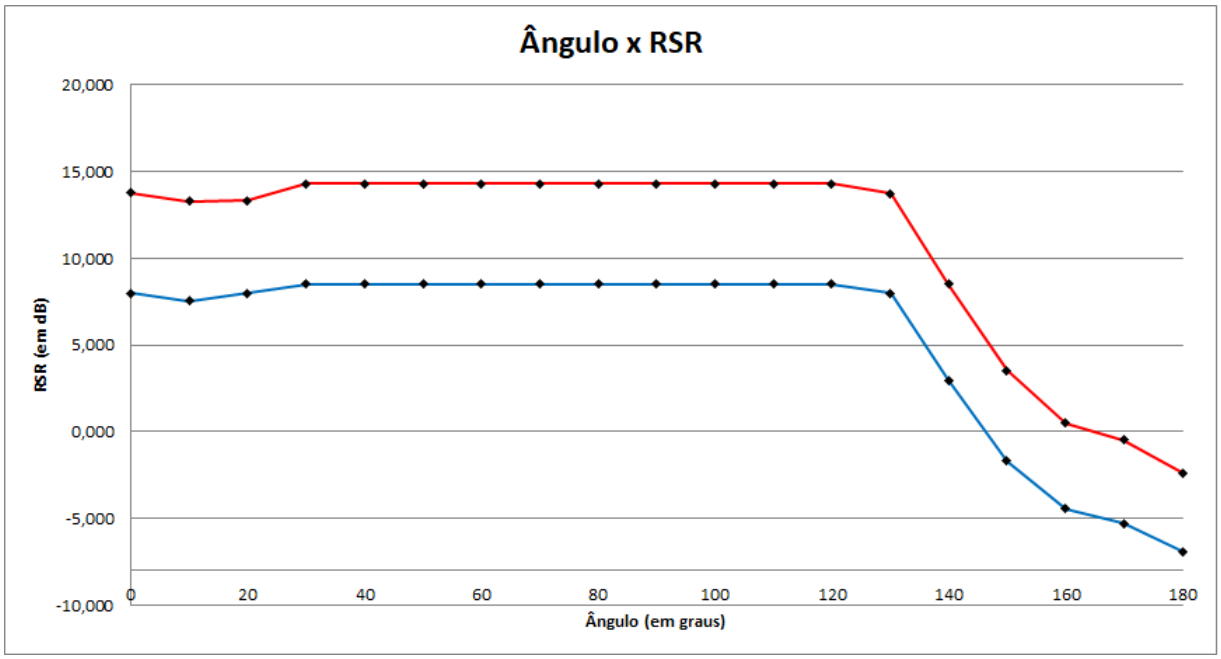


Figura 4.7. Curva característica da relação Ângulo x RSR. A curva em vermelho representa a potência máxima do sinal, a curva em azul representa a potência média do sinal.

4.2 Resultados

A partir da análise dos dados coletados nos experimentos realizados (Seção 4.1), das definições do sistema (Seção 3.1) e dos algoritmos implementados (Seção 3.4), podemos definir o cenário ideal para a correta utilização do protótipo, que é definido nas seções 4.2.1, 4.2.2 e 4.2.3.

4.2.1 Experimentais

Observando os gráficos gerados na Seção 4.1, é possível estabelecer as condições ideais para obter o melhor aproveitamento do sistema implementado. Essas condições são descritas na Tabela 4.2.

Tabela 4.2. Condições de melhor aproveitamento do sistema.

Parâmetro	Valor
Distância entre placas	$20cm$ - $45cm$
Ângulo entre placas	$30°$ - $120°$
Horário de utilização	$18:00$ - $06:00$
Potência máxima do sinal	$4,7V^2$
Potência máxima do ruído	$18,39V^2$

Como dito anteriormente, o teste de ruído é importante para definir o limiar de decisão do sistema. Dessa forma, analisando os dados obtidos na Seção 4.1.1, foi implementada, na inicialização do sistema, uma verificação que utiliza como critério o valor do ruído ambiente, medido pelo ADC, para determinar se o local no qual o protótipo está inserido é adequado para realizar transmissões.

A tomada de decisão relativa ao ambiente, leva em consideração a seguinte condição: Se o valor do ruído medido pelo ADC for igual ou próximo do valor máximo da medida do emissor (que nesse sistema obteve o valor de 864, medido pelo ADC), o sistema não pode operar nesse cenário.

Assim, se o cenário for considerado como não ideal, o sistema não inicializa e tem suas funcionalidades "bloqueadas" até que o mesmo seja levado a um outro ambiente ou o local de operação seja ajustado.

4.2.2 Sistemáticos

A fim de finalizar a caracterização do sistema, se faz necessário buscar informações sobre sua taxa de transmissão e frequência de operação para que um quadro de características possa ser corretamente descrito. A seguir, são descritos os procedimentos para obter as características restantes.

Taxa de operação

A taxa de operação do sistema é calculada pela Equação 4.9 (Lathi and Ding, 2012).

$$F = \frac{1}{T_b}, \tag{4.9}$$

onde F é a frequência e T_b é período do bit.

Adotando $T_b = 100ms$ e substituindo na Equação 4.9, temos que:

$$F = \frac{1}{100 \times 10^{-3}} = 10bauds$$

Taxa de transmissão

A taxa de transmissão do sistema pode ser calculada utilizando a Equação 4.10 (Bezerra, 2010).

$$R_{tx} = F \times log_2 L, \tag{4.10}$$

onde R_{tx} é a taxa de transmissão e L é quantidade de níveis possíveis.

Devido à utilização do código *Manchester*, são necessários 2 bits para representar cada bit dos pacotes originais (que contém 16 bits). Assim, apesar de utilizar apenas 2 das 4 combinações possíveis, $L = 2^2 = 4$ níveis. Substituindo o valor de $F = 10Hz$ e $L = 4$, temos que:

$$R_{tx} = 10 \times log_2 4 = 20bps$$

4.2.3 Gerais

Em resumo, as características gerais do sistema desenvolvido podem ser vistas na Tabela 4.3, que contém todas as informações descritas até aqui.

Tabela 4.3. Características do sistema implementado.

Parâmetro	**Característica**
Emissor	LED Branco de alto brilho
Receptor	CI OPT101
Codificação	*Manchester*
Modulação	Banda Base
Topologia da rede	Estrela
Microcontrolador	Atmega328
Distância entre placas	$20cm$ - $45cm$
Ângulo entre placas	30° - 120°
Horário de utilização	18 : 00 - 06 : 00
Taxa de transmissão	$20bps$
Taxa de operação	$10bauds$
Quantidade de caracteres	1 - 8
Potência máxima do sinal	$4,7V^2$
Potência máxima do ruído	$18,39V^2$

5 Validação da Rede

Os experimentos realizados para caracterização do sistema resultaram na Tabela 4.3 onde estão descritas as condições ideais de funcionamento do sistema. De posse dessa tabela, iniciou-se então, o processo de validação da rede utilizando todos os algoritmos e protocolos desenvolvidos no Capítulo 3.

A validação da rede deu-se gradativamente, onde, realizou-se modificações de acordo com as dificuldades que surgiam a cada implementação. Algumas delas, as quais estavam relacionadas ao sincronismo e aos protocolos, resultaram em mudanças significativas ao projeto inicial do sistema.

Nas seções seguintes, os problemas são apresentados individualmente, com sua descrição e a solução adotada. Por fim, feitas as alterações necessárias, é apresentada a rede em funcionamento.

5.1 Ajuste no Sincronismo

O sincronismo sempre foi um desafio para o projeto, tendo em vista que ele é essencial para a realização de uma transmissão. Inicialmente, com base nos resultados apresentados no trabalho do engenheiro Vilmey Filho, que serviu de base para esse projeto, adotou-se a frequência de operação de $300Hz$, pois segundo Vilmey, "A taxa de erro da transmissão aumenta exponencialmente entre os valores maiores que 300 Hz." (Romano Filho, 2016).

Entretanto, a essa frequência, o algoritmo de recepção desenvolvido não era capaz de detectar os bits de sincronismo. O problema era então causado pela forma como o receptor atua no sistema, pois apesar de captar variações de até $14KHz$, o CI OPT101 juntamente com o Atmega328 não conseguiam detectar as transições de nível lógico, o que causava perdas de pacotes durante a transmissão.

Assim, para solucionar o problema, a frequência de operação fui diminuída gradativamente até atingir a marca de $10Hz$, valor extremamente baixo, porém, com uma boa resposta do sistema.

Com essa frequência de operação, a taxa de transmissão obtida para o sistema foi de $20bps$, que também é muito baixa, entretanto, com esse valor é possível observar a rede em seu devido funcionamento.

5.2 Ajuste no Protocolo

Durante o processo de validação da rede, foi constatado que havia uma falha nos protocolos desenvolvidos na Seção 3.3. A falha foi percebida quando o dispositivo tentava

conseguir acesso ao meio de transmissão, por meio da emissão de um pedido de acesso.

Em sua composição, o pedido de acesso carrega a informação da quantidade de pacotes a serem enviados, e foi justamente nessa parte que a falha originou-se. Observando o código ASCII dos números, percebeu-se que os bits correspondentes aos números ímpares terminavam sempre em "1", enquanto que os números pares terminavam em "0" e, por esse motivo, ao tentar enviar um número par de pacotes uma falha na transmissão ocorria e o receptor não conseguia identificar o número enviado.

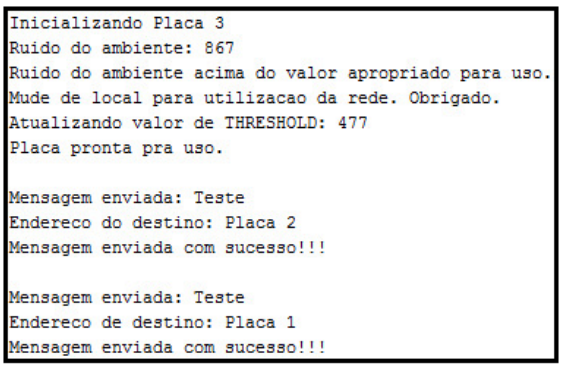
Para resolver esse problema, foi acrescentado ao fim do protocolo de acesso e de transmissão um "*stop bit*" de valor "1". Apesar de resolvido o problema, não fica claro o motivo da ocorrência desse erro. Assim, o comprimento do pacote passou a ser de 17 bits (34 em *Manchester*).

