UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE ELETRICIDADE

ADENILSON LIMA DINIZ

ACIONAMENTO E MONITORAMENTO DE SISTEMAS EMBARCADOS VIA WEB

São Luís
2017

ADENILSON LIMA DINIZ

ACIONAMENTO E MONITORAMENTO DE SISTEMAS EMBARCADOS VIA WEB

Monografia apresentada ao curso de Engenharia Elétrica da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Luciano Buonocore

São Luís

2017

Diniz, Adenilson Lima.
ACIONAMENTO E MONITORAMENTO DE SISTEMAS EMBARCADOS VIA WEB / Adenilson Lima Diniz. - 2017.
118 f.

Orientador(a): Prof. Dr. Luciano Buonocore.
Monografia (Graduação) - Curso de Engenharia Elétrica, Universidade Federal do Maranhão, CCET, 2017.

1. Acionamento e Monitoração remota. 2. Redes Locais. 3. Sistemas Embarcados. 4. Topologia Cliente-Servidor. I. Buonocore, Prof. Dr. Luciano. II. Título.

ADENILSON LIMA DINIZ

ACIONAMENTO E MONITORAMENTO DE SISTEMAS EMBARCADOS VIA WEB

Monografia apresentada ao curso de Engenharia Elétrica da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Luciano Buonocore

Aprovado em: 20 / 07 / 2017

BANCA EXAMINADORA

Prof. Dr. Luciano Buonocore (Orientador)
Universidade Federal do Maranhão

Prof. Dr. José de Ribamar Braga Pinheiro Júnior (Examinador)
Universidade Federal do Maranhão

Prof. MSc. Marcos Tadeu Rezende de Araújo (Examinador)
Universidade Federal do Maranhão

AGRADECIMENTOS

Agradeço a minha família, pelo apoio incondicional e presença constante.

Agradeço a todos os Docentes da UFMA e em particular ao Professor Luciano pela orientação e amizade.

A todos os amigos que direta e indiretamente fizeram parte da minha formação.

RESUMO

O acionamento e a monitoramento de sistemas embarcados a distância são uma realidade no mundo atual nas mais diversas áreas, como produção, serviços e também educação. O ensino à distância se beneficia dessa tecnologia de ação e monitoração remota, principalmente na realização de experimentos com sistemas físicos em diversos laboratórios espalhados pelo país. A proposta implementada neste trabalho de monografia, ainda que esteja restrita ao controle e monitoração de sistemas em uma rede local, usando a topologia cliente-servidor, permitiu o desenvolvimento de programas modulares para o ensino nos laboratórios locais de universidades, em especial ao LRC (Laboratório de Robótica Móvel e Comunicação Sem Fio) existente no Curso de Engenharia Elétrica da UFMA. O servidor *web* escolhido foi o Apache que é disponibilizado gratuitamente para uso em diversos sistemas operacionais que, neste trabalho, é o Linux. Após a instalação e configuração do servidor, foram feitas as programações das páginas que executam na máquina cliente (linguagem HTML) e dos códigos que permitem as comunicações entre o servidor e o sistema embarcado nos quatro experimentos implementados (linguagem PHP). Para a validação dos experimentos implementados, foram feitas as atividades de controle e monitoramento a partir de uma máquina cliente em duas situações extremas de tráfego na rede local do LRC. No primeiro tipo de teste, onde apenas o servidor estava conectado à rede local, foram feitos experimentos para obter o atraso entre o momento do pedido na página cliente (botão acionado) e a efetiva resposta observada visualmente. No segundo tipo de tese, o tráfego na rede local foi intensificado com a conexão de outras atividades na rede (*downloads* e acesso a sites *online*) e os mesmos tempos foram coletados. Os tempos foram baixos para ambos os testes, devido principalmente a boa velocidade apresentada na rede.

Palavras-chaves: Acionamento e monitoração remota; Topologia Cliente-Servidor; Rede Locais; Sistemas Embarcados.

ABSTRACT

Actuation and monitoring of remote embedded systems are a reality in today's world, in most diverse areas, such as production, services and education. Distance learning benefits from this technology of remote action and monitoring, mainly in conducting experiments with physical systems in laboratories scattered throughout the country. The proposal implemented in this work of monograph, although restricted to the control and monitoring of systems in a local network, using the client-server topology, allowed the development of modular programs for teaching in local university laboratories, especially to the LRC (Laboratory of Mobile Robotics and Wireless Communication) existing in the Electrical Engineering course of UFMA. The web server chosen was Apache, which is available free for use in various operating systems, which in this work is Linux. After the installation and configuration of the server, the programming of the pages running on the client machine (HTML language) and the codes that allow the communication between the server and the embedded system in the four implemented experiments (PHP language) were made. For the validation of the implemented experiments, control and monitoring actions were performed from a client machine in two extreme traffic situations in the LRC's local network. In the first type of test, where only the server was connected to the local network, experiments were performed to obtain the delay between the request time on the client page (button pressed) and the effective response observed visually. In the second type of test, traffic on the local network was intensified by connecting others network activities (downloads and access to online sites) and the same times were collected. For both tests the measured times were low, mainly due to a good network performance.

Keywords: Remote actuation and monitoring; Client-server Topology; Local Area Network; Embedded systems.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

A/D	– *Analógico/Digital*
ABS	– *Anti-lock Braking System*
ANSI	– *American National Standards Institute*
ASP	– *Active Server Pages*
AURESIDE	– *Associação Brasileira de Automação Residencial e Predial*
CPU	– *Central Processing Unit*
CC	– *Corrente Contínua*
CSS	– *Cascading Style Sheets*
DIP	– *Dual In-Line Package*
E2PROM	– *Electrically Erasable Programmable Read-Only Memory*
EDA	– *Electronic Design Atomation*
FPGA	– *Field Programmable Gate Array*
GND	– *Terra ou sinal de referência*
HTML	– *HyperText Markup Language*
I/O	– *Input/Output*
IBM	– *Internacional Business Machines*
LRC	– *Laboratório de Robótica Móvel e Comunicação Sem Fio*
LAN	– *Local Area Network*
NCSA	– *National Center for Supercomputing Applications*
P2P	– *Pee-to-Peer*
PWM	– *Pulse-Width Modulation*
RAM	– *Random Access Memory*
RISC	– *Reduced Instruction Set Computer*
ROM	– *Read-only Memory*
SO	– *Sistema Operacional*
SOC	– *Systems On Chip*
SSID	– *Service Set Identifier*
USB	– *Universal Serial Bus*
WWW	– *World Wide Web*

SUMÁRIO

1. INTRODUÇÃO

Uma das principais tendências atuais do ensino, associada ao contexto do ensino à distância, é o uso de recursos físicos de laboratórios que podem ser acessados remotamente (sem a presença física do aluno), facilitando o acesso às tecnologias da internet disponíveis. A realização de experimentos à distância está contemplada na proposta de uma federação de WebLabs coorporativos, onde cada WebLab da federação possui seus próprios recursos, disponibilizando-os para que outros centros cadastrados possam realizar seus experimentos de forma agendada (COELHO *et al.*, 2009).

Pode-se pensar no uso de redes locais para implementar experimentos remotamente e, ainda que apresente uma perspectiva mais restrita para o acesso no controle e monitoramento de dispositivos por estar localizado no âmbito de uma rede local, mostra-se um bom instrumento didático. Para isso, faz-se uso da topologia cliente-servidor, onde através de uma página cliente o usuário realiza pela rede local o controle e a monitoração de um sistema embarcado. A proposta contida neste trabalho de monografia vai ao encontro da realização de ações de controle e monitoração em quatro tipo de experimentos, todos fazendo uso do sistema embarcado Arduino, através de uma rede local no laboratório LRC (Laboratório de Robótica Móvel e Comunicação Sem Fio). Em três desses experimentos apenas se realiza o controle do sistema embarcado e no quarto faz-se tanto o controle quanto a monitoração de informação retornada pelo sistema embarcado à página cliente.

A estruturação proposta na topologia cliente-servidor possibilitou o desenvolvimento e estruturação de um servidor Apache com o objetivo de hospedar as páginas dos experimentos (programadas em HTML), bem como a programação da comunicação tanto com a máquina cliente quanto com o sistema embarcado (programado em PHP). O desenvolvimento produzido no trabalho permitiu a elaboração modular de *softwares*, facilmente exportáveis para outros experimentos que, inclusive, possam envolver outros tipos de sistemas embarcados, como o Raspberry Pi.

Dessa forma, o uso da proposta contida neste trabalho de monografia poderá beneficiar os laboratórios do curso da Engenharia Elétrica da Universidade Federal do Maranhão, independente do sistema embarcado a ser controlado e monitorado.

1.1 Objetivos

O objetivo geral deste trabalho de monografia foi desenvolver uma proposta de topologia cliente-servidor, operando em uma rede local, para controle e monitoramento de sistemas embarcados organizados em experimentos de laboratório.

Como objetivos específicos, podem-se destacar:

a) Implementação de um servidor Apache para troca de informações com a máquina cliente e com o sistema embarcado (detalhes Apêndice A);
b) Desenvolvimento de páginas clientes (linguagem HTML), programas de comunicação com a máquina cliente e com o sistema embarcado (linguagem PHP), além de programação no sistema embarcado Arduino (linguagem C) para cada um dos quatro experimentos. Os códigos desses programas, em linguagens HTML e PHP são disponibilizados no Apêndices B, enquanto o da linguagem C no Apêndice C;
c) Validar a estrutura proposta em quatro tipos distintos de experimentos, sendo que em um deles corresponde ao controle de movimento de um robô móvel e a leitura do valor de distância do robô a um obstáculo à sua frente por meio de um sonar; e
d) Testes de atrasos de resposta entre o pedido do comando ou de monitoração realizado na página cliente e sua efetiva ação, considerando-se a rede apenas sendo usada pelo servidor proposto e com uma carga de tráfego implementada com outras três máquinas na mesma rede fazendo download de vídeos no site < https://www.youtube.com/?gl=BR&hl=pt>.

1.2 Motivação

Existem algumas motivações que foram importantes na escolha do tema deste trabalho de monografia. A primeira foi o estudo e a implementação de uma proposta de topologia cliente-servidor em uma rede local de laboratório para o controle e a monitoração de sistema embarcado Arduino, o que requereu o aprendizado de instalação e configuração de um servidor Apache. Nesse processo, também foram necessários o aprendizado de programação de

linguagens HTML (elaboração de páginas clientes) e PHP (códigos de comunicação servidor-cliente e servidor-Arduino).

Um segundo aspecto para o desenvolvimento neste trabalho é a possibilidade de escalabilidade a outros experimentos na mesma topologia implementada que possa fazer uso de outros tipos de sistemas embarcados mais robustos que o empregado (Arduino), como são os casos do Raspberry Pi e BeagleBone.

Finamente, como terceiro elemento motivador, existe o aprendizado que foi adquirido na implementação destes experimentos usando a topologia cliente-servidor, não ministrada durante a graduação do curso de Engenharia Elétrica. Constitui, portanto, um acréscimo em conhecimento na graduação que tem grande aplicações em ensino de laboratório local, porém com possibilidades de extrapolar para aplicações remotas a distância, via internet.

1.3 Organização do trabalho

O trabalho está organizado em 6 capítulos, incluindo este de introdução do trabalho de monografia.

No Capítulo 2 é feita a fundamentação teórica dos temas essenciais utilizados no trabalho, no caso a descrição das tecnologias de rede disponíveis, tipos de servidores empregados e uma visão sobre sistemas embarcados, com detalhamento na placa Arduino Uno usada nos experimentos.

No Capítulo 3 é feita a descrição da topologia cliente-servidor implementada, cujo detalhamento do servidor Apache empregado é disponibilizado no Apêndice A. Ainda neste capitulo é mostrada a página inicial do sistema, feita em programação ´html'. Detalhes da programação das páginas cliente foram colocados no Apêndice B.

No Capítulo 4 são apresentados os quatro experimentos implementados e foram detalhadas as comunicações entre o servidor e esses experimentos em códigos programados em linguagem PHP (servidor) e C (Arduino), sendo a programação em PHP feita na forma de fluxograma. Os códigos detalhados e comentados em PHP e C foram colocados nos Apêndices B e C, respectivamente. Além disso, são apresentadas as páginas de acesso aos experimentos (programadas em 'html' e hospedadas no servidor) para realizar o controle e a monitoração do sistema embaraçado através da máquina cliente.

No capítulo 5 foram feitos testes e apresentados os resultados obtidos com o sistema proposto implementado, operando na rede local do LRC em duas situações extremas de tráfego na rede local do LRC, somente com o servidor conectado e com 3 outras máquinas fazendo *downloads* de vídeos em sites de alta velocidade.

Finalmente, no Capítulo 6 apresentam-se as conclusões e propõem-se possíveis trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Existe uma grande rede de computadores que se encontram interligados com alcance global, conhecida como internet. A idealização de concatenar computadores deu início em meados da década de 60, pois já preexistia o desejo de interligar computadores nos Estados Unidos (FOROUZAN, 2010).

A idealização de um meio de comunicação de informações foi motivada pela guerra fria, polarizada entre os Estados Unidos e a União Soviética. O governo norte-americano necessitava que as transferências de informações relacionadas às estratégias e às comunicações sigilosas ocorressem com maior segurança. Como consequência foi criado o projeto da ARPANET (*Advanced Research Projects Agency Network*), considerada atualmente como a mãe da rede de computadores chamada de Internet[1].

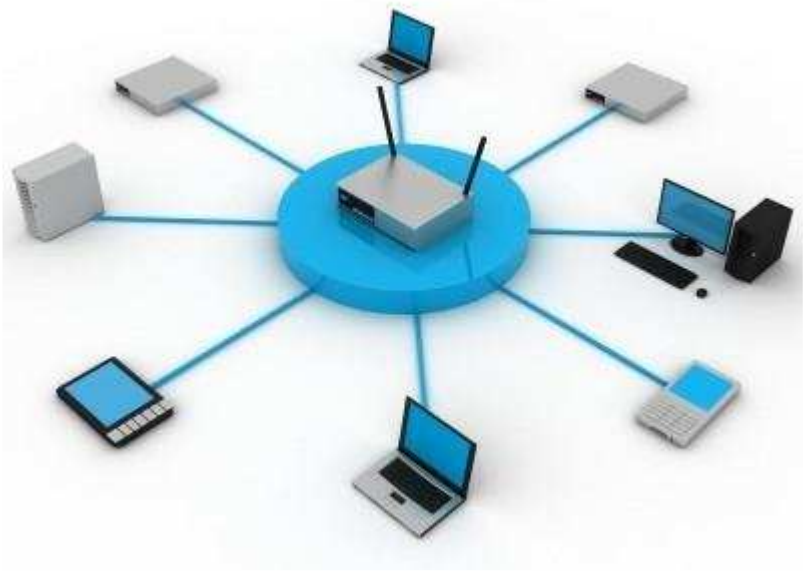
O projeto em questão tinha como função principal a realização de uma interconexão dos computadores existentes através de um sistema denominado como comutação. Nesse tipo de esquema, as informações eram divididas em pacotes que comportavam fragmentos de dados. Cada computador envolvido na transferência da informação encaminhava seu fragmento, sendo toda a informação reconstituída no computador que realizava a solicitação das informações (FRANCISCATTO, 2014). A Figura 1 ilustra essa técnica empregada na rede ARPNET.

Com o passar do tempo, o número de pessoas que se conectavam com o sistema ARPANET foi crescendo rapidamente, o que tornou essa tecnologia insuficiente para atender as necessidades essenciais dos usuários. Como consequência, o uso da ARPNET tornou-se seu uso menos eficiente para os propósitos inicialmente planejados. Por esse motivo foi necessário que se criasse um novo sistema, com uma melhor padronização ao nível de estruturação e manutenção. Esse novo projeto de comunicação à distância implementado foi denominado de Protocolo de Internet (*Internet Protocol*), também chamado de TCP/IP[2], tendo a mesma função do sistema ARPNET, porém conseguindo atender melhor a necessidade de tráfego de grande número de informações pela rede (FRANCISCATTO, 2014).

No período entre 1970 e 1973, com o advento da ARPENET, foi possível a criação de uma rede para interligação entre universidades, instituições militares e empresas. Os computadores que armazenavam informações para acesso pelos usuários nessa época eram os

[1] Site: https://pt.wikipedia.org/wiki/Hist%C3%B3ria_da_Internet
[2] Site: https://pt.wikipedia.org/wiki/TCP/IP

Figura 1 – Exemplo de conexão de computadores na rede ARPANET.



Fonte: https://www.tecmundo.com.br/982-o-que-e-cliente-servidor-.htm

mainframes, caracterizados por um poder de processamento baixo e com preços elevados (FRANCISCATTO, 2014).

As redes de computadores apesar da evolução e crescente propagação, mantém seu objetivo primordial: compartilhar recursos (tanto de hardware como software) e propiciar a troca de informações (MORIMOTO, 2005).

Diversos tipos de tecnologias de LAN[3] (*Local Area Network*) têm sido desenvolvidas, sendo importante conhecer como as tecnologias específicas são semelhantes e como diferem. Para o entendimento das semelhanças, cada rede é classificada em uma categoria de acordo com sua topologia ou forma geral (COMER, 2007).

2.1 Topologia Física de Rede

As redes de computadores, geralmente, são classificadas de acordo com sua disposição geográfica e hierarquia (FRANCISCATTO, 2014).

[3] Site: https://pt.wikipedia.org/wiki/LAN

As mais usadas e conhecidas são as Topologias de formato ponto-a-ponto, estrela, barramento e anel. Ainda que sejam as mais citadas, existem outros tipos de arranjo físico de redes.

A Topologia denominada ponto-a-ponto é a mais simples. Através de um meio de transmissão cabo ou *wifi* se estabelece uma comunicação entre dois computadores a partir desse arranjo simplificado, é possível a formação de novas topologias, englobando outros nós além dos dois envolvidos em seu arranjo físico (FRANCISCATTO, 2014).

Uma rede ponto-a-ponto normalmente é utilizada em pequenas redes. Neste tipo de rede os computadores trocam informações entre si, compartilhando arquivos e recursos (FRANCISCATTO, 2014).

Essa topologia será detalhada na seção mais adiante. A figura 2 ilustra a conexão na topologia ponto-a-ponto.

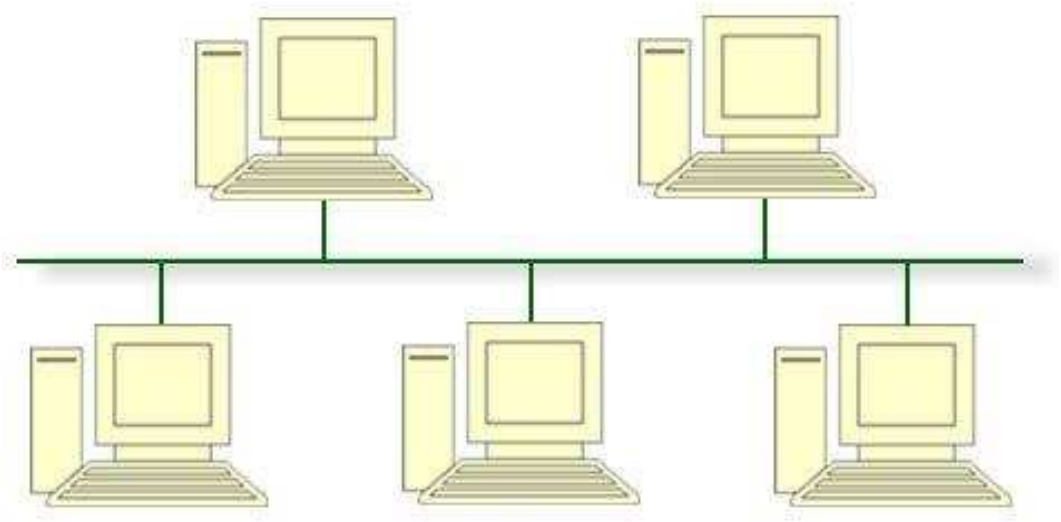
Figura 2 – Exemplo de conexão de computadores na rede ponto-a-ponto.

Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

A topologia conhecida por barramento é bastante usada e apresenta facilidade de expansão. Todos os nós estão interligados a um barramento físico que é compartilhado por todos os nós, podendo o gerenciamento tornar-se centralizada ou distribuída. O meio de difusão utilizado nesta topologia é o cabo coaxial. A Figura 03 mostra a conexão dos nós em uma rede com topologia barramento (MARTINS, 2002).

Diferente da topologia em anel, topologias em barra podem empregar interfaces passivas, nas quais falhas não causam a parada total do sistema. A confiabilidade deste tipo de topologia vai depender em muito da estratégia de controle. O controle centralizado oferece os mesmos problemas de confiabilidade de uma rede em estrela, com atenuante de que, nesta topologia, a redundância de um nó pode ser outro nó comum da rede. Mecanismos de controle descentralizados semelhantes aos empregados na topologia em anel podem também ser

Figura 3 – Exemplo de conexão de computadores na rede Barramento.



Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

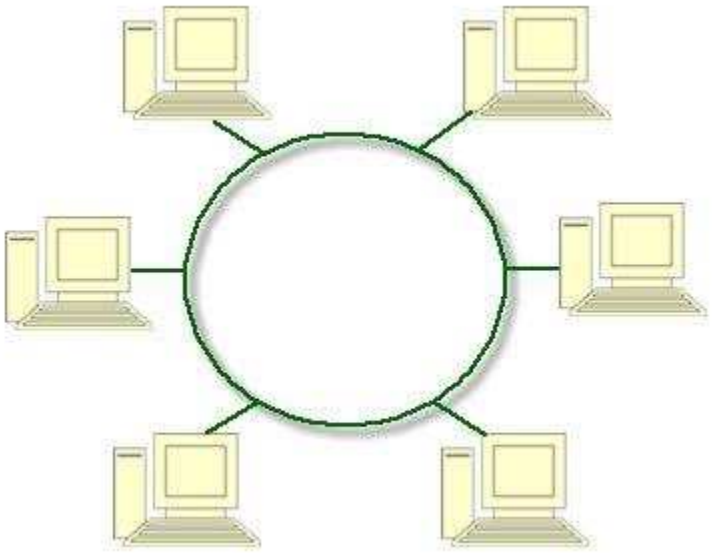
empregados neste tipo de topologia, acarretando os mesmos problemas quanto à detecção da perda do controle e sua recriação (MARTINS, 2002).

A topologia em anel faz uso em ligações ponto-a-ponto circulando por nós da rede dispostos fisicamente como um anel, cuja arquitetura física é ilustrada na Figura 4. Nessa conexão é defendido um nó central para gerenciar a comunicação entre os outros nós existentes na rede. É esta máquina que vai definir a velocidade de transferência, como também realizar a conversão dos sinais transmitidos por diversos tipos de protocolos (MARTINS, 2002).

Redes em anel são capazes de transmitir e receber dados em qualquer direção. As configurações mais usuais, no entanto, são unidirecionais para simplificar o projeto dos repetidores mais simples e tornar menos sofisticados os protocolos de comunicação que asseguram a entrega da mensagem corretamente. Os repetidores são em geral projetados de forma a transmitir e receber dados simultaneamente, diminuindo assim o retardo de transmissão (MARTINS, 2002).

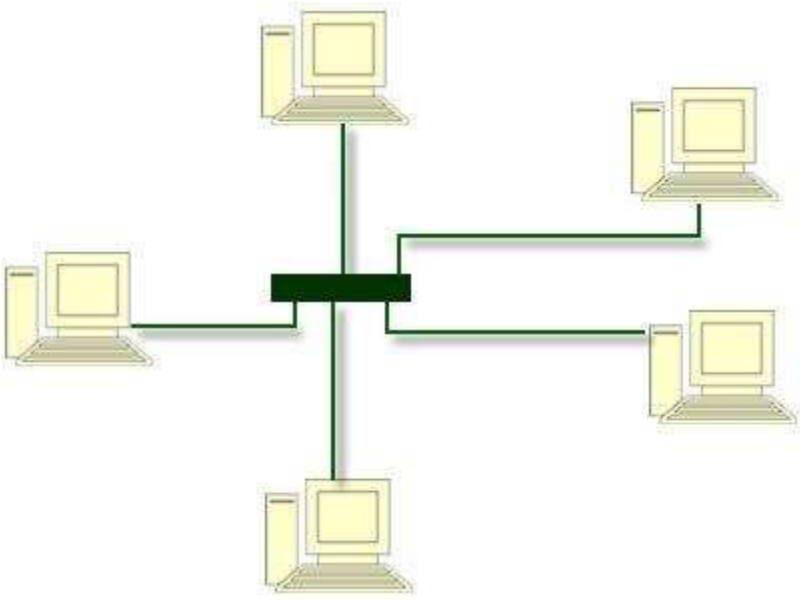
No tipo de rede com topologia em estrela, cada nó se encontra conectado a um computador central denominado ***hub***, o qual faz a intermediação entre dois nós que necessitam se comunicar pela rede, conforme ilustra a Figura 5 (MARTINS, 2002).

Figura 4 – Exemplo de conexão de computadores na rede Anel.



Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

Figura 5 – Exemplo de conexão de computadores na rede Estrela.

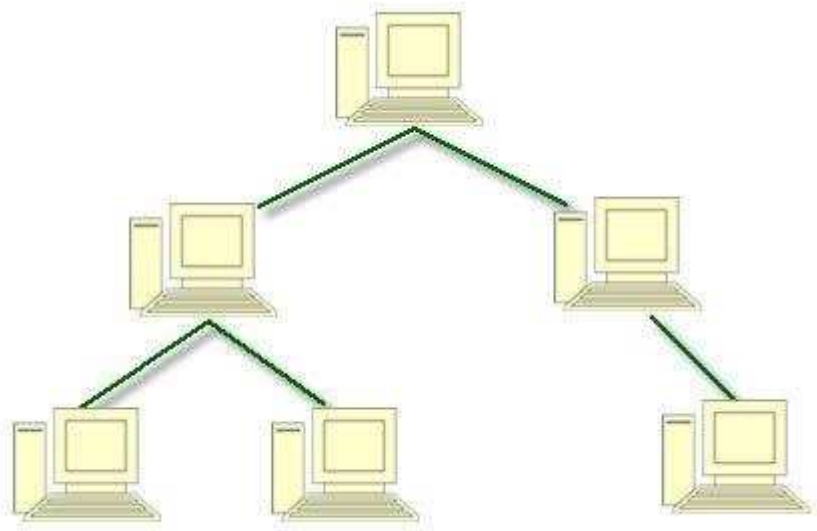


Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

A Topologia em árvore é basicamente uma junção de diversas topologias barras e estrelas interconectadas, conforme ilustra o exemplo de conexão física na Figura 6. Nessa arquitetura de rede existe uma barra principal onde os demais nós de menor porte se vinculam.