5.3 Rede em Funcionamento

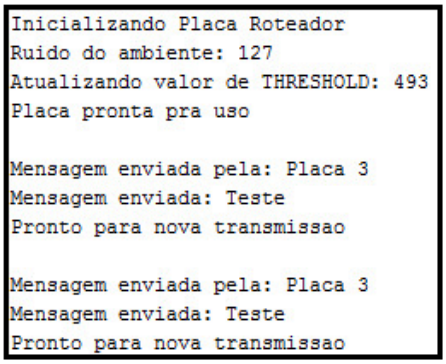
Após aplicadas as correções para os problemas encontrados, foi possível validar a rede desenvolvida através de testes de transmissão de palavras. Para cada placa foi repetido o mesmo teste, que consistia em:

- Realizar a autoconfiguração da rede;
- Da mesma origem enviar uma palavra para dois destinos;
- Comparar a palavra recebida nos destinos com a palavra enviada.

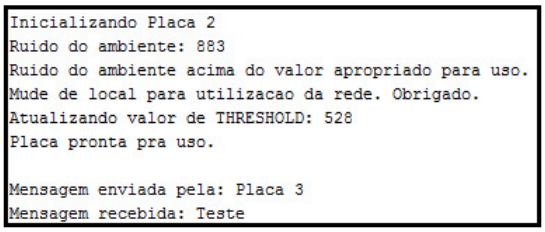
A Figura 5.1 apresenta as mensagens obtidas durante o teste. A Tabela 5.1 resume todos os testes realizados.



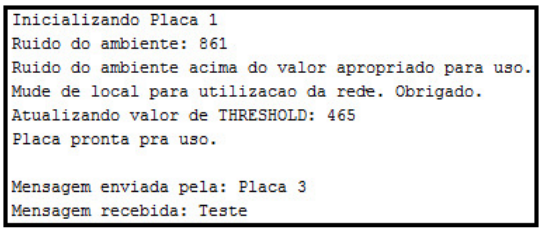
(a) Origem da mensagem (Placa 3).



(b) Roteador.



(c) Primeiro destino (Placa 2).



(d) Segundo destino (Placa 1).

Figura 5.1. Exemplo das mensagens exibidas durante o teste de validação.

Tabela 5.1. Relação de entradas e saídas do teste de validação.

Origem	**Destino**	**Entrada**	**Saída**
Placa 1	Placa 2	“2-Teste”	“Teste”
Placa 1	Placa 3	“3-Teste’	“Teste”
Placa 2	Placa 1	“1-Teste’	“Teste”
Placa 2	Placa 3	“3-Teste’	“Teste”
Placa 3	Placa 1	“1-Teste’	“Teste”
Placa 3	Placa 2	“2-Teste’	“Teste”

6 Considerações Finais

Na busca por inovações tecnológicas para as comunicações sem fio, o VLC destaca-se por suprir algumas das necessidades do mercado, tais como: elevadas taxas de transmissão, baixo custo de implementação e compatibilidade com outros sistemas *wireless* existentes. Dessa forma, este trabalho tem como objetivo: o projeto, a implementação e a caracterização de uma rede digital de transmissão de dados que utiliza o VLC como seu principal fundamento.

Ao longo deste documento, descreveram-se todas as etapas de projeto do sistema, as quais foram implementadas e testadas. Algumas falhas foram observadas durante a fase de testes e prontamente corrigidas, entretanto, essas correções causaram mudanças significativas no projeto idealizado (ver Seções 5.1 e 5.2).

A forma como a rede foi elaborada e as escolhas feitas têm impacto direto na sua caracterização, pois os parâmetros obtidos durante os testes estão diretamente relacionados com o *hardware* escolhido e suas limitações. Em alguns momentos durante a execução do trabalho com o *hardware* escolhido, tanto o microcontrolador quanto o receptor, apresentaram limitações, porém, as decisões de compromisso possibilitaram a realização da validação do sistema com as características apresentadas na Seção 4.2.

Este projeto trata-se de uma prova de conceito, cujo os objetivos propostos foram alcançados. Os resultados obtidos servem como ponto de partida para futuros trabalhos que abordem o uso do VLC em ambientes restritos.

6.1 Propostas de Melhorias Futuras

Apesar do êxito na validação do protótipo, existem melhorias que podem ser realizadas para que o mesmo possa ser aplicado em um ambiente doméstico de forma adequada. As melhorias a serem realizadas buscam aprimorar as características fundamentais do sistema para aproxima-lo do real. Assim, se faz necessária a análise da forma como os parâmetros foram obtidos durante a execução do projeto.

A análise da Tabela 4.3 mostra que alguns fundamentos do VLC não foram observados devidamente. A frequência de operação atingida pelos algoritmos elaborados está muito distante da idealizada para sistemas com essa tecnologia, pois a uma frequência de $10Hz$, é possível ver a olho nu a transição de nível lógico do LED, o que não deveria ocorrer no mesmo. Como consequência da baixa frequência de operação, a taxa de transmissão também mostrou-se bastante reduzida, o que também contraria os fundamentos do VLC.

Além disso, apesar de a taxa de operação ser muito baixa, os algoritmos desenvolvidos ainda apresentam algumas falhas na transmissão dos pacotes. Durante a realização dos testes observou-se que ao transmitir o mesmo pacote repetidas vezes, não era garantido

que o receptor os detectaria corretamente. É válido ressaltar que o ambiente de teste era o mesmo e as transmissões foram sequencialmente repetidas com um intervalo de $200ms$ entre os pacotes.

Não se sabe ao certo o motivo desse comportamento. Entretanto, acredita-se que há relação com a forma que o microprocessador em questão foi utilizado, pois sua capacidade não foi devidamente explorada. Assim, uma forma de solucionar esses problemas consiste em utilizar um microprocessador mais potente que possibilite o paralelismo e o uso de outras linguagens de programação, como por exemplo o VHDL.

Outras melhorias podem ser realizadas para otimizar a eficiência do protótipo, dentre elas pode-se citar: possibilitar transmissão de diversos tipos de arquivos, reelaborar os protocolos para agregar códigos corretores de erro e aumentar a quantidade de informação enviada em um único pacote, aprimorar os sistemas de emissão e recepção de modo a atingir distâncias superiores e desenvolver algoritmos mais eficientes que aproveitem devidamente o *hardware*.

Referências Bibliográficas

Aleixo, J., Meireles, J. F., et al. (2013). Modelagem experimental de um Link FSO com inserção de feixes não difrativos.

Arnon, S. (2003). Optical wireless communications. *Encyclopedia of Optical Engineering*.

Backers, F. (1988). Transparent bridges for interconnection of IEEE 802 LANs. *IEEE Network*, 2(1):5–9.

Bezerra, R. M. (2010). Transmissão digital e analógica. *CEFET/BA*.

Caetano, S. S. (2011). Sistemas de Comunicações Ópticas. `http://www.sj.ifsc.edu.br/~saul/sistemas%20opticos/comunicacoes%20opticas%20I.PDF`. Acesso em: 2017-09-07.

Dantas, M. (2010). *Redes de comunicação e computadores*. Florianópolis: Virtual Books.

Datasheet, T. I. (2003). OPT-101 Monolithic Photodiode and Single-Supply Transimpedance Amplifier. `http://www.ti.com/lit/ds/symlink/opt101.pdf`. Acesso em: 2016-10-13.

De Vries, J. P., Simić, L., Achtzehn, A., Petrova, M., and Mähönen, P. (2014). The wi-fi "congestion crisis": Regulatory criteria for assessing spectrum congestion claims. *Telecommunications Policy*, 38(8):838–850.

Devore, J. L. (2010). *Probabilidade e Estatística para Engenharia e Ciências*. Cengage Learning Edições Ltda.

Faria, V. S. (2011). Tabela ASCII Completa. `https://desenvolvedorinteroperavel.wordpress.com/2011/09/11/tabela-ascii-completa/`. Acesso em: 2017-10-23.

Ferreira, J. Z. (2014). Estudo comparativo entre lâmpadas fluorescentes tubulares T8 e tubulares de LED. *Universidade Tecnológica Federal do Paraná*.

Fornari, E. (1984). O "incrível" padre landell de moura: o brasileiro precursor das telecomunicações.

Haas, H. (2011). Wireless data from every light bulb.

Haruyama, S. (2011). Visible Light Communications: Recent Activities in Japan. *System Design and Management*.

Hecht, J. (2015). *Understanding fiber optics*. Jeff Hecht.

Hutt, D. L., Snell, K. J., and Belanger, P. A. (1993). Alexander graham bell's photophone. *Optics and Photonics News*, 4(6):20–25.

Junior, H. C. (2011). Comunicação de Dados Utilizando Luz Vísivel. *Universidade Federal Fluminense*.

Kachris, C., Bergman, K., and Tomkos, I. (2012). *Optical interconnects for future data center networks*. Springer Science & Business Media.

Kurose, J. F., Ross, K. W., and Zucchi, W. L. (2007). *Redes de Computadores e a Internet: uma abordagem top-down*. Pearson Addison Wesley.

Lathi, B. P. and Ding, Z. (2012). Sistemas de comunicação. In *Sistemas de comunicações analógicos e digitais modernos*, volume 4, pages 1–310.

Lima, C. B. (2012). *AVR e Arduino : tecnicas de projeto*. Pearson Prentice Hall.

Machado, R. (2013). Transmissão digital em banda base.

Mahdy, A. and Deogun, J. (2004). Wireless optical communications: a survey. *Wireless Communications and Networking Conference - WCNC*, Vol. 4.

Maiman, T. H. (1967). Laser applications. *Physics Today*, 20:24.

Navidi, W. (2012). *Probabilidade e estatística para ciências exatas*. AMGH Editora.

Pohlmann, C. (2010). Visible light communication. In *Seminar Kommunikationsstandards in der Medizintechnik*, pages 1–14.

Povey, G. (2011). VLC versus WiFi. Disponível em: `http://visiblelightcomm.com/vlc-versus-wifi-complementary-or-competitors/`. Acesso em: 2016-10-15.

Romano Filho, V. F. (2016). Título: VLC-Comunicação Óptica por Luz Visível. *Universidade de Brasília - FGA*.

Soares, L. F. G., Guido, L., and Colcher, S. (1995). *Redes de Computadores: das LANs, MANs e WANs às redes ATM*, volume 2. Editora Campus.