A conexão física da arquitetura híbrida[4] é baseada numa estrutura hierárquica de diversas redes e sub-redes. Há um ou mais concentradores (nós específicos) que ligam cada rede local e há outro concentrador que interliga todos os demais concentradores. Esta topologia favorece a preservação do sistema e permite detectar com mais agilidade o distúrbio ocorrido na comunicação, caso que demandará atividade de manutenção (MARTINS, 2002).

Figura 6 – Exemplo de conexão de computadores na rede Árvore.



Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

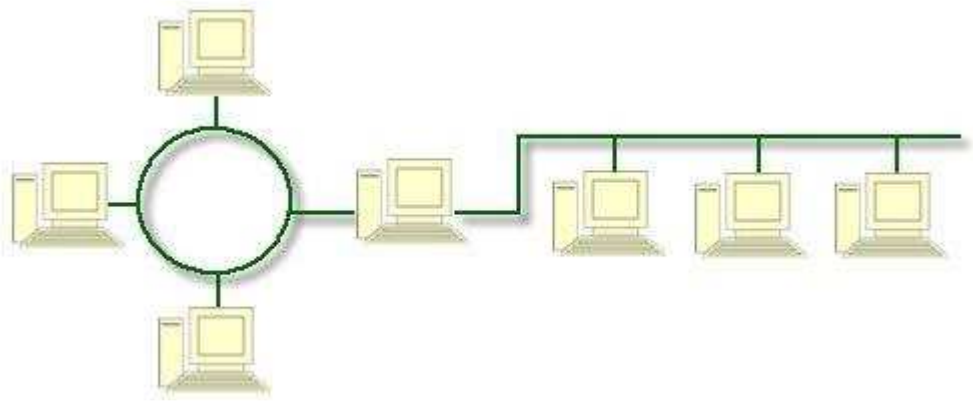
A topologia híbrida (Figura 7) é, na verdade, uma série de barramentos interconectados. Geralmente existe um barramento central onde outros ramos menores se conectam. Esta ligação é realizada através de *switches* e as conexões das estações realizadas do mesmo modo que no sistema de barramento padrão (MARTINS, 2002).

2.2 Arquitetura Cliente/Servidor via Web

A internet se tornou a maior rede de computadores interligados. Então, surgiu a necessidade de dedicar para alguns computadores a tarefa de prover os serviços disponíveis na rede, fazendo com que a grande maioria somente tivesse a função de acessar estes serviços. Foi

[4] Site: https://www.ibm.com/developerworks/br/library/mw-1606-clark-trs/index.html

Figura 7 – Exemplo de conexão de computadores na rede Híbrida.



Fonte: http://www.diegomacedo.com.br/topologias-de-rede-de-computadores/

dada para os computadores que oferecem os recursos a denominação de servidores e para os computadores que só tinham a função de acesso a essas informações o nome de clientes (FRANCISCATTO, 2014).

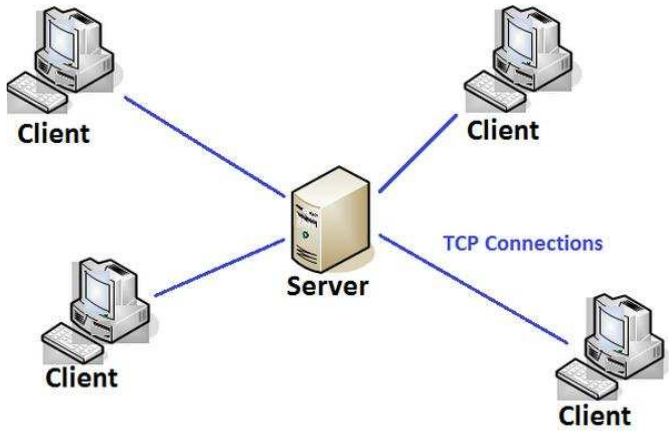
Uma rede de computadores com estrutura ou topologia cliente-servidor possui um ou mais servidores, responsáveis por prover serviços de rede aos demais computadores conectados a ele que são chamados clientes. Cada cliente (consumidor de informação) que deseja acessar um determinado serviço ou recurso, faz essa solicitação ao servidor da rede, por isso o nome cliente-servidor (FRANCISCATTO, 2014).

O servidor *web* é um equipamento que tem como responsabilidade principal a disponibilização de informações, nos mais diversos formatos (fotos, páginas, vídeos, áudios, animações gráficas, etc.) de maneira a encaminhá-las ao usuário do computador cliente. Essa máquina também pode funcionar como recebedor de dados pedidos pelos clientes, transformando ou convertendo as informações e retornando-as para que o cliente os consultem e faça outros acessos, caso necessário (ALBUQUERQUE, 2001).

Na arquitetura cliente/servidor, o computador cliente envia uma solicitação para o servidor através da conexão de rede, que é então processada pelo servidor que retorna a resposta para o cliente (ALBUQUERQUE, 2001).

A topologia Cliente-Servidor (Figura 8) é uma formação de rede que tem em sua estrutura básica microcomputadores, uns definidos como servidores e outros com a função específica de clientes, sendo a grande maioria pertencente a esse segundo grupo. Essa

Figura 8 – Arquitetura de Aplicação do tipo Cliente e Servidor.



Fonte: http://www.redesprogressiva.net/Arquiteturas-de-Aplicacao-em-Redes-Cliente-Servidor.html

comunicabilidade recebe a definição em função do tipo de serviço que executa, ou seja, solicita ou presta serviço, por intermédio de um processo iniciado no computador do cliente, através de um determinado navegador, o qual interage com o servidor onde estão localizadas as informações, normalmente organizadas em banco de dados (ROCHA, 2002).

Dessa forma, estabelecem-se as definições:

a) Cliente, a máquina que solicita a informação ou serviço; e
b) Servidor, o computador que dispõe a informação ou serviço.

Aplicações como a simples troca de e-mail, navegar na internet em hipertextos ou até mesmo acessar a um banco de dados, são recursos desenvolvidos fundamentado na topologia Cliente-Servidor. Como exemplo, um navegador *web* em execução no equipamento do usuário, busca as informações nas bases de dados que se encontram guardadas em uma máquina servidor *web* na internet (BOCHENSKI, 1994).

A topologia Cliente-Servidor foi implementada como uma das necessidades principais no que tange as áreas de computação de rede. Vários softwares e sistemas de negócios, elaborados hoje, fazem uso desse modelo. O termo igualmente tem sido empregado para diferenciar da computação distribuída por diversos microcomputadores da computação centralizada em mainframe (MARTINS,2002)

A máquina computacional denominada de mainframe constitui um grande equipamento, contendo uma grande quantidade de recursos (processadores, memória principal, armazenamento em massa), capaz de operar com grande quantidade de informações e em alta

velocidade, acessado de forma direta ou por rede5. Como exemplos de servidores mantidos em mainframes, existem os bancos e as universidades, com a vantagem de manter a compatibilidade do uso tanto de programas mais antigos com os mais recentes. A Figura 9 mostra o mais recente modelo de mainframe lançado pela empresa IBM no mercado, denominado de z136, com capacidade de lidar com 2,5 bilhões de transações por dia, com um custo de desenvolvimento de U$ 1 bilhão.

Figura 9 – Foto do *mainframe* IBM z13.

Fonte: https://tecnoblog.net/172796/mainframe-ibm-z13/

Cada campo do *browser* do cliente pode remeter solicitações a diversos servidores. Os servidores, por sua vez, podem acatar essas solicitações, processá-las e devolver os resultados solicitados pelo cliente. Apesar de que o conceito aqui exposto possa ser aplicado por uma variedade de razões e para vários tipos de aplicações, a estrutura fundamental permanece a mesma (MARTINS,2002).

2.3 Arquitetura *Peer-To-Peer* (P2P)

Os sistemas peer-to-peer, abreviado pela sigla P2P, representam um modelo para a construção de sistemas e de aplicativos distribuídos onde dados e recursos computacionais são provenientes da colaboração uniforme de diversos hosts7 através da Internet. As máquinas host são computadores da rede com capacidade de fornecer dados, recursos e serviços a usuários ou

[5] Site: https://pt.wikipedia.org/wiki/Mainframe
[6] Site: https://tecnoblog.net/172796/mainframe-ibm-z13/
[7] Site: https://pt.wikipedia.org/wiki/Host

outros hosts da rede. Tais sistemas são desenvolvidos para permitir o compartilhamento de recursos em larga escala e sem a necessidade de servidores e infraestruturas associadas para gerenciamento. Seu sucesso é consequência do rápido crescimento da Internet e aumento no número de usuários exigindo acesso a recursos compartilhados. A popularidade destes sistemas ocorreu, principalmente, devido às aplicações utilizadas para compartilhamento de arquivos (REZENDE, 2009).

Em Androutsellis-Theotokis e Spinellis (2004) são apresentadas duas características que definem os sistemas P2P:

- Compartilhamento de recursos computacionais de forma direta entre as partes envolvidas, sem intermédio de servidores centrais. Entretanto, aceita-se a existência de servidores centrais para a execução de tarefas específicas, como a localização de *peers* e autenticação de usuários;

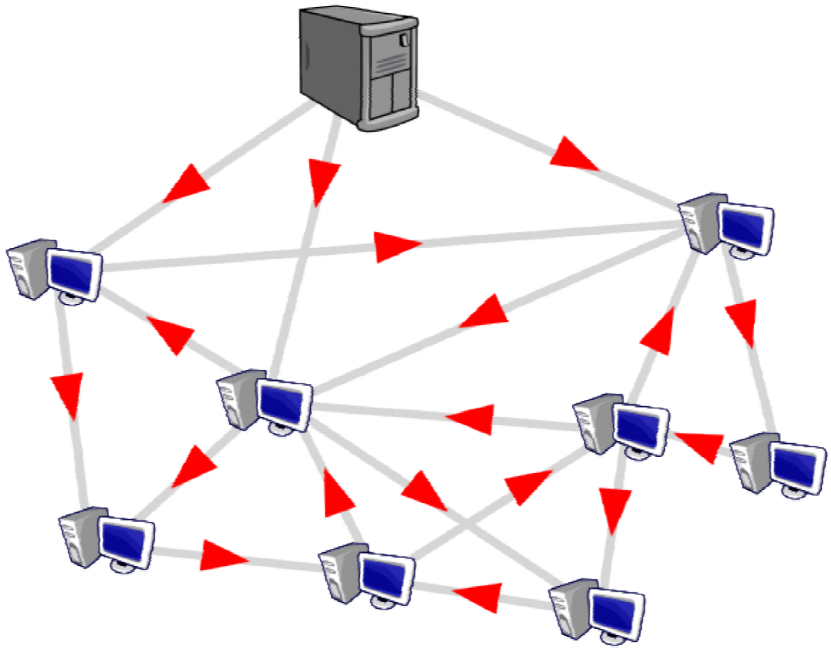
- Habilidade para tratar instabilidade e variação de conectividade, adaptando-se de forma automática a falhas nas conexões de rede e computadores, bem como comportamento transitório dos *peers*.

Sistemas P2P são sistemas distribuídos, constituídos por *peers* interconectados, habilitados para a auto-organização em uma topologia de rede, com o propósito de compartilhar recursos como conteúdo, ciclos de CPU, armazenamento e largura de banda. Além disso, esses sistemas possuem a capacidade de se adaptar à falhas e acomodar-se à variação no número de *peers* participantes, mantendo aceitáveis os níveis de conectividade e desempenho, sem a necessidade de intermediação ou suporte de um servidor global centralizado (ANDROUTSELLIS-THEOTOKIS & SPINELLIS, 2004). A Figura 10 ilustra um esquema de conexão ponto-a-ponto.

2.4 Projeto de Páginas Clientes

O advento e principalmente o sucesso da topologia Cliente-Servidor teve um impulso através da computação em baixo desempenho, através da incorporação dos microcomputadores aos ambientes corporativos. Concomitantemente, aconteceu o advento e emprego em grande

Figura 10 – Arquitetura de Aplicação do tipo peer-to-peer.



Fonte: http://www.clubedohardware.com.br/forums/topic/364213-p2p-x-torrent/

escala das redes locais de computadores, LAN (*Local Area Network*), em contraposição ao modelo concentrado em computação de grande porte executando em máquinas *mainframes.*

A tecnologia de comunicação das LANs pode ser via conexão sem fio, fibra ótica ou através de um cabo de cobre, as quais todas as máquinas estão conectadas. As LANs tradicionais funcionam em velocidade de 10 a 100 Mbps, têm baixo retardo (na ordem de microssegundos ou nanosegundos) e cometem pouquíssimos erros de comunicação. As LANs mais modernas operam em até 10 Gbps (TANENBAUM, 2003).