Souza, J. J., Fonseca, Z. P., and Junior, S. L. S. (2013). Projeto de um sistema de comunicação por luz visível (VLC) baseado em iluminação LED. *8° Encontro de Engenharia e Tecnologia dos Campos Gerais*, 1(1):9.

Tannenbaum, A. S. and Zucchi, W. L. (2009). *Organização estruturada de computadores*. Pearson Prentice Hall.

Tyndall, J. (1886). *Fragments of science: A series of detached essays, addresses and reviews*. Appleton.

Vieira, A. B., Vieira, L. F., Vieira, M., Freire, J., Matheus, L. M., and Gnawali, O. (2017). Comunicação por luz visível: conceito, aplicações e desafios. In *SBRC 2017 - Minicursos*.

Weller, A. (2009). Codificação Manchester (Ethernet). `http://coelhorede.blogspot.com.br/2009/06/codificacao-manchester-ethernet.html`. Acesso em: 2017-08-25.

A Tabelas dos experimentos

A.1 Distância

A.1.1 Potência Máxima

Dados do experimento

Tabela A.1. Resultados experimentais do teste de distância com potência máxima.

Distância (em cm)	Medida do ADC	Potência (em V^2)
20	864	17,798
25	864	17,798
30	864	17,798
35	864	17,798
40	864	17,798
45	864	17,798
50	775	14,320
55	721	12,394
60	620	9,165
65	522	6,497
70	451	4,637
75	437	4,553
80	399	3,796
85	355	3,005
90	329	2,581
95	311	2,306
100	287	1,964
105	256	1,563
110	248	1,466
115	229	1,250
120	214	1,092
125	212	1,072
130	202	0,973
135	193	0,888
140	185	0,816
145	183	0,798
150	175	0,730
155	175	0,730
160	167	0,665
165	167	0,665

Relação Sinal Ruído

Tabela A.2. Valores de RSR obtidos para potência máxima.

Distância (em cm)	**RSR (em dB)**
20	19,240
25	19,240
30	19,240
35	19,240
40	19,240
45	19,240
50	18,296
55	17,669
60	16,358
65	14,863
70	13,399
75	13,320
80	12,530
85	11,515
90	10,854
95	10,365
100	9,668
105	8,675
110	8,399
115	7,707
120	7,118
125	7,037
130	6,617
135	6,221
140	5,853
145	5,759
150	5,371
155	5,371
160	4,964
165	4,964

Regressão não linear

Tabela A.3. Potências recalculadas e erro associado à medida.

Distância (em cm)	**Potência (em V^2) Experimental**	**Potência (em V^2) Regressão**	**Erro associado (e)**
45	17,798	17,610	$\pm 0,187$
50	14,320	13,268	$\pm 1,051$
55	12,394	10,271	$\pm 2,123$
60	9,165	8,129	$\pm 1,035$
65	6,497	6,556	$\pm 0,059$
70	4,637	5,372	$\pm 0,735$
75	4,553	4,463	$\pm 0,089$
80	3,796	3,753	$\pm 0,042$
85	3,005	3,189	$\pm 0,183$
90	2,581	2,735	$\pm 0,158$
95	2,306	2,365	$\pm 0,058$
100	1,964	2,060	$\pm 0,096$
105	1,563	1,807	$\pm 0,244$
110	1,466	1,595	$\pm 0,128$
115	1,250	1,415	$\pm 0,165$
120	1,092	1,262	$\pm 0,170$
125	1,072	1,131	$\pm 0,059$
130	0,973	1,018	$\pm 0,045$
135	0,888	0,920	$\pm 0,031$
140	0,816	0,834	$\pm 0,018$
145	0,798	0,759	$\pm 0,039$
150	0,730	0,693	$\pm 0,037$
155	0,730	0,635	$\pm 0,095$
160	0,665	0,583	$\pm 0,082$
165	0,665	0,536	$\pm 0,128$

A.1.2 Potência Média

Dados do experimento

Tabela A.4. Resultados experimentais do teste de distância com potência média.

Distância (em cm)	**Medida do ADC**	**Potência (em V^2)**
20	444	4,700
25	444	4,700
30	444	4,700
35	444	4,700
40	444	4,700
45	444	4,700
50	399,5	3,805
55	372,5	3,308
60	322	2,472
65	273	1,777
70	232,5	1,289
75	230,5	1,267
80	211,5	1,066
85	189,5	0,856
90	176,5	0,743
95	167,5	0,669
100	155,5	0,577
105	140	0,467
110	136	0,441
115	126,5	0,382
120	119	0,338
125	118	0,332
130	113	0,304
135	108,5	0,281
140	104,5	0,260
145	103,5	0,255
150	99,5	0,236
155	99,5	0,236
160	95,5	0,217
165	95,5	0,217

Relação Sinal Ruído

Tabela A.5. Valores de RSR obtidos para potência média.

Distância (em cm)	**Relação RSR (em db)**
20	13,458
25	13,458
30	13,458
35	13,458
40	13,458
45	13,458
50	12,540
55	11,933
60	10,667
65	9,233
70	7,839
75	7,763
80	7,016
85	6,062
90	5,445
95	4,990
100	4,345
105	3,433
110	3,181
115	2,552
120	2,021
125	1,948
130	1,572
135	1,219
140	0,892
145	0,809
150	0,467
155	0,467
160	0,110
165	0,110

Regressão não linear

Tabela A.6. Potências recalculadas e erro associado à medida.

Distância (em cm)	**Potência (em V^2) Experimental**	**Potência (em V^2) Regressão**	**Erro associado (e)**
45	4,700	4,523	$\pm 0,177$
50	3,805	3,475	$\pm 0,330$
55	3,308	2,737	$\pm 0,571$
60	2,472	2,202	$\pm 0,270$
65	1,777	1,802	$\pm 0,025$
70	1,289	1,497	$\pm 0,208$
75	1,267	1,259	$\pm 0,007$
80	1,066	1,072	$\pm 0,005$
85	0,856	0,921	$\pm 0,065$
90	0,743	0,798	$\pm 0,055$
95	0,669	0,697	$\pm 0,028$
100	0,577	0,613	$\pm 0,036$
105	0,467	0,543	$\pm 0,075$
110	0,441	0,483	$\pm 0,042$
115	0,382	0,432	$\pm 0,051$
120	0,338	0,388	$\pm 0,051$
125	0,332	0,351	$\pm 0,019$
130	0,304	0,318	$\pm 0,013$
135	0,281	0,289	$\pm 0,009$
140	0,260	0,264	$\pm 0,004$
145	0,255	0,242	$\pm 0,014$
150	0,236	0,222	$\pm 0,014$
155	0,236	0,205	$\pm 0,031$
160	0,217	0,189	$\pm 0,028$
165	0,217	0,175	$\pm 0,042$

A.2 Angulação

A.2.1 Potência Máxima

Dados do experimento

Tabela A.7. Resultados experimentais do teste de angulação com potência máxima.

Ângulo (em graus)	**Medida do ADC**	**Potência (em V^2)**
0	815	15,836
10	771	14,173
20	775	14,320
30	866	17,880
40	866	17,880
50	866	17,880
60	865	17,839
80	865	17,839
90	865	17,839
100	865	17,839
110	865	17,839
120	865	17,839
130	813	15,759
140	446	4,743
150	251	1,502
160	177	0,747
170	158	0,595
180	127	0,385

Relação Sinal Ruído

Tabela A.8. Valores de RSR obtidos para potência máxima.

Ângulo (em graus)	**Relação RSR (em db)**
0	13,769
10	13,287
20	13,332
30	14,296
40	14,296
50	14,296
60	14,286
70	14,286
80	14,286
90	14,286
100	14,286
110	14,286
120	14,286
130	13,748
140	8,533
150	3,539
160	0,505
170	-0,481

A.2.2 Potência Média

Dados do experimento

Tabela A.9. Resultados experimentais do teste de angulação com potência média.

Ângulo (em graus)	**Medida do ADC**	**Potência (em V^2)**
0	419,5	4,196
10	397,5	3,767
20	419,5	4,196
30	445	4,721
40	445	4,721
50	445	4,721
60	444,5	4,711
70	444,5	4,711
80	444,5	4,711
90	444,5	4,711
100	444,5	4,711
110	444,5	4,711
120	444,5	4,711
130	418,5	4,176
140	235	1,317
150	137,5	0,451
160	100,5	0,241
170	91	0,197

Relação Sinal Ruído

Tabela A.10. Valores de RSR obtidos para potência média.

Ângulo (em graus)	**Relação RSR (em db)**
0	8,000
10	7,533
20	8,000
30	8,513
40	8,513
50	8,513
60	8,503
70	8,503
80	8,503
90	8,503
100	8,503
110	8,503
120	8,503
130	7,980
140	2,967
150	-1,688
160	-4,411
170	-5,273

B Códigos dos Experimentos

A seguir são apresentados todos os códigos dos algoritmos desenvolvidos para realização dos experimentos caracterizadores da rede.

B.1 Nível lógico constante

```
//-----------------------------------------------//
//Universidade de Brasilia -- Faculdade do Gama
//Trabalho de Conclusao de Curso
//Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
//Autor : Kaio Diego e Paulo Afonso
//-----------------------------------------------//

int i=0;
long int resultado=0;
void setup()
{
  // inicializa com serial com o computador
  Serial.begin(9600);
  pinMode(6,OUTPUT);
}

void loop()
{
    digitalWrite(6,HIGH);

    for (int a = 0; a < 1000; a++)
    {
      resultado += analogRead(A1);
    }
    Serial.println(resultado/1000);
    resultado=0;
  delay(3000);
}
```

B.2 Nível lógico variável

B.2.1 Emissor

```
//-------------------------------------------------//
//Universidade de Brasilia -- Faculdade do Gama
//Trabalho de Conclusao de Curso
//Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
//Autor : Kaio Diego e Paulo Afonso
//-------------------------------------------------//

// Emissor

//-------------------------------------------------//
void setup()
{
  // coloca o pino 8 como saida digital
  DDRB |= (1 << DDB0);
}

void loop()
{
    //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de clock
    PORTB |= B00000001;
    delay(50);

    //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
    PORTB &= B11111110;
    delay(50);

}
```

B.2.2 Receptor

```
//-----------------------------------------------//
//Universidade de Brasilia -- Faculdade do Gama
//Trabalho de Conclusao de Curso
//Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
//Autor : Kaio Diego e Paulo Afonso
//-----------------------------------------------//

//Receptor

//-----------------------------------------------//
int i=0;
long int resultado=0;
void setup()
{
  // inicializa com serial com o computador
  Serial.begin(9600);
  pinMode(6,OUTPUT);
}

void loop()
{
    digitalWrite(6,HIGH);

    for (int a = 0; a < 1000; a++)
    {
      resultado += analogRead(A1);
      delay(50);
    }
    Serial.println(resultado/1000);
    resultado=0;
}
```

C Códigos do Sistema

A seguir são apresentados todos os códigos dos algoritmos desenvolvidos para os dispositivos que compõem a rede.

C.1 Códigos dos Dispositivos Ativos

C.1.1 Placa 1

```
//-----------------------------------------------//
// Universidade de Brasilia - Faculdade do Gama
// Trabalho de Conclusao de Curso
// Ano: 2/2017
// Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
// Autor : Kaio Diego e Paulo Afonso
//-----------------------------------------------//

// Placa P1

//-----------------------------------------------//
// VARIAVEIS GLOBAIS //

// Limiar de descisao entre bit 1 ou 0
int THRESHOLD = 800;

// Matriz de armazenamento das bordas do protocolo manchester enviado e recebidas pelo adc com valor maximo de 8 pacotes
int mensagem[8][26];
// Contador do indice para idenficar o sincronismo
int cont=0;
// Contador da quantidade de pacotes
int qnt_pacotes=0;
// Contador das linhas da matriz mensagem
int linha=0;
// Contador das colunas da matriz mensagem
int coluna=0;
// Limitador de bordas do pacote
int pacote_coluna=26;
// Limitador de pacotes recebidos
int pacote_linha=1;

// Vetor que contem o padrao de sincronismo correto
char sincronismo[8]={'0','1','1','0','0','1','1','0'};
```

```
// Vetor que armazena as bordas e compara com o vetor de sincronismo
char valor_comp[8];

// String que armazena o Destino e Mensagem
String pacote = "";
// String que armazena o Destino
String endereco_dest = "";

// Flag de leitura do adc - Verdadeiro(lendo sincronismo), Falso(lendo
    mensagem)
boolean flag_adc = false;
// Flag que alterna leitura de dados e envio de dados - Verdadeiro(modo
    leitura), Falso(modo mensagem)
boolean flag_TX = false;
// Flag de requesicao
boolean flag_requisicao = true;
// Flag de retorno
boolean flag_retorno= true;

//-------------------------------------------//
//PROTITIPO DE FUNCOES //
void decodifica();
void acesso();
void mensg();
void pacote_retorno();
void pacote_acesso(String endereco_dest ,String endereco_origem, int
    tamanho);
void pacote_mensagem(String endereco_origem, int tamanho);
void trasmissao(int tamanho, String stringOne);
void print_mensagem(char endereco_origem[], int pacote_asc2[]);

//-------------------------------------------//
void setup()
{
    // Inicia a porta serial com velocidade de 9600 bauds
    Serial.begin(9800);
    // Coloca o pino 8 como saida digital
    DDRB |= (1 << DDB0);
    Serial.println("Inicializando Placa 1");
    delay(1000);
    int ruido = analogRead(A0);
    Serial.print("Ruido do ambiente: ");
    Serial.println(ruido);
    if(ruido > THRESHOLD)
    {
        Serial.println("Ruido do ambiente acima do valor apropriado para
            uso.");
```

```
77          Serial.println("Mude de local para utilizacao da rede. Obrigado.");
            while(1)
79          {
            if(analogRead(A0) < THRESHOLD) break;
81          }
        }
83      delay(500);
        ruido = analogRead(A0);
85      Serial.print("Atualizando valor de THRESHOLD: ");
        int media = ruido + ((860-ruido)/2);
87      Serial.println(media);
        THRESHOLD = media;
89      delay(1000);
        Serial.println("Placa pronta pra uso.");
91      Serial.println();
        delay(1000);
93  }

95  void loop()
    {
97      // Coloca o pino 8 em modo HIGH
        PORTB |= B00000001;
99  //-----------------------------------------------//
    // Bloco 1 = Responsavel por ler os valores da porta adc
101
        if(!flag_TX)
103     {
            int valor = analogRead(A0);
105         if(flag_adc)
            {
107             mensagem[linha][coluna] = valor;
                coluna++;
109             if((coluna == pacote_coluna))
                {
111                 flag_adc = false;
                    linha++;
113                 coluna=0;
                }
115             if(linha == pacote_linha)
                {
117                 linha=0;
                    decodifica();
119             }
            }
121         if(!flag_adc)
            {
123             if(valor < THRESHOLD)
```

```
            {
125             valor_comp[cont] = '0';
            }
127         else valor_comp[cont] = '1';

129         if(valor_comp[cont] == sincronismo[cont])
            {
131             cont++;
            }
133         else cont=0;

135         if(cont == 8)
            {
137             cont=0;
                flag_adc = true;
139         }
        }
141     delay(50);
    }
143 //-----------------------------------------------//
// Bloco 2 = Responsavel por iniciar a rotina de trasmissao
145
    else
147 {
        acesso();
149     flag_TX = false;
    }
151 }
//-----------------------------------------------//
153 void decodifica()
{
155 int l=0,c=0,soma=0,cont1=0,cont2=0;
    int EXP[8] = {128, 64, 32, 16, 8, 4, 2, 1};
157 int pacote_asc2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
    char endereco_origem[2];
159
    if(flag_requisicao)
161 {
       for(l=0; l<pacote_linha; l++)
        {
163         for(c=0; c<4; c+=2)
            {
165                 if(mensagem[l][c] < mensagem[l][c+1])
                    {
167                     endereco_origem[cont1] = '1';
                        cont1++;
169                 }
```

```
                    else
171                 {
                        endereco_origem[cont1] = '0';
173                     cont1++;
                    }
175             }
                cont1=0;
177         }

179         for(l=0; l<pacote_linha; l++)
            {
181             for(c=8; c<24; c+=2)
                {
183                     if(mensagem[l][c] < mensagem[l][c+1])
                        {
185                         soma += EXP[cont2];
                            cont2++;
187                     }
                        else
189                     {
                            cont2++;
191                     }
                }
193             pacote_asc2[l] = soma;
                cont2=0;
195             soma=0;
            }
197
            if(flag_retorno)
199         {
                pacote_retorno();
201             pacote_linha = pacote_asc2[0] - 48;
                flag_retorno = false;
203         }
            else
205         {
                print_mensagem(endereco_origem, pacote_asc2);
207             pacote_linha = 1;
                pacote_coluna = 26;
209             flag_retorno = true;
            }
211     }
        else
213     {
            if(mensagem[0][0] < THRESHOLD && mensagem[0][1] > THRESHOLD)
215         {
                delay(500);
```

```
217             mensg();
            }
219     }
}
221 //-------------------------------------------------//
// Bloco 4 = Responsavel por verificar o endereco de destino digitado,
    pegar a quantidade de pacotes a ser enviado e modificar as variaveis
    pacote_linha e pacote_coluna para receber a liberacao para envio da
    mensagem
223
void acesso()
225 {
    String endereco_origem = "01";
227
    if((pacote[0] == '0') || (pacote[0] == '1') )
229     {
        Serial.println("Endereco de destino invalido. Digite novamente!!!")
            ;
231         pacote = "";
        qnt_pacotes=0;
233         Serial.println();
    }
235     else
    {
237         Serial.print("Mensagem enviada: ");
        for(int c=2; c<qnt_pacotes; c++)
239         {
            Serial.print(pacote[c]);
241         }
        Serial.println();
243         if(pacote[0] == '2')
        {
245             Serial.print("Endereco de destino: ");
            Serial.println("Placa 2");
247             endereco_dest = "10";
        }
249         else
        {
251             Serial.print("Endereco do destino: ");
            Serial.println("Placa 3");
253             endereco_dest = "11";
        }
255
        int tamanho = qnt_pacotes - 2;
257         pacote_acesso(endereco_dest,endereco_origem,tamanho+48);
        pacote_coluna = 2;
259         pacote_linha = 1;
```

```
    }
}
//-----------------------------------------------//
// Bloco 5: Responsavel por montar o pacote de acesso ao roteador, enviando o endereco de destino e a quantidade de pacotes a ser enviados

void pacote_acesso(String endereco_dest ,String endereco_origem,int tamanho)
{
    //Converte int para ASCII
    String stringOne =  String(tamanho, BIN);

    while(stringOne.length() < 8)
    {
        //Completa o vetor ASCII com os zeros iniciais
        stringOne = 0 + stringOne;
    }

    //Pacote: Endereco + Mensagem
    stringOne = endereco_dest + stringOne;
    //Pacote: Endereco + Destino + Mensagem
    stringOne = endereco_origem + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "1010" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
    stringOne = stringOne + "1";

    trasmissao(16,stringOne);
}
//-----------------------------------------------//
// Bloco 6: Responsavel por montar o pacote de retorno ao roteador para liberar o envio da mensagem
void pacote_retorno()
{
    String stringOne = "1";
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "1010" + stringOne;
    delay(50);
    trasmissao(4,stringOne);
}
//-----------------------------------------------//
// Bloco 11: Responsavel por imprimir na serial a mansagem e o endereco do remetente

void print_mensagem(char endereco_origem[2], int pacote_asc2[8])
{
    int a=0;
```

```
303     Serial.print("Mensagem enviada pela: ");
        if(endereco_origem[0] == '0' && endereco_origem[1] == '1')
305     {
            Serial.println("Placa 1");
307     }
        else
309     {
            if(endereco_origem[0] == '1' && endereco_origem[1] == '0')
311         {
                Serial.println("Placa 2");
313         }
            else Serial.println("Placa 3");
315     }

317     Serial.print("Mensagem recebida: ");

319     for(a=0; a<pacote_linha; a++)
        {
321         Serial.write(pacote_asc2[a]);
        }
323     Serial.println();
        Serial.println();
325 }
    //-----------------------------------------------//
327 // Bloco 7: Responsavel por enviar a mensagem ao roteador e limpar as
        variaveis para uma nova trasmissao

329 void mensg()
    {
331     String endereco_origem = "01";

333     for(int b=2; b<qnt_pacotes; b++)
        {
335         pacote_mensagem(endereco_origem, b);
            delay(200);
337     }
        Serial.println("Mensagem enviada com sucesso!!!");
339     Serial.println();
        pacote = "";
341     qnt_pacotes=0;
        flag_requisicao = true;
343     pacote_coluna = 26;
        pacote_linha = 1;
345 }
    //-----------------------------------------------//
347 // Bloco 8: Responsavel por montar cada pacote da mensagem a ser enviada
```

```
349 void pacote_mensagem(String endereco_origem, int b)
{
351   char posicao = pacote.charAt(b);
   //Converte Char para Binario
353   String stringOne =  String(posicao, BIN);

355   while(stringOne.length() < 8)
   {
357     //Completa o vetor ASCII com os zeros iniciais
     stringOne = 0 + stringOne;
359   }
     //Pacote: Destino + Mensagem
361     stringOne = endereco_dest + stringOne;
     //Pacote: Endereco + Destino + Mensagem
363     stringOne = endereco_origem + stringOne;
     //Pacote: Sincronismo + Endereco + Destino + Mensagem
365     stringOne = "1010" + stringOne;
     //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
367     stringOne = stringOne + "1";

369     Serial.println(stringOne);
     trasmissao(16,stringOne);
371 }
//-----------------------------------------//
373 // Bloco 9: Responsavel por converter o pacote ASCII em codificacao
    manchester

375 void trasmissao(int tamanho, String stringOne)
{
377     for(int i=0; i<=tamanho; i++)
     {
379         if(stringOne[i] == '1')
         {
381             //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
             PORTB &= B11111110;
383             delay(50);
             //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
                 clock _1
385             PORTB |= B00000001;
             delay(50);
387         }
         else
389         {
             //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
                 clock
391             PORTB |= B00000001;
             delay(50);
```

```
393             //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
                PORTB &= B11111110;
395             delay(50);
            }
397     }
}
399 //-----------------------------------------------//
// Bloco 3 = Interrupcao responsavel por ler os dados digitados na serial e
    montar o pacote da mensagem
401
void serialEvent()
403 {
    while (Serial.available())
405     {
        // get the new byte:
407         char inChar = (Serial.read());
        if (inChar == '\n')
409         {
            flag_TX = true;
411             flag_requisicao = false;
        }
413         else
        {
415             qnt_pacotes++;
            pacote += inChar;
417         }
    }
419 }
//-----------------------------------------------//
```