Diversos tipos de tecnologias de LAN têm sido implementados, sendo importante conhecer como as tecnologias específicas são semelhantes e como diferem. Para ajudar a entender as semelhanças, cada rede é classificada em uma categoria de acordo com sua topologia ou forma geral (COMER, 2007).

2.5 Tipos de Servidores

O servidor representa a camada número dois no modelo cliente/servidor. Ele é o conjunto *hardware*/*software* (microcomputador, *mainframe*) que oferece um conjunto de

serviços necessários aos mais variados tipos de solicitações realizados pelo cliente (ROCHA, 2002).

O servidor é basicamente definido como um *equipamento* computacional que em sua função primordial faz a centralização e o fornecimento de dados a uma rede de computadores. O servidor é uma máquina que tende a possuir uma potência de armazenamento e processamento bem mais poderoso que de uma máquina comum (cliente).

Para que o servidor funcione de acordo com o que se deseja, existem algumas observações ou características diferenciadas. A principal é o sistema operacional (SO)[8] que é instalado na máquina, o qual deve ser apropriado para as funções que o servidor irá desempenhar. Dessa forma, o SO representa um sistema especializado em várias das funções que executa, sendo de uso apropriado para o funcionamento de diversos aplicativos ao mesmo tempo (e-mails, base de dados, páginas 'html', etc.)

Um servidor é uma máquina que trabalha de forma intensiva, sendo disponibilizadas 24 horas por dia, exceto para manutenção. Essa condição é válida para os mais diversos tipos de servidores (MARIMOTO, 2006).

Os servidores são responsáveis por manter aplicativos necessários para os diversos serviços que fornece para serem acessados pelos clientes a qualquer momento, através da rede *internet*. Dado a importância do servidor em uma ou mais tarefas que executa, é indispensável comentar sobre as características de alguns que apresentam funções distintas (CARVALHO, 2006).

2.5.1 Servidor Apache

O servidor *web* Apache (*Apache HTTP Server*) é um *software* livre mais utilizado do mundo. Foi criado em 1995 por Rob McCool[9], então funcionário do NCSA (*National Center for Supercomputing Applications*). Em uma pesquisa realizada em dezembro de 2007, foi constatado que a utilização do Apache representa cerca de 47.20% dos servidores ativos no mundo. Em maio de 2010, o Apache serviu aproximadamente 54,68% de todos os sites e mais

[8] Site: https://www.gcfaprendelivre.org/tecnologia/curso/informatica_basica/sistemas_operacionais/1.do
[9] Site: https://pt.wikipedia.org/wiki/Rob_McCool

de 66% dos milhões de sites mais movimentados. É a principal tecnologia produzida pela empresa *Apache Software Foundation*, responsável por mais de uma dezena de projetos envolvendo tecnologias de comunicação via *web*, processamento de dados e execução de aplicativos distribuídos (ALBUQUERQUE, 2001).

Assim como todos os servidores do tipo aqui mencionado, o Apache é incumbido por tornar disponível as páginas e todos os demais recursos que o usuário necessita acessar. Atividades como o envio de e-mails, compras online, mensagens, além de diversas outras funções se tornam possíveis de serem executadas. O que precisa ser destacado, é que no Apache, a versão é desenvolvida em gratuidade (código *free*), ou seja, passível de modificações e alterações no seu código, porém não podendo ser comercializados tais modificações (ALBUQUERQUE, 2001).

O nome do servidor faz referência a tribo americana Apache, no que diz respeito a sua capacidade de resistência que as comunidades de *softwares* autônomos impuseram em relação aos desenvolvedores privados de diversos setores e empresas. Além disso, o nome possui vinculação direta a estabilidade que o servidor possui, além da sua grande diversidade de ferramentas e recursos disponibilizados, que são capazes de atender qualquer tipo de solicitação de clientes (ALBUQUERQUE, 2001).

O Apache é escrito na linguagem C ANSI, sendo um *software open source d*istribuído no site **www.apache.org** pela empresa *The Apache Software Foundation"*, onde diversos voluntários contribuem para o aprimoramento do projeto Apache[10].

Em referências ao *hardware* necessário para o devido suporte ao servidor, tem uma variação de capacidade de processamento, armazenamento e periféricos que dependem da finalidade para a qual o Apache está sendo empregado. Por exemplo, uma máquina com processador Pentium e 64 MB de memória RAM já é capaz de garantir sua execução sem maiores dificuldades em uma rede corporativa de pequeno porte (ALBUQUERQUE, 2001).

O Apache em função de seu desempenho otimizado e executar em SO Linux, foi o escolhido para implementar a topologia cliente-servidor para acesso a sistemas embarcados Arduino desenvolvidos neste trabalho de monografia.

[10] Site: http://www.apache.org/LICENSE.txt

2.5.2 Servidor IIS

O IIS (*Internet Information Services*) é um servidor *web* desenvolvido pela Microsoft. A versão mais atual é o IIS 8 e somente é executável nos sistemas operacionais Windows 8 e Windows Server 2012.

IIS é um *software* pago, pois o servidor somente está disponível para instalação juntamente com o sistema operacional da Microsoft. Embora o servidor possua uma licença paga, o IIS é o segundo servidor *web* mais utilizado do mundo (NETCRAFT, 2014).

O IIS, ao contrário do seu concorrente, o Apache, é um servidor *web* que só pode ser instalado nos sistemas operacionais Windows. O servidor *web* IIS suporta uma grande variedade de linguagens, dentre as principais estão ASP, PHP e Java. Ao contrário do Apache, a linguagem ASP funciona muito bem no servidor IIS (LINHARES, 2014).

Segundo Côrte (2002): "[...] o Apache se constitui em um servidor modular, fazendo com que somente as mais básicas funcionalidades sejam disponibilizadas pelo núcleo do sistema". O que pode explicar um desempenho melhor em relação ao seu concorrente, o IIS.

2.5.3 Servidor NGINX

O NGINX se originou em 2002, com o objetivo de suportar o tráfego intenso de um serviço *web* Russo chamado Rambler[11], o qual recebia em torno de 500 milhões de acesos por dia em 2008. É usado em serviços *web* bem populares, tais como Wordpress, PHP, .NET, Instagram, Gamespot, Tumblr, Gomarketingblog, entre outros. Suas principais características são apresentar resposta rápida, ser computacionalmente leve e eficiente no consumo de recursos da plataforma de *hardware* onde está instalado (CANÊDO, 2015).

O NGINX é um servidor HTTP e Proxy[12] reverso que é amplamente utilizado no mundo. Um proxy reverso é um servidor de rede geralmente instalado para ficar na frente de um servidor *web* com a finalidade, por exemplo, de balancear a carga de um cluster de servidores *web*[13]. Em condições de igualdade, tem desempenho melhor do que o Apache pois não precisa

[11] Site: https://www.ramblers-webs.org.uk/
[12] Site: https://pt.wikipedia.org/wiki/Proxy
[13] Site: https://pt.wikipedia.org/wiki/Proxy_reverso

criar novos processos ou *threads*[14] para cada requisição. Também consome menos memória das máquinas em sua operação (CANÊDO, 2015).

Basicamente, o NGINX é responsável por cuidar das requisições de imagens e estáticos em geral, como por exemplo, Javascripts e CSS, bem como as requisições que precisam ser geradas dinamicamente, como as páginas que necessitam de códigos para serem executadas (CANÊDO, 2015).

Outro aspecto que ganha significante destaque é o desempenho do servidor *web* NGINX, visto que vários administradores e técnicos já realizaram testes comparativos destes dois servidores, principalmente relacionado à métrica do tempo de resposta e do uso de largura de banda. Muito provavelmente, quando o Apache foi pensando na década de 90, não foi considerado que um dia receberia centenas de milhares de requisições diárias. Algo em torno de 10.000 conexões simultâneas era algo improvável e que, atualmente, é um valor realista, dada a evolução tecnológica alcançada (NEDELCU, 2013).

2.6 Sistemas Embarcados

Existem diversas definições para designar o significado de sistemas embarcados, ou sistemas embutidos, embora a maioria converge para a mesma ideia básica[15]: é um sistema baseado em um microprocessador ou microcontrolador, de aplicação dedicada, realizando uma quantidade de tarefas pré-estabelecidas, sendo implementados em placas de tamanhos reduzidos e geralmente requerendo pouca energia para o seu funcionamento.

A gama de aplicação dos sistemas embarcados é bastante grande, indo de simples sistemas com um mínimo de complexidade, como são os casos dos fornos micro-ondas, até sistemas bem complexos que envolvem várias unidades de processamentos com diversos recursos de *hardware*, como são os casos dos sistemas existentes nas aeronaves comerciais.

A automação residencial, exemplos de aplicações embarcadas, está em crescimento contínuo e acelerado com os avanços da tecnologia. Sua finalidade é proporcionar aos seus utilizadores o conforto, antes não imaginado pelo fato de ser facilmente adaptado a qualquer utilidade doméstica. Desse modo, utiliza uma tecnologia modular (capacidade fácil de

[14] Site: https://www.tecmundo.com.br/9669-o-que-sao-threads-em-um-processador-.htm
[15] Site: http://www.petccufpb.com.br/wp-content/uploads/sistemas_embarcados.ppt

expansão) e flexível proporcionados pelos sistemas embarcados, onde o próprio habitante do lar define como será beneficiado com essa automação. Entre os principais acréscimos trazidos com a automação, pode-se citar: o conforto do ambiente, otimização do tempo causado pela diminuição das tarefas rotineiras e principalmente segurança nas tarefas executadas[16].

Segundo a AURESIDE (Associação Brasileira de Automação Residencial e Predial): "Soluções de automação estão evoluindo constantemente nos dias de hoje em termos de eficiência, capacidade e desempenho geral". Em uma pesquisa realizada no ano de 2015 em 6 países, incluindo o Brasil, mostrou já um número significativo em automação residencial, sendo que no Brasil o potencial chega a ser mais de 6 vezes o real instalado[17].

A questão de reação do sistema embarcado a um evento sob seu gerenciamento, pode defini-lo como sendo aplicado ao controle de processo com ou sem requisitos tempo real. Em sistemas que tem requisito tempo real, a reação a um evento do processo deve ser fornecida dentro de um tempo especificado sob pena de comprometer o adequado funcionamento do processo. Um exemplo bem simples desse tipo de sistema é o freio ABS (*Antilock Braking System*) presentes nos carros atuais, onde o bloqueio das rodas do carro é evitado pela ação conjunta de sensores e atuadores de frenagem de maneira a não permitir que o carro deslize[18]. O fato das rodas deslizarem sobre o piso de navegação do automóvel, pode causar acidentes. Nos sistemas que não possuem requisitos de controle em tempo real, não existe uma restrição na ação do sistema a um evento que ele controla, como por exemplo, em elevadores, onde o usuário pode aguardar a chegada do primeiro disponível, sem comprometer o funcionamento do sistema.

Por fim, a questão de o sistema ser baseado em microprocessadores (de um ou mais núcleos) ou microcontroladores, normalmente definem a capacidade de processamento, de armazenamento, recursos diversos e manipulação de informações. Os sistemas embarcados microprocessados consomem mais energia, porém possuem mais recursos de *hardware* e maior desempenho na execução de funções mais complexas que, em alguns casos, não podem ser solucionadas por sistemas microcontrolados. Sistema que requerem processamento paralelo, que requerem execução de programas em mais de um núcleo do processador, não podem ser

[16] Site: http://www.forumdaconstrucao.com.br/conteudo.php?a=11&Cod=980
[17] Site: http://www.aureside.org.br/_pdf/potencial_2015.pdf
[18] Site: http://www.usp.br/ldsv/wp-content/uploads/2014/10/Artigo_Prandy_Final.pdf

executados em sistemas microcontrolados, pelo fato desses serem de natureza monoprocessada.

A escolha do sistema embarcado depende do processo que se queira controlar, dos requisitos de tempo, dos recursos de *hardware* necessários (memória, canais A/D, padrões de barramento, número de pinos digitais etc.) e do tipo de ambiente que se deseja implementar na programação (*monothread* ou *multithread*). Um sistema embarcado microprocessado de baixo custo bastante utilizado na atualidade é o Raspberry Pi. Sistemas embarcados microcontrolado de baixo custo e com *hardware* bem mais modesto, porém muito útil em diversas aplicações pertencem a família Arduino[19], o qual será utilizado na implementação dos experimentos controlados via página cliente. A seção 2.3.3 apresenta as principais características da placa Arduino Uno utilizada como sistema embarcado na implementação física dos quatro experimentos deste trabalho de monografia.

2.6.1 Histórico

O primeiro sistema embarcado considerado com tal designação foi o *Apollo Guidance Computer*, um computador de guia operando em tempo real no projeto Apollo dos EUA. O primeiro verdadeiro sistema de computação embarcada produzido em massa foi o Minuteman míssil balístico intercontinental, construído em 1961, tendo sido atualizado em 1966 em sua segunda versão com uma grande quantidade de circuitos integrados. Esse fato permitiu uma significativa queda no preço de portas NAND que possibilitou o seu uso para fins comerciais[20].

Os sistemas embarcados tiveram seu emprego comercial massificado na década de 1980, devido em grande parte à integração dos recursos de *hardware* em SOC (*Systems On Chip*). Estes sistemas integram em uma única pastilha um microcontrolador ou microprocessador, memória (RAM, ROM, E2PROM), canais analógicos-digitais (A/D), *timmers*, controladores de entrada /saída, etc. facilitando o seu com poucos elementos de *hardware* externo.

O grande crescimento na produção de sistemas embarcados ocorreu em meados da década de 1980, com o surgimento de aparelhos eletrônicos (vídeo games e brinquedos com

[19] Site: http://br-arduino.org/2014/11/arduino-o-que-e-e-pra-que-serve.html
[20] Site: https://pt.wikipedia.org/wiki/Sistema_embarcado

alguma programação), de aplicações particulares (máquinas de lavar, celulares) e na indústria automotiva (computador de bordo de automóveis), etc (CAVALCANTE, 2005).

O avanço atual em recurso de *hardware* em quantidade, qualidade e capacidade de processamento está em ritmo sempre crescente, principalmente para atender às aplicações de sistema embarcados. O mercado dos microprocessadores para as aplicações embarcadas corresponde a 99% da produção em relação aos de propósitos gerais[21].

2.6.2 Aplicações e Classificações

Quando se escolhe um sistema embarcada para ser empregado no controle de um determinado processo, deve-se levar em consideração os recursos de *hardware* requeridos (tamanho de memória e dados, quantidade de pinos de propósitos gerais, canais A/D, *timmers*, tipos de barramentos que opera, etc.) e, principalmente, a sua capacidade de processamento (processador mono ou multinúcleo).

As aplicações desses sistemas podem ser divididas em cinco tipos principais: a) sistemas de controle; b) processamento de sinais; c) comunicações e redes; d) aquisição de dados; e e) setor automobilístico (CUNHA, 2013).

Os sistemas de controle podem ser implementados em malha aberta ou fechada, sendo preferido o segundo tipo devido à realimentação de informações da planta automatizada. No controle de um determinado processo é comum a utilização de sistemas embarcados que possuem baixo nível de interface com o usuário como, por exemplo, os motores de automóveis (computador de bordo), processos industriais em diversos espectros de complexidades, aplicações aeroespaciais, etc. Normalmente nas aplicações citadas, as interfaces são providenciadas por outro tipo de programa, conhecidos como supervisórios (CUNHA, 2013).

Os processamentos de sinais, em função de uma alta quantidade de dados para processar, requerem uma alta capacidade de processamento em um baixo tempo de execução. Os sistemas embarcados que empregam *hardware* reconfigurável FPGA (*Field Programável Gate Array*), por possuírem capacidade de processamento paralelo, tornam-se extremamente

[21] Site: http://home.ufam.edu.br/lucascordeiro/str/slides/01-introducao-sistemas-de-tempo-real.pdf

adequados para aplicação com requisitos de resposta em tempo real. Como exemplo, têm-se compressão de vídeo, radares, etc (CHASE, 2007).

Os sistemas de comunicações e redes são típicos para aplicações dedicadas que realizam a tarefa de distribuição e chaveamento da informação. Como exemplo têm-se os sistemas de telefonia e telecomunicações, além da internet (CUNHA, 2013).

A aquisição de dados são sistemas que capturam informações através de sensores e as armazenam para uma futura consulta. Esses dispositivos são outros exemplos de sistemas embarcados e o tipo de variáveis analisadas são temperatura, umidade, pH (potencial Hidrogeniônico), etc (CHASE, 2007).

Os veículos atualmente possuem um complexo sistema embarcado, que capturam informações completas do veículo e até tomam algumas decisões, caso seja necessário acionar algum atuador (CHASE, 2007).

2.6.3 Sistema embarcado Arduino

Entre as plataformas de sistemas embarcados presente no mercado, a placa Arduino é a que apresenta valor comercial mais acessível para a tecnológica totalmente *open source.* As características dessa plataforma atenderam satisfatoriamente a todas as condições para os sistemas desenvolvidos que foram os objetos de controle e monitoramento via *web* propostos nesse trabalho.

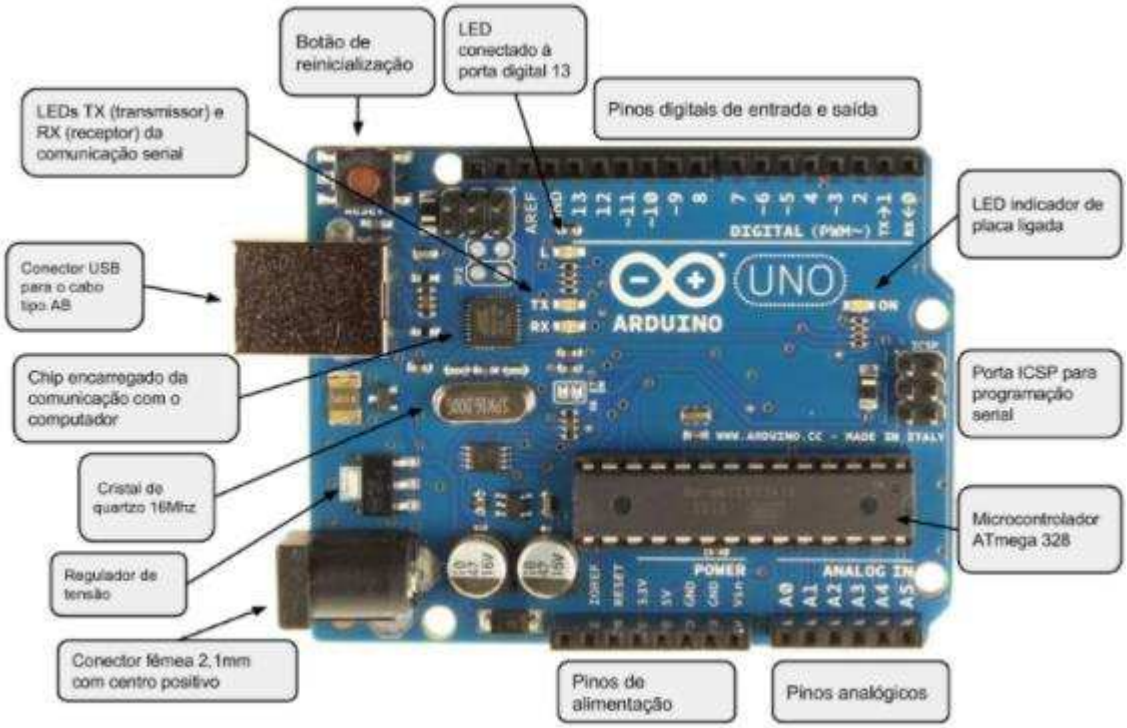
A plataforma Arduino é um microcomputador que pode processar entrada e saída de dados entre dispositivos e componentes externos conectados a ele, ou seja, o Arduino é uma plataforma de computação embarcada que interage com outros ambientes por meio de *software* e *hardware* (MACROBERTS, 2010). Foi projetado na Itália em 2005 com a intenção de uso em projetos de baixos custos[22].

A versão da plataforma que foi utilizado nessa monografia foi o Arduino UNO que é a mais conhecida, sendo considerada responsável pela popularização da plataforma. Apresenta na sua arquitetura o microprocessador ATmega 328, contendo 13 pinos de entrada e saída (I/O) digital, dos quais seis tem suporte PWM (*Pulse Width Modulation*) e seis entradas analógicas

[22] Site: https://pt.wikipedia.org/wiki/Arduino

(utilizadas para acoplar sensores com informações analógicas, por exemplo o sensor de distância infravermelho)[23]. Na Figura 11 tem-se uma visão das principais partes eletrônicas da placa.

Figura 11 – Características da placa Arduino UNO.



Fonte: http://natalmakers.blogspot.com.br/2015/08/dispositivo-conhecendo-as-partes-do.html

A Alimentação da placa pode ser feita pelo conector USB ou por uma fonte externa. Nesta última opção é disponibilizado um conector JACK fêmea da família TRS[24] usado para transmissão de sinais analógicos. O valor da tensão de entrada por esse conector pode variar de 7 a 12 V, enquanto pelo conector USB é no máximo 5V[25].

Outra fonte de alimentação presente na placa fica nos pinos designada para o valor de 3,3 V, tensão que server para alimentar os módulos conectados à placa para expansão de seus recursos. Como exemplo, pode-se citar o *shield* de ethernet que permite que a placa seja conectada à Internet, enquanto a fonte de 5V é empregada para alimentação de módulos externos como um pequeno motor DC[26]. A placa possui três pinos de referência (GND).

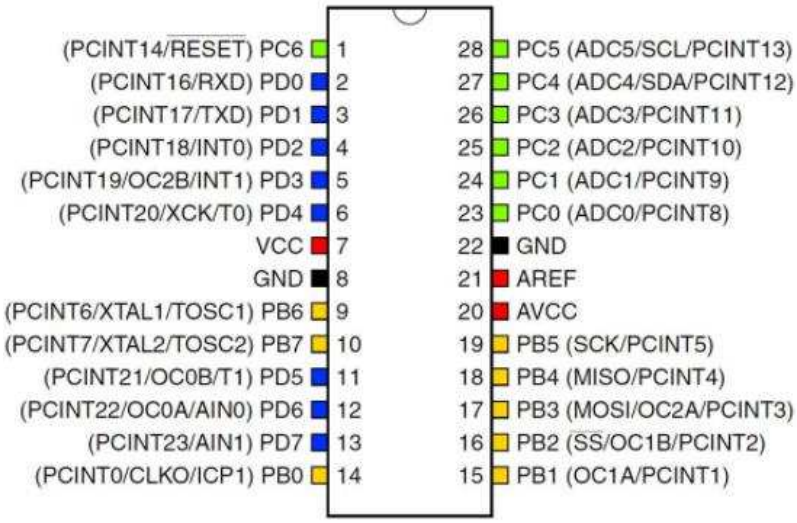
[23] Site: http://natalmakers.blogspot.com.br/2015/08/dispositivo-conhecendo-as-partes-do.html
[24] Site: https://pt.wikipedia.org/wiki/Conector_TRS
[25] Site: http://www.webtronico.com/arduino-uno-r3.html
[26] Site: http://wiki.seeedstudio.com/wiki/Ethernet_Shield

A comunicação entre a placa Arduino e outro computador é gerenciada pelo chip ATMEL ATMEGA16U2, o qual possibilita o *upload* do código binário após a compilação feita por um usuário da placa a partir de outro equipamento computacional. Outra forma de comunicação é pelos pinos TX e RX controlados pelo software do microcontrolador, indicando o envio e recebimento de dados serialmente[27]. A placa possui um conector de comunicação denominado ICSP[28] que server para gravação de *firmware* feita por programação. Esse microcontrolador possui um cristal de frequência de 16 MHz, o componente responsável para determinar o *clock* do microcontrolador[29].

O microcontrolador ATMEL ATEMGA328 é o responsável pelo gerenciamento dos recursos da placa. É um dispositivo de 8 bits da família AVR com arquitetura RISC avançada e encapsulamento DIP de 28 pinos. Ele tem 32 KB de memória *flash*, 2 KB da RAM e 1 KB de EEPROM, operando em uma frequência máxima de 20 MHz[30]. Na imagem da Figura 12 é apresentada a pinagem ATEMGA328.

A programação da plataforma foi feita pelo Arduino IDE (*Integrated Development Environment*) que é uma multiplataforma escrita em linguagem JAVA[31], requisitada para

Figura 12 – Pinagem do microcontrolador ATMEGA328.

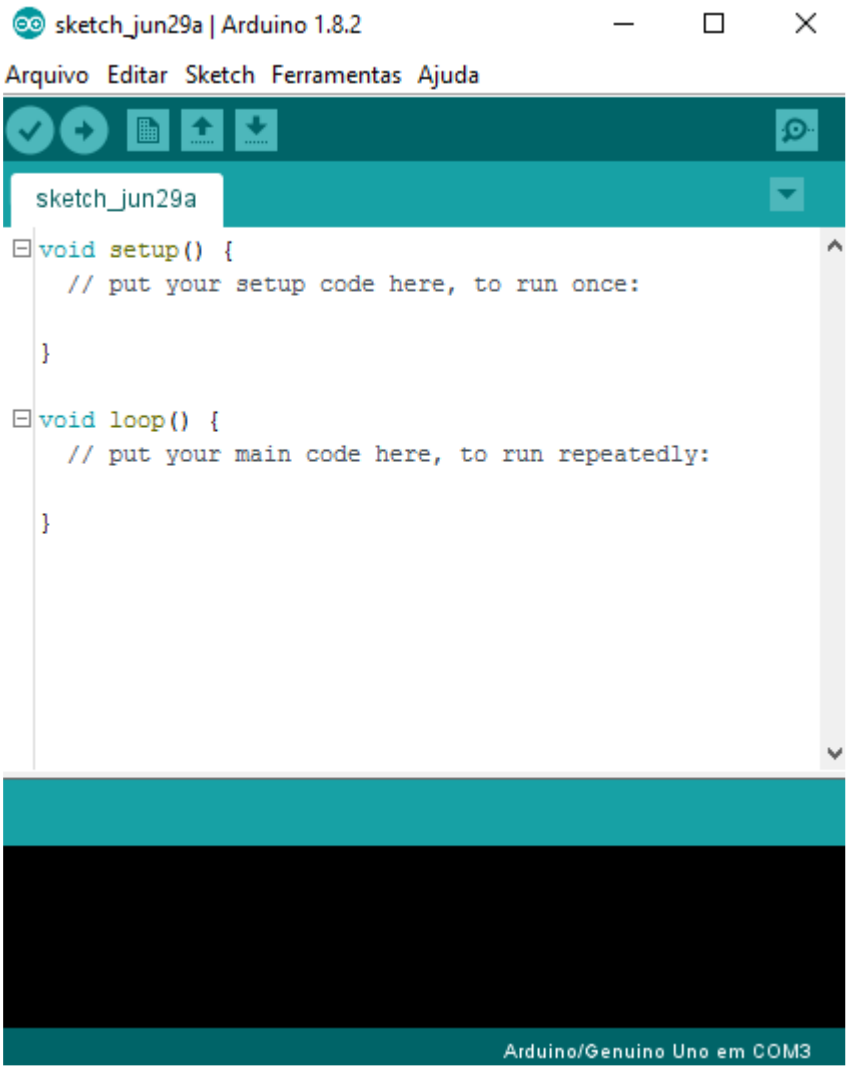


Fonte: http://produto.mercadolivre.com.br/MLB-719924682-atmega328p-pu-bootloader-atmega328-adesivo-pinagem-_JM

[27] Site: https://playground.arduino.cc/Referencia/Serial
[28] Site: http://br-arduino.org/2015/05/arduino-icsp-attiny-atmega.html
[29] Site: http://www.filipeflop.com/pd-2269b0-cristal-oscilador-16mhz.html
[30] Site: https://en.wikipedia.org/wiki/ATmega328
[31] Site: http://www.programacaoprogressiva.net/2012/08/comece-programar-linguagem-de.html

desenvolvimentos de *softwares* que serão executados na placa Arduino. A Figura 13 mostra o terminal do IDE mencionado (MACROBERTS, 2010).