C.1.2 Placa 2

```
//-----------------------------------------------//
// Universidade de Brasilia - Faculdade do Gama
// Trabalho de Conclusao de Curso
// Ano: 2/2017
// Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
// Autor : Kaio Diego e Paulo Afonso
//-----------------------------------------------//

// Placa P2

//-----------------------------------------------//
// VARIAVEIS GLOBAIS //

// Limiar de descisao entre bit 1 ou 0
int THRESHOLD = 800;

// Matriz de armazenamento das bordas do protocolo manchester enviado e recebidas pelo adc com valor maximo de 8 pacotes
int mensagem[8][26];
// Contador do indice para idenficar o sincronismo
int cont=0;
// Contador da quantidade de pacotes
int qnt_pacotes=0;
// Contador das linhas da matriz mensagem
int linha=0;
// Contador das colunas da matriz mensagem
int coluna=0;
// Limitador de bordas do pacote
int pacote_coluna=26;
// Limitador de pacotes recebidos
int pacote_linha=1;

// Vetor que contem o padrao de sincronismo correto
char sincronismo[8]={'0','1','1','0','0','1','1','0'};
// Vetor que armazena as bordas e compara com o vetor de sincronismo
char valor_comp[8];

// String que armazena o Destino e Mensagem
String pacote = "";
// String que armazena o Destino
String endereco_dest = "";

// Flag de leitura do adc - Verdadeiro(lendo sincronismo), Falso(lendo mensagem)
```

```
boolean flag_adc = false;
// Flag que alterna leitura de dados e envio de dados - Verdadeiro(modo leitura), Falso(modo mensagem)
boolean flag_TX = false;
// Flag de requesicao
boolean flag_requisicao = true;
// Flag de retorno
boolean flag_retorno= true;

//-------------------------------------------//
//PROTITIPO DE FUNCOES //
void decodifica();
void acesso();
void mensg();
void pacote_acesso(int tamanho);
void pacote_mensagem(int tamanho);
void trasmissao(int tamanho, String stringOne);
void print_mensagem(char endereco_origem[], int pacote_asc2[]);

//-------------------------------------------//
void setup()
{
    // Inicia a porta serial com velocidade de 9600 bauds
    Serial.begin(9800);    ,
    // Coloca o pino 8 como saida digital
    DDRB |= (1 << DDB0);
    Serial.println("Inicializando Placa 2");
    delay(1000);
    int ruido = analogRead(A0);
    Serial.print("Ruido do ambiente: ");
    Serial.println(ruido);
    if(ruido > THRESHOLD)
    {
        Serial.println("Ruido do ambiente acima do valor apropriado para uso.");
        Serial.println("Mude de local para utilizacao da rede. Obrigado.");
        while(1)
        {
            if(analogRead(A0) < THRESHOLD) break;
        }
    }
    delay(500);
    ruido = analogRead(A0);
    Serial.print("Atualizando valor de THRESHOLD: ");
    int media = ruido + ((860-ruido)/2);
    Serial.println(media);
    THRESHOLD = media;
```

```
88      delay(1000);
        Serial.println("Placa pronta pra uso.");
90      Serial.println();
        delay(1000);
92  }

94  void loop()
    {
96      // Coloca o pino 8 em modo HIGH
        PORTB |= B00000001;
98  //-----------------------------------------//
    // Bloco 1 = Responsavel por ler os valores da porta adc
100
        if(!flag_TX)
102     {
            int valor = analogRead(A0);
104         if(flag_adc)
            {
106             mensagem[linha][coluna] = valor;
                coluna++;
108             if((coluna == pacote_coluna))
                {
110                 flag_adc = false;
                    linha++;
112                 coluna=0;
                }
114             if(linha == pacote_linha)
                {
116                 linha=0;
                    decodifica();
118             }
            }
120         if(!flag_adc)
            {
122             if(valor < THRESHOLD)
                {
124                 valor_comp[cont] = '0';
                }
126             else valor_comp[cont] = '1';

128             if(valor_comp[cont] == sincronismo[cont])
                {
130                 cont++;
                }
132             else cont=0;

134             if(cont == 8)
```

```
            {
136             cont=0;
                flag_adc = true;
138         }
        }
140     delay(50);
    }
142 //-------------------------------------------------//
 // Bloco 2 = Responsavel por iniciar a rotina de trasmissao
144
    else
146 {
        acesso();
148     flag_TX = false;
    }
150 }
//-------------------------------------------------//
152 // Bloco 3 = Interrupcao responsavel por ler os dados digitados na serial e
    montar o pacote da mensagem

154 void serialEvent()
{
156     while (Serial.available())
    {
158     // get the new byte:
        char inChar = (Serial.read());
160
        if (inChar == '\n')
162     {
            flag_TX = true;
164         flag_requisicao = false;
        }
166     else
        {
168         qnt_pacotes++;
            pacote += inChar;
170     }
    }
172 }
//-------------------------------------------------//
174 // Bloco 4 = Responsavel por verificar o endereco de destino digitado,
    pegar a quantidade de pacotes a ser enviado e modificar as variaveis
    pacote_linha e pacote_coluna para receber a liberacao para envio da
    mensagem

176 void acesso()
{
```

```
178     if((pacote[0] == '0') || (pacote[0] == '2') )
        {
180         Serial.println("Endereco de destino invalido. Digite novamente!!!")
                ;
            pacote = "";
182         qnt_pacotes=0;
            Serial.println();
184     }
        else
186     {
            Serial.print("Mensagem enviada: ");
188         for(int c=2; c<qnt_pacotes; c++)
            {
190             Serial.print(pacote[c]);
            }
192         Serial.println();
            if(pacote[0] == '1')
194         {
                Serial.print("Endereco de destino: ");
196             Serial.println("Placa 1");
                endereco_dest = "01";
198         }
            else
200         {
                Serial.print("Endereco do destino: ");
202             Serial.println("Placa 3");
                endereco_dest = "11";
204         }

206         int tamanho = qnt_pacotes - 2;
            pacote_acesso(tamanho+48);
208         pacote_coluna = 2;
            pacote_linha = 1;
210     }
}
212 //-------------------------------------------//
// Bloco 5: Responsavel por montar o pacote de acesso ao roteador, enviando
    o endereco de destino e a quantidade de pacotes a ser enviados
214
void pacote_acesso(int tamanho)
216 {
    //Converte int para ASCII
218 String stringOne =  String(tamanho, BIN);

220 while(stringOne.length() < 8)
    {
222     //Completa o vetor ASCII com os zeros iniciais
```

```
        stringOne = 0 + stringOne;
    }

    //Pacote: Endereco + Mensagem
    stringOne = endereco_dest + stringOne;
    //Pacote: Endereco + Destino + Mensagem
    stringOne = "10" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "1010" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
    stringOne = stringOne + "1";

    trasmissao(16,stringOne);
}
//-------------------------------------------//
 // Bloco 6: Responsavel por montar o pacote de retorno ao roteador para
     liberar o envio da mensagem

void pacote_retorno()
{
    String stringOne = "1";
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "1010" + stringOne;
    delay(50);
    trasmissao(4,stringOne);
}
//-------------------------------------------//
// Bloco 7: Responsavel por enviar a mensagem ao roteador e limpar as
    variaveis para uma nova trasmissao

void mensg()
{
    for(int b=2; b<qnt_pacotes; b++)
    {
        pacote_mensagem(b);
        delay(200);
    }
    Serial.println("Mensagem enviada com sucesso!!!");
    Serial.println();
    pacote = "";
    endereco_dest = "";
    qnt_pacotes=0;
    flag_requisicao = true;
    pacote_coluna = 26;
    pacote_linha = 1;
}
//-------------------------------------------//
```

```
// Bloco 8: Responsavel por montar cada pacote da mensagem a ser enviada

void pacote_mensagem(int b)
{
    char posicao = pacote.charAt(b);
    //Converte Char para Binario
    String stringOne =  String(posicao, BIN);

    while(stringOne. length() < 8)
    {
        //Completa o vetor ASCII com os zeros iniciais
        stringOne = 0 + stringOne;
    }

    //Pacote: Destino + Mensagem
    stringOne = endereco_dest + stringOne;
    //Pacote: Endereco + Destino + Mensagem
    //stringOne = "01" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "101010" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
    stringOne = stringOne + "1";

    trasmissao(16,stringOne);
}
//-----------------------------------------------//
// Bloco 9: Responsavel por converter o pacote ASCII em codificacao
   manchester

void trasmissao(int tamanho, String stringOne)
{
    for(int i=0; i<=tamanho; i++)
    {
        if(stringOne[i] == '1')
        {
            //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
            PORTB &= B11111110;
            delay(50);
            //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
               clock _1
            PORTB |= B00000001;
            delay(50);
        }
        else
        {
            //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
               clock
```

```
312             PORTB |= B00000001;
                delay(50);
314             //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
                PORTB &= B11111110;
316             delay(50);
            }
318     }
}
320 //-----------------------------------------------//
// Bloco 10: Responsavel por decotificar a mensagem recebida, responder ao
    pedido de acesso e limpar as variaveis
322
void decodifica()
324 {
    int l=0,c=0,soma=0,cont1=0,cont2=0;
326     int EXP[8] = {128, 64, 32, 16, 8, 4, 2, 1};
    int pacote_asc2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
328     char endereco_origem[2];

330     if(flag_requisicao)
    {
332         for(l=0; l<pacote_linha; l++)
        {
334             for(c=0; c<4; c+=2)
            {
336                 if(mensagem[l][c] < mensagem[l][c+1])
                {
338                     endereco_origem[cont1] = '1';
                    cont1++;
340                 }
                else
342                 {
                    endereco_origem[cont1] = '0';
344                     cont1++;
                }
346             }
            cont1=0;
348         }

350         for(l=0; l<pacote_linha; l++)
        {
352             for(c=8; c<24; c+=2)
            {
354                 if(mensagem[l][c] < mensagem[l][c+1])
                {
356                     soma += EXP[cont2];
                    cont2++;
```

```
358                 }
                    else
360                 {
                        cont2++;
362                 }
                }
364             pacote_asc2[1] = soma;
                cont2=0;
366             soma=0;
            }
368
            if(flag_retorno)
370         {
                pacote_retorno();
372             pacote_linha = pacote_asc2[0] - 48;
                flag_retorno = false;
374         }
            else
376         {
                print_mensagem(endereco_origem, pacote_asc2);
378             pacote_linha = 1;
                pacote_coluna = 26;
380             flag_retorno = true;
            }
382     }
        else
384     {
            if(mensagem[0][0] < THRESHOLD && mensagem[0][1] > THRESHOLD)
386         {
                delay(500);
388             mensg();
            }
390     }
}
392 //-------------------------------------------------//
// Bloco 11: Responsavel por imprimir na serial a mansagem e o endereco do
    remetente
394
void print_mensagem(char endereco_origem[2], int pacote_asc2[8])
396 {
    int a=0;
398 Serial.print("Mensagem enviada pela: ");
    if(endereco_origem[0] == '0' && endereco_origem[1] == '1')
400 {
        Serial.println("Placa 1");
402 }
    else
```

```
404     {
            if(endereco_origem[0] == '1' && endereco_origem[1] == '0')
406         {
                Serial.println("Placa 2");
408         }
            else Serial.println("Placa 3");
410     }

412     Serial.print("Mensagem recebida: ");

414     for(a=0; a<pacote_linha; a++)
        {
416         Serial.write(pacote_asc2[a]);
        }
418     Serial.println();
        Serial.println();
420 }
    //---------------------------------------------//
```

C.1.3 Placa 3

```
//-----------------------------------------------//
// Universidade de Brasilia - Faculdade do Gama
// Trabalho de Conclusao de Curso
// Ano: 2/2017
// Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
// Autor : Kaio Diego e Paulo Afonso
//-----------------------------------------------//

// Placa P3

//-----------------------------------------------//
// VARIAVEIS GLOBAIS //

// Limiar de descisao entre bit 1 ou 0
int THRESHOLD = 800;

// Matriz de armazenamento das bordas do protocolo manchester enviado e recebidas pelo adc com valor maximo de 8 pacotes
int mensagem[8][26];
// Contador do indice para idenficar o sincronismo
int cont=0;
// Contador da quantidade de pacotes
int qnt_pacotes=0;
// Contador das linhas da matriz mensagem
int linha=0;
// Contador das colunas da matriz mensagem
int coluna=0;
// Limitador de bordas do pacote
int pacote_coluna=26;
// Limitador de pacotes recebidos
int pacote_linha=1;

// Vetor que contem o padrao de sincronismo correto
char sincronismo[8]={'0','1','1','0','0','1','1','0'};
// Vetor que armazena as bordas e compara com o vetor de sincronismo
char valor_comp[8];

// String que armazena o Destino e Mensagem
String pacote = "";
// String que armazena o Destino
String endereco_dest = "";

// Flag de leitura do adc - Verdadeiro(lendo sincronismo), Falso(lendo mensagem)
```

```
boolean flag_adc = false;
// Flag que alterna leitura de dados e envio de dados - Verdadeiro(modo
    leitura), Falso(modo mensagem)
boolean flag_TX = false;
 // Flag de requesicao
boolean flag_requisicao = true;
// Flag de retorno
boolean flag_retorno= true;

//-----------------------------------------------//
//PROTITIPO DE FUNCOES //
void decodifica();
void acesso();
void mensg();
void pacote_acesso(int tamanho);
void pacote_mensagem(int tamanho);
void trasmissao(int tamanho, String stringOne);
void print_mensagem(char endereco_origem[], int pacote_asc2[]);

//-----------------------------------------------//
void setup()
{
    // Inicia a porta serial com velocidade de 9600 bauds
    Serial.begin(9800);
    // Coloca o pino 8 como saida digital
    DDRB |= (1 << DDB0);
    Serial.println("Inicializando Placa 3");
    delay(1000);
    int ruido = analogRead(A0);
    Serial.print("Ruido do ambiente: ");
    Serial.println(ruido);
    if(ruido > THRESHOLD)
    {
        Serial.println("Ruido do ambiente acima do valor apropriado para
            uso.");
        Serial.println("Mude de local para utilizacao da rede. Obrigado.");
        while(1)
        {
            if(analogRead(A0) < THRESHOLD) break;
        }
    }
    delay(500);
    ruido = analogRead(A0);
    Serial.print("Atualizando valor de THRESHOLD: ");
    int media = ruido + ((860-ruido)/2);
    Serial.println(media);
    THRESHOLD = media;
```

```
    delay(1000);
    Serial.println("Placa pronta pra uso.");
    Serial.println();
    delay(1000);
}

void loop()
{
    // Coloca o pino 8 em modo HIGH
    PORTB |= B00000001;
//-----------------------------------------//
// Bloco 1 = Responsavel por ler os valores da porta adc

    if(!flag_TX)
    {
        int valor = analogRead(A0);

        if(flag_adc)
        {
            mensagem[linha][coluna] = valor;
            coluna++;
            if((coluna == pacote_coluna))
            {
                flag_adc = false;
                linha++;
                coluna=0;
            }
            if(linha == pacote_linha)
            {
                linha=0;
                decodifica();
            }
        }
        if(!flag_adc)
        {
            if(valor < THRESHOLD)
            {
                valor_comp[cont] = '0';
            }
            else valor_comp[cont] = '1';

            if(valor_comp[cont] == sincronismo[cont])
            {
                cont++;
            }
            else cont=0;
```

```
135             if(cont == 8)
                {
137                 cont=0;
                    flag_adc = true;
139             }
            }
141         delay(50);
        }
143 //-------------------------------------------------//
    // Bloco 2 = Responsavel por iniciar a rotina de trasmissao
145
        else
147     {
            acesso();
149         flag_TX = false;
        }
151 }
    //-------------------------------------------------//
153 // Bloco 3 = Interrupcao responsavel por ler os dados digitados na serial e
        montar o pacote da mensagem

155 void serialEvent()
    {
157     while (Serial.available())
        {
159         // get the new byte:
            char inChar = (Serial.read());
161
            if (inChar == '\n')
163         {
                flag_TX = true;
165             flag_requisicao = false;
            }
167         else
            {
169             qnt_pacotes++;
                pacote += inChar;
171         }
        }
173 }
    //-------------------------------------------------//
175 // Bloco 4 = Responsavel por verificar o endereco de destino digitado,
       pegar a quantidade de pacotes a ser enviado e modificar as variaveis
       pacote_linha e pacote_coluna para receber a liberacao para envio da
       mensagem

177 void acesso()
```

```
{
179     if((pacote[0] == '0') || (pacote[0] == '3') )
        {
181         Serial.println("Endereco de destino invalido. Digite novamente!!!")
                ;
            pacote = "";
183         qnt_pacotes=0;
            Serial.println();
185     }
        else
187     {
            Serial.print("Mensagem enviada: ");
189         for(int c=2; c<qnt_pacotes; c++)
            {
191             Serial.print(pacote[c]);
            }
193         Serial.println();
            if(pacote[0] == '1')
195         {
                Serial.print("Endereco de destino: ");
197             Serial.println("Placa 1");
                endereco_dest = "01";
199         }
           else
201        {
                Serial.print("Endereco do destino: ");
203             Serial.println("Placa 2");
                endereco_dest = "10";
205        }

207        int tamanho = qnt_pacotes - 2;
           pacote_acesso(tamanho+48);
209        pacote_coluna = 2;
           pacote_linha = 1;
211    }
}
213 //-----------------------------------------------//
// Bloco 5: Responsavel por montar o pacote de acesso ao roteador, enviando
    o endereco de destino e a quantidade de pacotes a ser enviados
215
void pacote_acesso(int tamanho)
217 {
    //Converte int para ASCII
219 String stringOne =  String(tamanho, BIN);

221 while(stringOne.length() < 8)
    {
```

```
223         //Completa o vetor ASCII com os zeros iniciais
            stringOne = 0 + stringOne;
225     }

227     //Pacote: Endereco + Mensagem
        stringOne = endereco_dest + stringOne;
229     //Pacote: Endereco + Destino + Mensagem
        stringOne = "11" + stringOne;
231     //Pacote: Sincronismo + Endereco + Destino + Mensagem
        stringOne = "1010" + stringOne;
233     //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
        stringOne = stringOne + "1";
235
        trasmissao(16,stringOne);
237 }
    //-----------------------------------------------//
239 // Bloco 6: Responsavel por montar o pacote de retorno ao roteador para
        liberar o envio da mensagem

241 void pacote_retorno()
    {
243     String stringOne = "1";
        //Pacote: Sincronismo + Endereco + Destino + Mensagem
245     stringOne = "1010" + stringOne;
        delay(50);
247     trasmissao(4,stringOne);
    }
249 //-----------------------------------------------//
    // Bloco 7: Responsavel por enviar a mensagem ao roteador e limpar as
        variaveis para uma nova trasmissao
251
    void mensg()
253 {
        for(int b=2; b<qnt_pacotes; b++)
255     {
            pacote_mensagem(b);
257         delay(200);
        }
259     Serial.println("Mensagem enviada com sucesso!!!");
        Serial.println();
261     pacote = "";
        endereco_dest = "";
263     qnt_pacotes=0;
        flag_requisicao = true;
265     pacote_coluna = 26;
        pacote_linha = 1;
267 }
```

```
//-------------------------------------------//
// Bloco 8: Responsavel por montar cada pacote da mensagem a ser enviada

void pacote_mensagem(int b)
{
    char posicao = pacote.charAt(b);
    //Converte Char para Binario
    String stringOne =  String(posicao, BIN);

    while(stringOne.length() < 8)
    {
        //Completa o vetor ASCII com os zeros iniciais
        stringOne = 0 + stringOne;
    }

    //Pacote: Destino + Mensagem
    stringOne = endereco_dest + stringOne;
    //Pacote: Endereco + Destino + Mensagem
    //stringOne = "01" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "101011" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
    stringOne = stringOne + "1";

    trasmissao(16,stringOne);
}
//-------------------------------------------//
// Bloco 9: Responsavel por converter o pacote ASCII em codificacao
   manchester

void trasmissao(int tamanho, String stringOne)
{
    for(int i=0; i<=tamanho; i++)
    {
        if(stringOne[i] == '1')
        {
            //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
            PORTB &= B11111110;
            delay(50);
            //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
               clock _1
            PORTB |= B00000001;
            delay(50);
        }
        else
        {
```

```
            //pino 8 - HIGH; Uso de registrador pra diminuir ciclos de
                clock
313         PORTB |= B00000001;
            delay(50);
315         //pino 8 - LOW; Uso de registrador pra diminuir ciclos de clock
            PORTB &= B11111110;
317         delay(50);
        }
319 }
}
321 //-----------------------------------------------//
// Bloco 10: Responsavel por decotificar a mensagem recebida, responder ao
    pedido de acesso e limpar as variaveis
323
void decodifica()
325 {
    int l=0,c=0,soma=0,cont1=0,cont2=0;
327 int EXP[8] = {128, 64, 32, 16, 8, 4, 2, 1};
    int pacote_asc2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
329 char endereco_origem[2];

331 if(flag_requisicao)
    {
333     for(l=0; l<pacote_linha; l++)
        {
335         for(c=0; c<4; c+=2)
            {
337             if(mensagem[l][c] < mensagem[l][c+1])
                {
339                 endereco_origem[cont1] = '1';
                    cont1++;
341             }
                else
343             {
                    endereco_origem[cont1] = '0';
345                 cont1++;
                }
347         }
            cont1=0;
349     }

351     for(l=0; l<pacote_linha; l++)
        {
353         for(c=8; c<24; c+=2)
            {
355             if(mensagem[l][c] < mensagem[l][c+1])
                {
```

```
357                         soma += EXP[cont2];
                            cont2++;
359                     }
                        else
361                     {
                            cont2++;
363                     }
                }
365             pacote_asc2[1] = soma;
                cont2=0;
367             soma=0;
            }
369
            if(flag_retorno)
371         {
                pacote_retorno();
373             pacote_linha = pacote_asc2[0] - 48;
                flag_retorno = false;
375         }
            else
377         {
                print_mensagem(endereco_origem, pacote_asc2);
379             pacote_linha = 1;
                pacote_coluna = 26;
381             flag_retorno = true;
            }
383     }
        else
385     {
            if(mensagem[0][0] < THRESHOLD && mensagem[0][1] > THRESHOLD)
387         {
                delay(500);
389             mensg();
            }
391     }
}
393 //-------------------------------------------------//
// Bloco 11: Responsavel por imprimir na serial a mensagem e o endereco do
    remetente
395
void print_mensagem(char endereco_origem[2], int pacote_asc2[8])
397 {
        int a=0;
399     Serial.print("Mensagem enviada pela: ");
        if(endereco_origem[0] == '0' && endereco_origem[1] == '1')
401     {
            Serial.println("Placa 1");
```

```
403     }
        else
405     {
            if(endereco_origem[0] == '1' && endereco_origem[1] == '0')
407         {
                Serial.println("Placa 2");
409         }
            else Serial.println("Placa 3");
411     }

413     Serial.print("Mensagem recebida: ");

415     for(a=0; a<pacote_linha; a++)
        {
417         Serial.write(pacote_asc2[a]);
        }
419     Serial.println();
        Serial.println();
421 }
    //-----------------------------------------------//
```