Figura 13 – Terminal de programação do IDE Arduino.

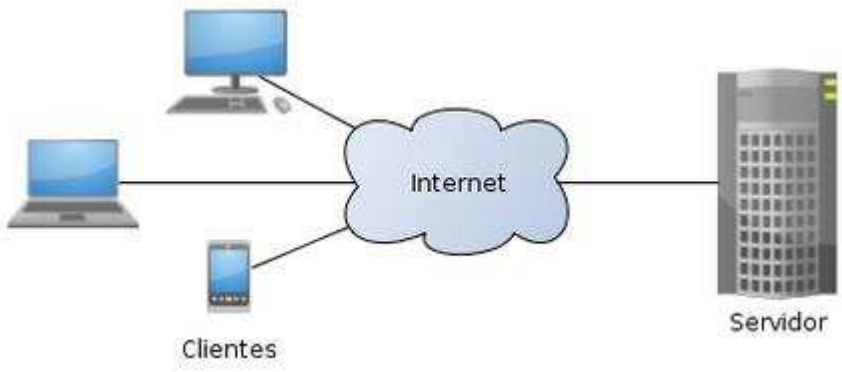


Fonte: MACROBERTS, 2010

3. DESCRIÇÃO DA TOPOLOGIA CLIENTE-SERVIDOR IMPLEMENTADA

A topologia cliente-servidor compreende uma rede de computadores onde dois ou mais equipamentos computacionais compartilham informações. A máquina cliente é a que requisita a informação, podendo ser um computador ou outro dispositivo que solicite trafego de rede. O computador servidor é a que processa as solicitações e responde aos pedidos requisitados pelo cliente (TORRES,2001). A Figura 14 ilustra um computador servidor sendo acessado via internet por 3 máquinas clientes distintas, no caso, um PC, um notebook e um *tablet*, constituindo uma exemplificação da topologia cliente-servidor.

Figura 14 – Exemplificação de elementos na topologia Cliente-Servidor.



Fonte: *https://pt.wikipedia.org/wiki/Cliente-servidor*

A comunicação entre eles acontece quando o cliente envia uma solicitação de sincronização para um servidor (exemplo: '**www.ufma.br**'). O servidor recebe a solicitação e envia ao cliente uma solicitação de sincronização como resposta de confirmação para a sincronização. Na sequência, o cliente envia ao servidor uma última mensagem, anexada com a confirmação da solicitação de sincronização enviada pelo servidor, criando o canal de conexão com entre ambas as máquinas, de forma que o servidor permita o acesso (KUROSE, 2010). Esse protocolo inicial de conexão do cliente com o servidor é mostrado na Figura 15.

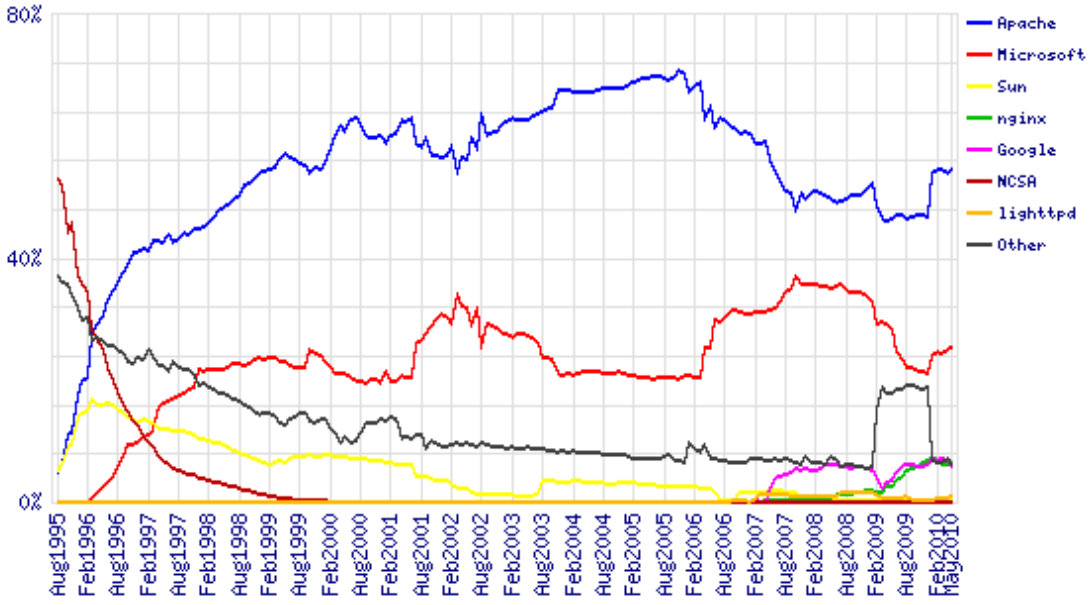
O servidor *web* utilizado foi o Apache que segundo uma pesquisa realizada pela Netcraf[32], realizada entre agosto de 1995 até maio de 2010, foi o servidor *web* mais usado no mundo como mostra a linha azul do gráfico na Figura 16.

[32] Site: https://news.netcraft.com/archives/2010/06/16/june-2010-web-server-survey.html

Figura 15 – Movimentação de envio e resposta de solicitações entre o cliente e o servidor.

Fonte: (KUROSE, J. e ROSS, W., 2010)

Figura 16 – Percentual de Compartilhamento de mercado para melhores servidores em todos os domínios (1995-2010).



Fonte: https://news.netcraft.com/archives/2010/06/16/june-2010-web-server-survey.html

Em pesquisa realizada no site ***The Apache Software Foundation***[33], o servidor Apache apresenta características satisfatórias para o desenvolvimento e operação do servidor que será utilizado por essa monografia, que são:

a) Suporte para plataformas *open source* (Linux) e proprietárias (Windows);

b) Não possui custo de utilização;

c) Estruturação do programa servidor em módulos; e

d) Suporta várias linguagens: PHP, PEARL, HTML, PYTHON.

Nesta monografia, o servidor foi desenvolvido para operar em uma LAN (*Local Area Network*) juntamente com a máquina cliente, tendo a proposta sido implementada e testada na rede LMI do LRC. Ainda que restrita a dimensão geográfica de uma LAN, apresenta como vantagem uma implementação mais simplificada de rede e a possibilidade de se obter uma velocidade de comunicação de dados em torno de 100 a 1000 Mbps. O desenvolvimento modular dos experimentos ao nível da rede local pode ser expandido para acesso de outra rede, dependendo apenas de configurações ao nível de servidor e de adaptação dos códigos PHP.

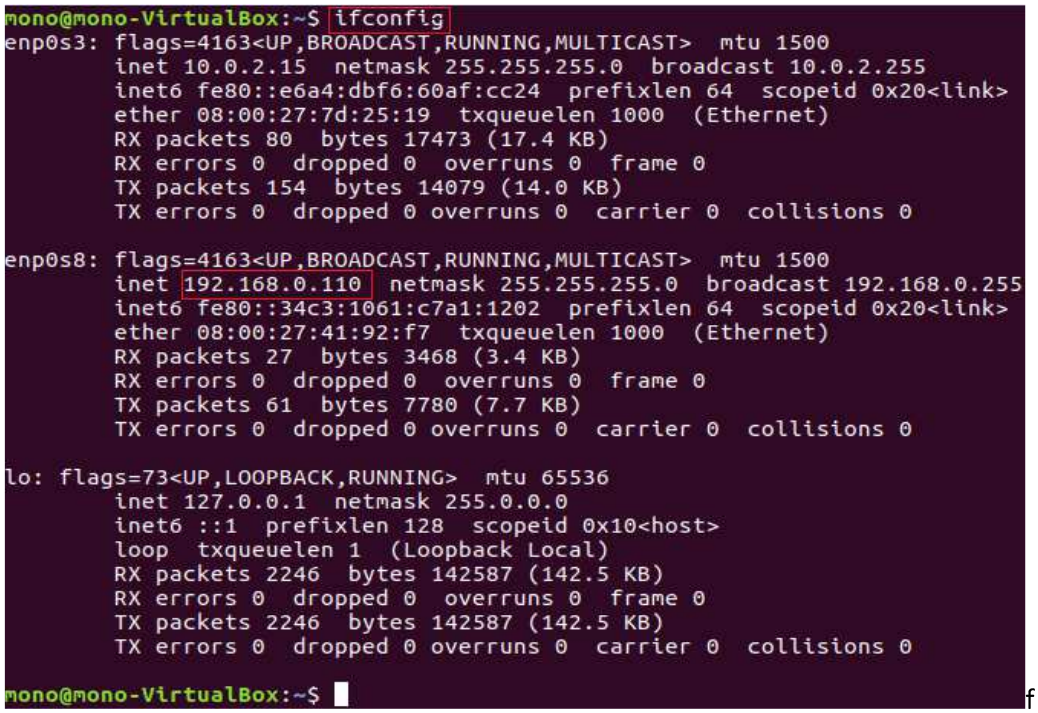
A forma do cliente acessar o servidor Apache é por meio de um numero IP (*Internet Protocol*), que consiste em um código de acesso composto de números separados por campos de um até três dígitos, os quais identificam uma máquina em particular conectada à rede (endereço na rede é único). Dessa forma, o endereço IP compreende quatro conjuntos de números de 0 a 255 (octetos), separados por três pontos como, por exemplo '192.168.0.110'. (TANEBAUM, 2011).

O IP do servidor implementado será do tipo dinâmico. Esse tipo de IP é gerado por roteadores, que são aparelhos que interligam os computadores através da rede. Nesse caso, a cada roteador de uma rede local ao qual o servidor se conectar gera ao cliente, de forma dinâmica, um número de IP diferente. Para conhecer o IP de acesso do cliente se faz uma consulta pelo terminal do sistema operacional, por exemplo no LINUX com o comando '**ifconfig**', conforme tela do terminal mostrada na Figura 17.

A primeira parte do endereço é formada pelo primeiro e segundo octetos, definindo a rede em que o servidor está conectado. A segunda parte identifica o endereço do computador

[33] Site: https://www.apache.org/foundation/

Figura 17 – Usando o comando ifconfig no terminal do Linux.



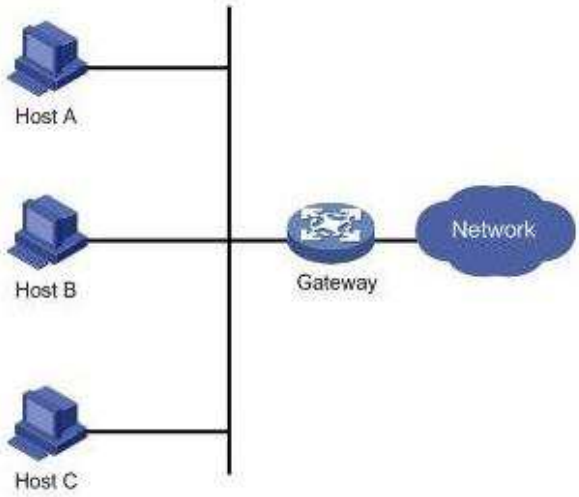
Fonte: Autor

(*host*) conectado a essa rede. No exemplo de IP citado acima, '192.168.0.110', a rede é identificada por '192.168' e o *host* por '0.110' (TANEBAUM, 2011).