C.2 Código do Roteador

```
//-------------------------------------------------//
//Universidade de Brasilia - Faculdade do Gama
//Trabalho de Conclusao de Curso
//Projeto, Implementacao e Caracterizacao de uma rede de comunicacao via luz visivel
//Autor : Kaio Diego e Paulo Afonso
//-------------------------------------------------//

// Placa Roteador

//-------------------------------------------------//
// VARIAVEIS GLOBAIS //

// Limiar de descisao entre bit 1 ou 0
int THRESHOLD = 800;
// Variavel de escolha do pino do adc
int adc = 0;

// Matriz de armazenamento das bordas do protocolo manchester enviado e recebidas pelo adc com valor maximo de 8 pacotes
int mensagem[8][26];
// Contador do indice para idenficar o sincronismo
int cont=0;
// Contador das linhas da matriz mensagem
int linha=0;
// Contador das colunas da matriz mensagem
int coluna=0;
// Limitador de bordas do pacote
int pacote_coluna=26;
// Limitador de pacotes recebidos
int pacote_linha=1;
// Variavel que armazena qnt de pacotes a ser enviado
int qnt=0;

// Vetor que contem o padrao de sincronismo correto
char sincronismo[8]={'0','1','1','0','0','1','1','0'};
// Vetor que armazena as bordas e compara com o vetor de sincronismo
char valor_comp[8];

// String que armazena o Destino e Mensagem
String pacote = "";

// Flag de leitura do adc - Verdadeiro(lendo sincronismo), Falso(lendo mensagem)
boolean flag_adc = false;
```

```
// Flag que alterna leitura de dados e envio de dados - Verdadeiro(modo leitura), Falso(modo mensagem)
boolean flag_TX = false;
// Flag de requesicao para envio de mensagem - Verdadeiro(envia pacote com a requisicao)
boolean flag_requisicao = true;
// Flag para modo de leitura de liberacao de mensagem para escrita da mensagem final
boolean flag_retorno = true;
// Flag do multiplexador do adc
boolean flag_roteador = false;
//-------------------------------------------//
//PROTITIPO DE FUNCOES //
void decodifica();
void acesso(char endereco[], int qnt_pacotes);
void pacote_acesso(String endereco_dest ,String endereco_origem, int destino, int qnt_pacotes);
void trasmissao(int destino, int tamanho, String stringOne);
void pacote_retorno(char endereco);
void print_mensagem(char endereco[], int pacote_asc2[]);
void mensg(char endereco[], int pacote_asc2[]);
void pacote_mensagem(String endereco_dest, String endereco_origem, int destino, int b, int pacote_asc2[]);
//-------------------------------------------//

void setup()
{
    // Inicia a porta serial com velocidade de 9800 bauds
    Serial.begin(9800);
    //coloca os pinos 5,6,7 como saida digital
    DDRD |= B11100000;
    Serial.println("Inicializando Placa Roteador");
    delay(1000);
    int ruido = ((analogRead(A0) + analogRead(A1) + analogRead(A2))/3);
    Serial.print("Ruido do ambiente: ");
    Serial.println(ruido);
    // Rotina de calibracao do valor de threshold utlizado no limiar de descisao
    if(ruido > THRESHOLD)
    {
        Serial.println("Ruido do ambiente acima do valor apropriado para uso.");
        Serial.println("Mude de local para utilizacao da rede. Obrigado.");
        while(1)
        {
            if(analogRead(A0) < THRESHOLD) break;
        }
```

```
    }
    delay(500);
    ruido = ((analogRead(A0) + analogRead(A1) + analogRead(A2))/3);
    Serial.print("Atualizando valor de THRESHOLD: ");
    int media = ruido + ((860-ruido)/2);
    Serial.println(media);
    THRESHOLD = media;
    delay(1000);
    Serial.println("Placa pronta pra uso");
    Serial.println();
    delay(1000);
}

//-----------------------------------------------//
void loop()
{
    //coloca os pinos 5,6,7 como HIGH
    PORTD |= B11100000;

    // Bloco 1:  Responsavel por ler os valores da porta adc qnd a
       flag_roteador tiver habilitada
    if(flag_roteador)
    {
        int valor = analogRead(adc);

        if(flag_adc)
        {
            mensagem[linha][coluna] = valor;
            coluna++;
            if((coluna == pacote_coluna))
            {
                flag_adc = false;
                linha++;
                coluna=0;
            }
            if(linha == pacote_linha)
            {
                linha=0;
                decodifica();
            }
       }
       if(!flag_adc)
       {
            if(valor < THRESHOLD)
            {
                valor_comp[cont] = '0';
            }
```

```
            else valor_comp[cont] = '1';

            if(valor_comp[cont] == sincronismo[cont])
            {
                cont++;
            }
            else cont=0;

            if(cont == 8)
            {
                cont=0;
                flag_adc = true;
            }
        }
        delay(50);
    }

    // Bloco 2: Responsavel por ler os valores das 3 postas adc e liberar
       flag_roteador quando houver um pedido de acesso
    else
    {
        int valor = analogRead(adc);
        if(valor < THRESHOLD)
        {
            flag_roteador = true;
        }
        else
        {
            if(adc == 2) adc=0;
            else adc++;
        }
    }
}
//-----------------------------------------------//
// Bloco 3: Responsavel por decodificar o pedido de acesso, encaminhar para
    o destino, ler a resposta, liberar a transmissao da mensagem,
   decodificar a mensagem, encaminhar para o destino e limpar as variaveis
   para uma nova trasmissao;

void decodifica()
{
    int l=0,c=0,soma=0,cont1=0,cont2=0,qnt_pacotes=0;
    int EXP[8] = {128, 64, 32, 16, 8, 4, 2, 1};
    int pacote_asc2[8] = {0, 0, 0, 0, 0, 0, 0, 0};
    char endereco[4];

    if(flag_requisicao)
```

```
172     {
            for(l=0; l<pacote_linha; l++)
174         {
                for(c=0; c<8; c+=2)
176             {
                    if(mensagem[l][c] < mensagem[l][c+1])
178                 {
                        endereco[cont1] = '1';
180                     cont1++;
                    }
182                 else
                    {
184                     endereco[cont1] = '0';
                        cont1++;
186                 }
                }
188             cont1=0;
            }
190
            for(l=0; l<pacote_linha; l++)
192         {
                for(c=8; c<24; c+=2)
194             {
                    if(mensagem[l][c] < mensagem[l][c+1])
196                 {
                        soma += EXP[cont2];
198                     cont2++;
                    }
200                 else
                    {
202                     cont2++;
                    }
204             }
                pacote_asc2[l] = soma;
206             cont2=0;
                soma=0;
208         }

210         if(flag_retorno)
            {
212             qnt = pacote_asc2[0];
                acesso(endereco, qnt);
214             pacote_coluna = 2;
                pacote_linha = 1;
216             flag_requisicao = false;
                flag_retorno=false;
218         }
```

```
        else
        {
            print_mensagem(endereco, pacote_asc2);
            mensg(endereco, pacote_asc2);
            pacote_linha = 1;
            pacote_coluna = 26;
            flag_retorno = true;
            flag_roteador = false;
            Serial.println("Pronto para nova transmissao");
            Serial.println();
        }
    }
    else
    {
        if(mensagem[0][0] < THRESHOLD && mensagem[0][1] > THRESHOLD)
        {
            pacote_retorno(endereco);
            flag_requisicao=true;
            pacote_linha=qnt-48;
            pacote_coluna=26;
        }
    }
}
//-----------------------------------------------//
// Bloco 4: Responsavel por montar o endereco de origem e destino
    utilizados no pecote de acesso
void acesso(char endereco[4], int qnt_pacotes)
{
    int destino=0;
    String endereco_origem = "";
    String endereco_dest = "";

    if(endereco[0] == '0' && endereco[1] == '1')
    {
        endereco_origem = "01";
    }
    else
    {
        if(endereco[0] == '1' && endereco[1] == '0')
        {
        endereco_origem = "10";
        }
        else endereco_origem = "11";
    }

    if(endereco[2] == '0' && endereco[3] == '1')
    {
```

```
        destino=1;
        adc=0;
        endereco_dest= "01";
    }
    else
    {
        if(endereco[2] == '1' && endereco[3] == '0')
        {
        destino=2;
        adc=1;
        endereco_dest= "10";
        }
        else
        {
            destino=3;
            adc=2;
            endereco_dest= "11";
        }
    }
    pacote_acesso(endereco_dest, endereco_origem, destino, qnt_pacotes);
}
//-----------------------------------------------//
// Bloco 5: Responsavel por montar o pacote de acesso, enviando o endereco
   de destino e a quantidade de pacotes a ser enviados
void pacote_acesso(String endereco_dest ,String endereco_origem , int
   destino, int qnt_pacotes)
{
    //Converte Char para Binario
    String stringOne =  String(qnt_pacotes, BIN);

    while(stringOne.length() < 8)
    {
        //Completa o vetor ASCII com os zeros iniciais
        stringOne = 0 + stringOne;
    }

    //Pacote: Destino + Mensagem
    stringOne = endereco_dest + stringOne;
    //Pacote: Endereco + Destino + Mensagem
    stringOne = endereco_origem + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem
    stringOne = "1010" + stringOne;
    //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
    stringOne = stringOne + "1";

    trasmissao(destino,16,stringOne);
}
```

```
310 //-------------------------------------------//
    // Bloco 6: Responsavel por montar o pacote de retorno para liberar o envio
        da mensagem
312 void pacote_retorno(char endereco[4])
    {
314     int destino=0;
        if(endereco[0] == '0' && endereco[1] == '1')
316     {
            destino=1;
318         adc=0;
        }
320     else
        {
322         if(endereco[0] == '1' && endereco[1] == '0')
            {
324             destino=2;
                adc=1;
326         }
            else
328         {
                destino=3;
330             adc=2;
            }
332     }

334     String stringOne = "1";
        //Pacote: Sincronismo + Endereco + Destino + Mensagem
336     stringOne = "1010" + stringOne;
        delay(50);
338     trasmissao(destino,4,stringOne);
    }
340 //-------------------------------------------//
    // Bloco 7: Responsavel por imprimir na serial a mansagem e o endereco do
       remetente
342 void print_mensagem(char endereco[4], int pacote_asc2[8])
    {
344     int a=0;
        Serial.print("Mensagem enviada pela: ");
346     if(endereco[0] == '0' && endereco[1] == '1')
        {
348         Serial.println("Placa 1");
        }
350     else
        {
352         if(endereco[0] == '1' && endereco[1] == '0')
            {
354             Serial.println("Placa 2");
```

```
        }
356         else Serial.println("Placa 3");
    }
358
    Serial.print("Mensagem enviada: ");
360
    for(a=0; a<pacote_linha; a++)
362 {
        Serial.write(pacote_asc2[a]);
364 }
    Serial.println();
366 }
//-------------------------------------------//
368 // Bloco 8: Responsavel por enviar a mensagem
void mensg(char endereco[4], int pacote_asc2[8])
370 {
    int destino=0;
372 String endereco_dest="";
    String endereco_origem = "";
374
    if(endereco[0] == '0' && endereco[1] == '1')
376 {
        endereco_origem = "01";
378 }
    else
380 {
        if(endereco[0] == '1' && endereco[1] == '0')
382     {
            endereco_origem = "10";
384     }
        else endereco_origem = "11";
386 }

388 if(endereco[2] == '0' && endereco[3] == '1')
    {
390     destino=1;
        endereco_dest="01";
392 }
    else
394 {
        if(endereco[2] == '1' && endereco[3] == '0')
396     {
            destino=2;
398         endereco_dest="10";
        }
400     else
        {
```

```
402                 destino=3;
                    endereco_dest="11";
404             }
        }
406     for(int b=0; b<pacote_linha; b++)
        {
408         pacote_mensagem(endereco_dest,endereco_origem ,destino,b,
                pacote_asc2);
            delay(200);
410   }
}
412 //-----------------------------------------------//
// Bloco 9: Responsavel por montar cada pacote da mensagem a ser enviada
414 void pacote_mensagem(String endereco_dest ,String endereco_origem, int
    destino, int b, int pacote_asc2[8])
{
416     //Converte Char para Binario
        String stringOne =  String(pacote_asc2[b], BIN);
418
        while(stringOne.length() < 8)
420     {
            //Completa o vetor ASCII com os zeros iniciais
422         stringOne = 0 + stringOne;
        }
424
        //Pacote: Destino + Mensagem
426     stringOne = endereco_dest + stringOne;
        //Pacote: Endereco + Destino + Mensagem
428     stringOne = endereco_origem + stringOne;
        //Pacote: Sincronismo + Endereco + Destino + Mensagem
430     stringOne = "1010" + stringOne;
        //Pacote: Sincronismo + Endereco + Destino + Mensagem + StopBit
432     stringOne = stringOne + "1";

434     trasmissao(destino,16,stringOne);
}
436 //-----------------------------------------------//
// Bloco 10: Responsavel por converter o pacote ASCII em codificacao
    manchester
438 void trasmissao(int destino, int tamanho, String stringOne)
{
440     switch(destino)
        {
442         case 1:
                for(int i=0; i<=tamanho; i++)
444             {
                    if(stringOne[i] == '1')
```

```
446                 {
                        //pino 5 - LOW; Uso de registrador pra diminuir ciclos
                            de clock
448                     PORTD &= B11011111;
                        delay(50);
450                     //pino 5 - HIGH; Uso de registrador pra diminuir ciclos
                             de clock
                        PORTD |= B00100000;
452                     delay(50);
                    }
454                 else
                    {
456                     //pino 5 - HIGH; Uso de registrador pra diminuir ciclos
                             de clock
                        PORTD |= B00100000;
458                     delay(50);
                        //pino 5 - LOW; Uso de registrador pra diminuir ciclos
                            de clock
460                     PORTD &= B11011111;
                        delay(50);
462                 }
                }
464         break;

466         case 2:
                for(int i=0; i<=tamanho; i++)
468             {
                    if(stringOne[i] == '1')
                    {
470                     //pino 6 - LOW; Uso de registrador pra diminuir ciclos
                            de clock
                        PORTD &= B10111111;
472                     delay(50);
                        //pino 6 - HIGH; Uso de registrador pra diminuir ciclos
474                          de clock
                        PORTD |= B01000000;
476                     delay(50);
                    }
478                 else
                    {
480                     //pino 6 - HIGH; Uso de registrador pra diminuir ciclos
                             de clock
                        PORTD |= B01000000;
482                     delay(50);
                        //pino 6 - LOW; Uso de registrador pra diminuir ciclos
                            de clock
484                     PORTD &= B10111111;
```

```
                    delay(50);
                }
            }
        break;

        case 3:
            for(int i=0; i<=tamanho; i++)
            {
                if(stringOne[i] == '1')
                {
                    //pino 7 - LOW; Uso de registrador pra diminuir ciclos
                        de clock
                    PORTD &= B01111111;
                    delay(50);
                    //pino 7 - HIGH; Uso de registrador pra diminuir ciclos
                         de clock
                    PORTD |= B10000000;
                    delay(50);
                }
                else
                {
                    //pino 7 - HIGH; Uso de registrador pra diminuir ciclos
                         de clock
                    PORTD |= B10000000;
                    delay(50);
                    //pino 7 - LOW; Uso de registrador pra diminuir ciclos
                        de clock
                    PORTD &= B01111111;
                    delay(50);
                }
            }
        break;
    }
}
//-------------------------------------------------//
```