Além do endereço IP, tem-se o *gateway*, no caso, o "portão" que permite o acesso de dados entre os *hosts* de diferentes redes que conecta o servidor na rede quando for acessado, como ilustra a Figura 18. O *gateway* possui um formato do mesmo tipo do IP que, no caso de um computador conectado ao modem em uma residência, tem o mesmo número do IP do modem[34]. O *gateway* do servidor implementado neste trabalho de monografia é '192.168.0.1'. O IP possui também uma máscara, denominada de *subnet mask*[35], que tem como função dividir o endereço IP em duas porções citadas. A máscara também possui 4 octetos, onde os valores "1" representam a parcela de um endereço IP visto pela rede e os valores "0" representam a parcela do endereço do *host*. O valor do servidor implementado neste trabalho para a *subnet mask* foi definido como 255.255.255.0. A porta de comunicação utilizada é a 8081, usada na de comunicação do protocolo HTTP (*HyperText Transfer Protocol*), conforme será explicado mais

[34] Site: https://www.portaleducacao.com.br/conteudo/artigos/direito/afinal-o-que-e-ip-mascara-gateway-e-dns/49129
[35] Site: http://www.informabr.com.br/ipsubnet.htm

Figura 18 – Exemplo ilustrativo de um *gateway*.



Fonte: https://www.palpitedigital.com/o-que-e-gateway/

adiante. O número da porta será atribuído a depender do tipo de protocolo que irá trafegar pela internet (TANEMBAUM, 2011).

No apêndice A é descrito todo o processo de instalação e configuração do servidor Apache no SO Linux Ubuntu, executando na máquina servidor neste trabalho de monografia. Após todo o processo de construção do servidor, o sistema cria automaticamente a pasta denominada ‘www’ que ficara localizada na pasta ‘root’ do servidor. Dentro da pasta ‘www’ cria-se a pasta ‘html’, que é o local da instalação do arquivo ‘index.hml’ (página inicial) dentro do diretório do servidor web. O servidor ficará encarregado de procurar pelo arquivo, quando o cliente for acessá-lo pelo endereço IP. A página principal e as demais subpáginas dos experimentos foram configuradas como página principal e programada para este trabalho na linguagem HTML (*Hyper Text Markup Language*).

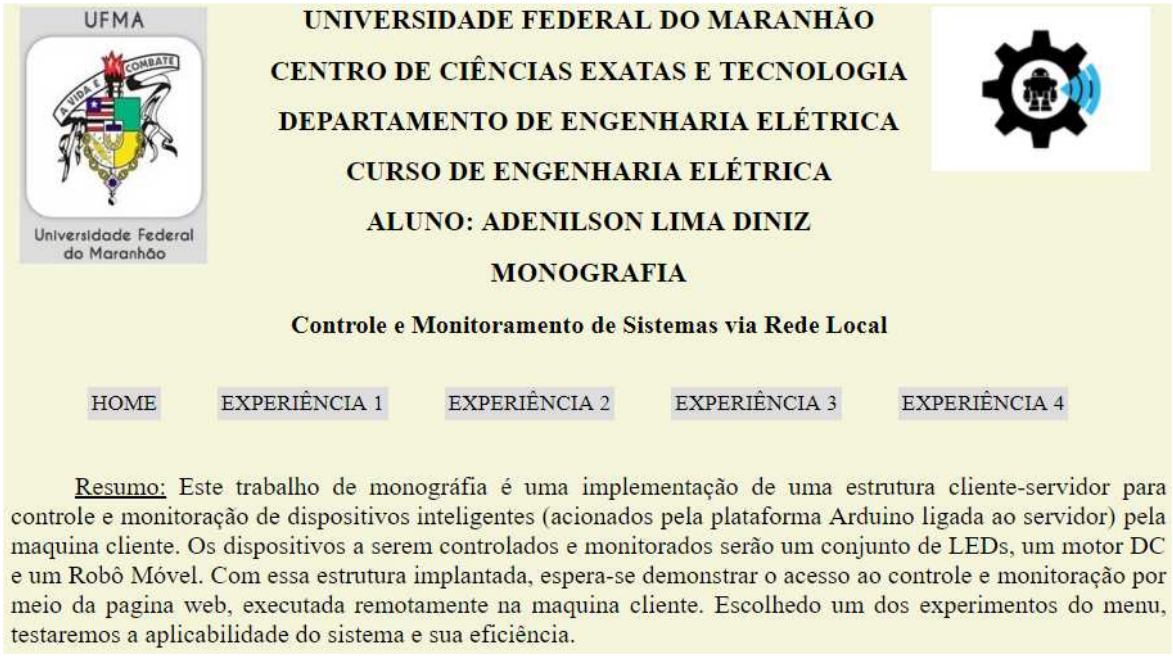
O HTML é uma linguagem de marcação usada para construção de sites que serão interpretados pelos negadores (por exemplo, o Google Chrome). Ela é considerada uma linguagem estática por permitir definir o *layout* de um documento na *web*. Após sua definição e armazenamento no local de hospedagem da página (servidor), os arquivos em ‘**html**’ são processados pela máquina quando for acessado.

Outra linguagem usada neste trabalho foi o PHP (*Hypertext Preprocessor*), uma linguagem de script *open source* de uso geral, especialmente adequada para desenvolvimento

web e que pode ser embutido dentro do código HTML. É considerada uma linguagem dinâmica, porque pode processar a informação solicitada pelo cliente e exibir o conteúdo em HTML. Sua escolha foi em função de sua facilidade de uso de seus comandos para os fins de programação.

Como dito, neste trabalho a página principal 'index.html' é a interface que o cliente faz o acesso ao servidor pelo endereço IP. O *layout* da página que aparece na máquina cliente é apresentada na Figura 19.

Figura 19 – Interface principal do programa 'index.html'.

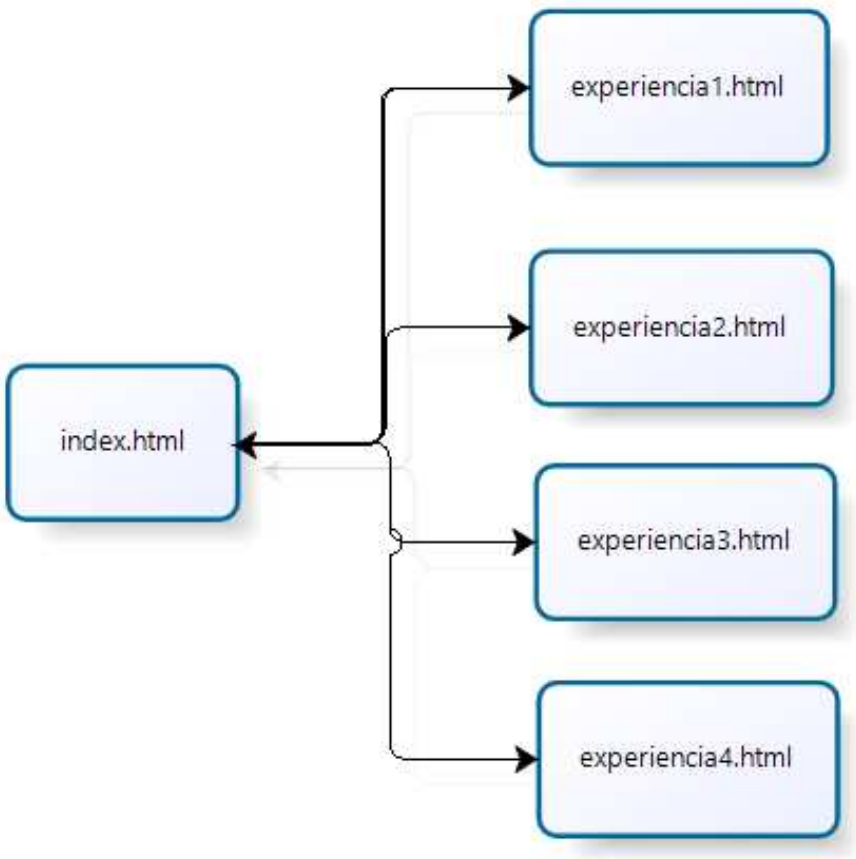


Fonte: Autor

Na interface principal facilmente se identifica o menu de acesso aos experimentos, sendo cada um deles uma subpágina em HTML que ficam armazenadas na pasta '**html**' criada no servidor. Elas são conectadas à página 'index.html' por meio do comando '**href < >**' em HTML, o qual possibilita uma comunicação bidirecional entre a página principal e uma das subpáginas. (NIEDERAUER, 2008). Na figura 20 se tem uma visão bidirecional na comunicação entre a página principal e as subpáginas.

Uma outra conexão existente é entre as páginas experimentos e os programas em PHP que processarão as solicitações do cliente no servidor e depois passando o devido comando ao sistema embarcado através da porta USB. É necessário que o cliente escolha uma opção do menu da interface presente no 'index.html' para que a subpágina seja disponibilizada e o usuário possa fazer a opção de acionamento do dispositivo conectado ao sistema embarcado.

Figura 20 – Comunicação bidirecional entre a página principal e as subpáginas do sistema.



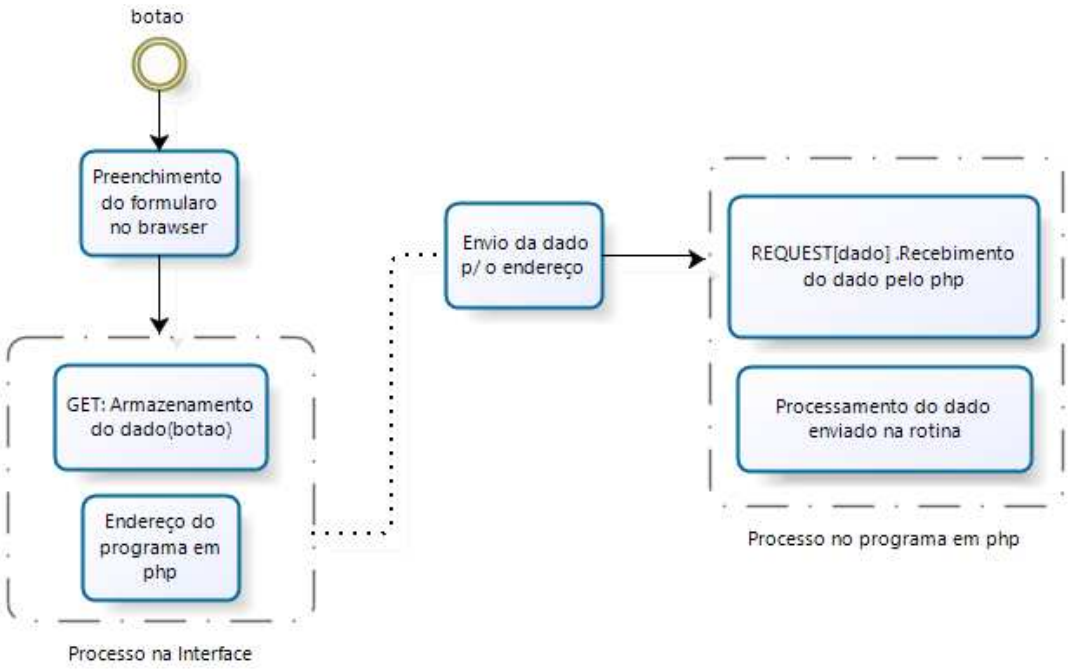
Fonte: Autor

Acionado um botão, o cliente está preenchendo automaticamente um formulário web no *browser* que guardara este dado na variável '*value*', sendo armazenado por meio do comando '**GET**' do PHP. Essa instrução é um atributo que permite ao *browser* armazenar requerimento feitos pelo cliente. (NIEDERAUER, 2008).

Após o armazenamento, endereça-se para qual programa em PHP será enviado esse dado. O endereçamento é feito pelo atributo da linguagem dinâmica denominado '<form action= >'. Um exemplo de endereçamento com esse atributo é '<form action = "Arduino.php">' (NIEDERAUER, 2008).

O recebimento do dado pelo programa PHP é feito pela variável '**REQUEST[ ]**', sendo processado na rotina do programa para definir a ação correspondente à solicitação feita remotamente pelo usuário através da página 'html'. (NIEDERAUER, 2008). Na Figura 21 é exemplificado o processo das ações realizadas após a pressão de um botão na página cliente em HTML até o seu atendimento e processamento pelo código PHP.

Figura 21 – Processo exemplificado a seleção de uma ação no programa PHP pedida no acionamento de uma ação na página ‘html’.



Fonte: Autor

4. OPERAÇÃO DO SISTEMA EMBARCADO VIA WEB.

Neste capitulo serão apresentados os experimentos implementados neste trabalho de monografia, visando demonstrar a conexão entre o servidor *web* (operando em uma rede local) com o sistema embarcado Arduino, bem como a realização dos experimentos remotamente via página cliente. Busca-se demonstrar a dinâmica de comunicação de envio e recebimento de requisições entre o servidor *web* e o sistema Arduino, além de validar a operação remota do sistema

A comunicação servidor-Arduino se dará via porta USB do servidor com a porta serial da placa Arduino por meio de um cabo. Esse tipo de conexão foi escolhido por garantir mais segurança, confiabilidade na transferência de dados, além de possibilitar um maior tráfego de dados possível entre o Arduino e o servidor[36].

4.1. Sistema para Ligar/Desligar um *LED*

Esse primeiro experimento consiste em acionar um *led* tendo como possibilidades a escolha do controle de intensidade da luminosidade (brilho) e da frequência de acendimento. Para realizar o controle da luminosidade do *led*, utilizou-se um sinal de modulação por largura de pulso (*Pulse Width Modulation* - PWM).

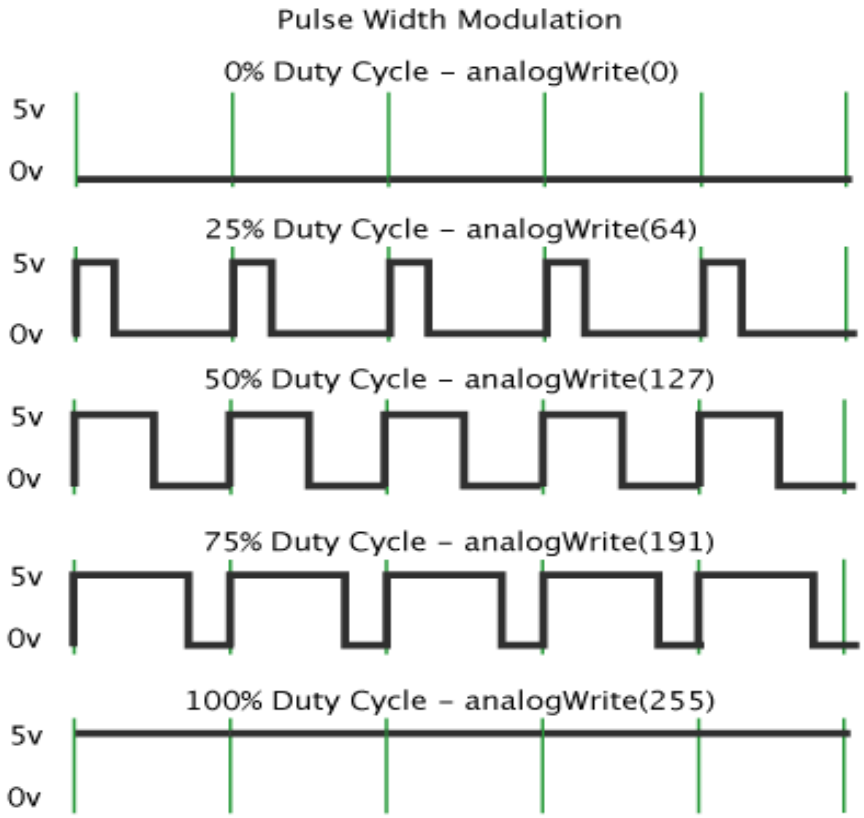
O PWM é uma técnica para obter resultados analógicos por meios digitais. O controle digital é usado para criar uma onda quadrada, um sinal alternado entre ligar (nível de tensão alto) e desligar (nível de tensão de referência ou terra do circuito). Este padrão de liga/desliga simula tensões médias entregues a dispositivos (*leds*, motores DC, etc.), definido pelo tempo de nível alto (*duty cycle*) em relação ao período do sinal PWM. Quanto maior for o *duty cycle* no período do sinal, maior será o valor médio fornecido. A Figura 22 mostra alguns gráficos de modulação do tempo de alternância de níveis do sinal PWM[37].

As linhas verdes do gráfico demonstram um período de tempo regular (exemplo do sinal superior da Figura 22), sendo a duração do período definida pelo inverso da frequência do sinal PWM. A escala de níveis do Arduino varia entre 0 (nível baixo) a 255 (nível alto), e as

[36]Site: http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/02/internet-a-cabo-ou-wi-fi-saiba-qual-e-a-mais-indicada-para-voce.html

[37] Site: www.arduino.cc/en/Tutorial/PWM

Figura 22 – Exemplos de Modulação de sinal PWM.



Fonte: https://www.arduino.cc/en/Tutorial/PWM

saídas para PWM na placa Arduino são os pinos que possuem o caractere '~' na frente do número do pino[38] como mostra a Figura 23.

Para o controle da luminosidade do *led* foram definidas 3 possibilidades de *duty cycle*, que possibilitou gerar sinais PWM por divisão da frequência do *clock* interno (1, 4 e 20) para se obter os efeitos de luminosidade na faixa considerada de alta luminosidade (divisão por 1), média luminosidade (divisão por 4) e baixa (divisão por 20). Percebe-se que quanto maior o *duty cycle* do sinal PWM enviado ao *led*, maior será o nível da tensão média e, portanto, a sua luminosidade.

Para o controle da frequência de acendimento do led em uma determinada luminosidade, foram definidos três valores, denominados: alta, média e baixa. Para o acionamento em alta frequência o período escolhido foi de 1ms, para média de 2ms e baixa de

[38] Site: http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/02/internet-a-cabo-ou-wi-fi-saiba-qual-e-a-mais-indicada-para-voce.html.

Figura 23 – Pinos de saídas do PWM com o caractere '~'.

Fonte: http://arduino.labdegaragem.com/Guia_preview/arduino_ide.html

5ms. As definições desses valores foram feitas de maneira a caracterizar uma diferença visual no acionamento.

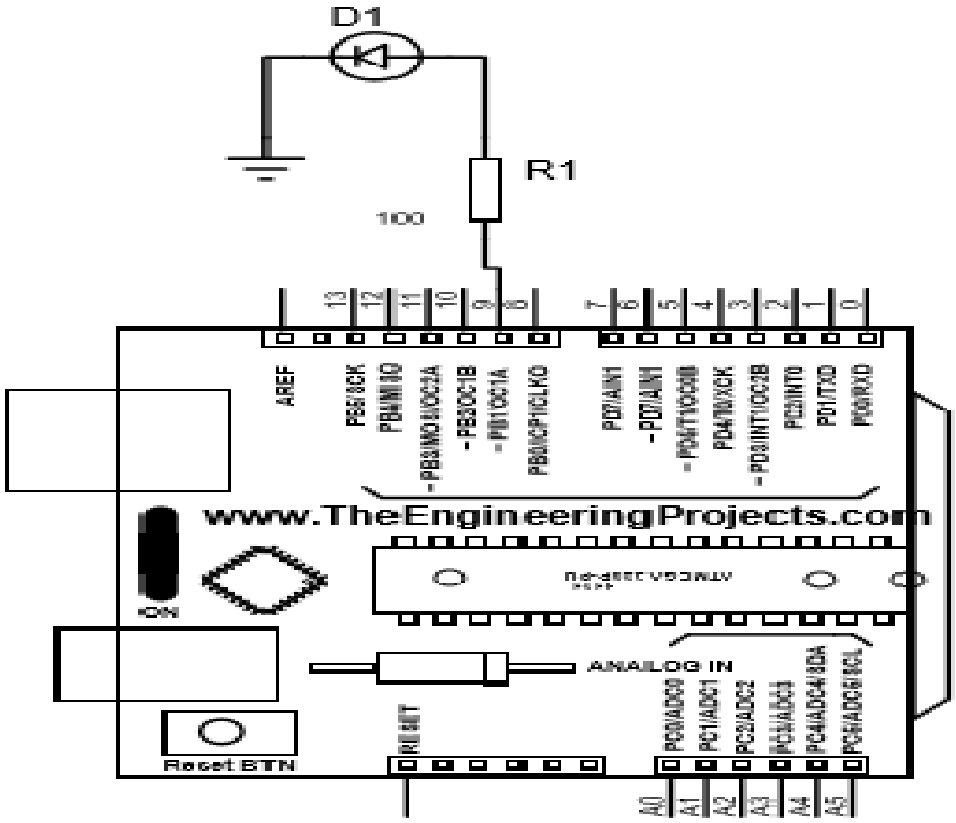
A montagem do circuito é de modelo simples, tendo o *layout* do circuito sido simulado no programa Proteus, que é uma ferramenta de projeto eletrônico automatizado, denominado EDA (*Eletronic Design Automation*). Esse programa possibilita montar o esquemático ideal do circuito, realizar a simulação e gerar o modulo de *layout* da PCB[39]. Na Figura 24 apresenta-se o *layout* do circuito do acionamento do *led*, o qual teve seus terminais conectados ao pino 9 do Arduino (programado como saída) e à referência do circuito (terra).

No servidor *web* Apache foi instalado o programa do IDE (*Integrated Drive Eletronics*) do Arduino contendo o programa chamado '**PWMLED.c**', cujo código encontra-se no apêndice C.1. O programa, executando no Arduino, recebe o comando via cabo do servidor e possui todas as três possibilidades de luminosidade e as três de frequência de acendimento, possíveis de serem combinadas e comandadas via página do cliente, conforme é descrito no Capítulo 5.

Além de possibilitar o controle do *led*, a porta do Arduino que está conectada ao servidor é usada para realizar a transferência do programa (*upload*) que é executado na placa.

[39] Site: https://pt.wikipedia.org/wiki/Proteus_(programa_de_computador)

Figura 24 – *Layout* do circuito do experimento 1 simulado no programa Proteus.



Fonte: Autor

Na placa há dois pinos que tem a função de realizar a comunicação com a porta USB do servidor, no caso os pinos digitais numerados como 0 (RX) e 1(TX)[40] como mostra a foto na Figura 25. As portas USB do servidor LINUX estarão aptas para a leitura e a gravação com o comando '**chmod 666 /dev/ttyUSB?**' (por exemplo, '**/dev/ttyUSB0**'), de maneira a possibilitar o tráfego de dados entre o servidor e a placa Arduino. (MARIMOTO,2005).

Outro programa desenvolvido para o acionamento do led via web é o 'arduinoa.php', que recebe as requisições do programa 'experiencia1.html' (ambos serão explicados no capitulo 5) que interage com o cliente. O programa PHP enviará comandos de execução para o software embarcado na placa Arduino 'PWMLED.c'. Primeiramente, O 'arduinoa.php' receberá as solicitações enviadas pelo programa 'experiencia1.html', depois abrirá a conexão de comunicação na porta USB, utilizando a porta '/dev/ttyUSB?', através do

[40] Site: https://pt.wikipedia.org/wiki/Proteus_(programa_de_computador)

Figura 25 – Portas de comunicação serial (1) TX e (0) RX (retângulo amarelo).

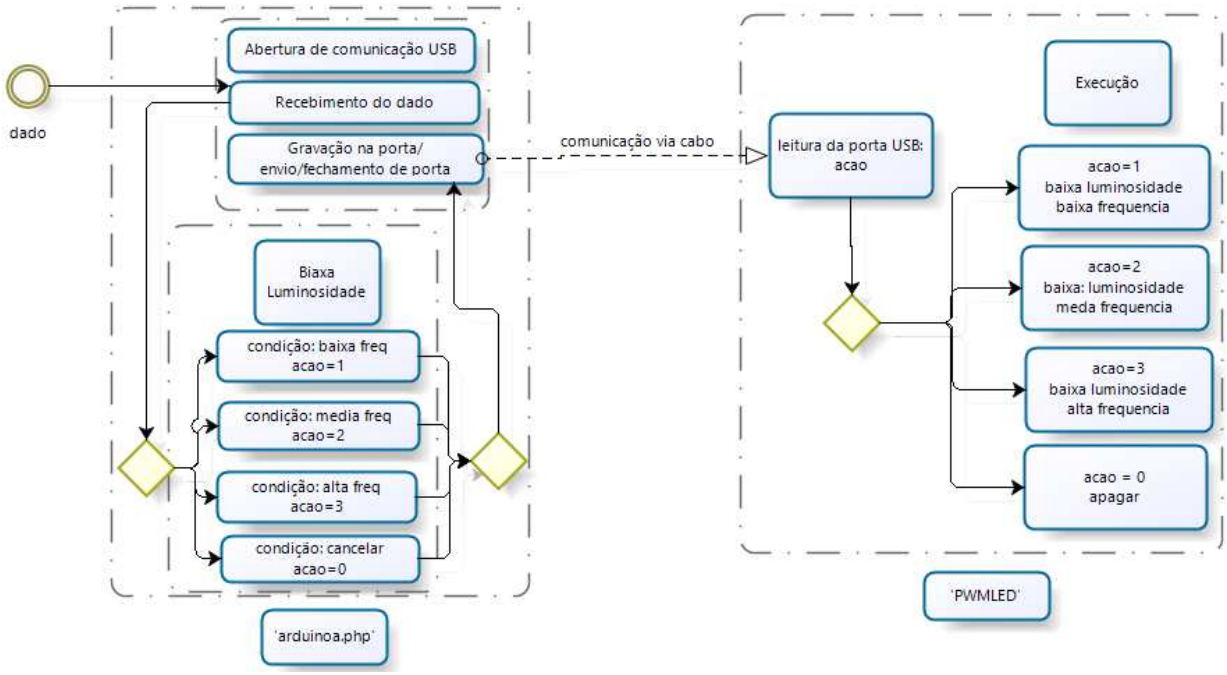
Fonte: https://www.embarcados.com.br/arduino-comunicacao-serial/

comando ‘fopen ()’. O programa em PHP também escrever na porta (parâmetro ‘w’), seguindo o formato ‘fopen (‘/dev/ttyUSB0’, ‘w’) ’ (NIEDERAUER, 2008).

Após isso, o código ‘**arduinoa.php**’ (apêndice B.1) busca em sua rotina de condições o modelo de ação que condiz com a solicitação enviada a ele. Localizado a condição solicitada, ele escreve essa ação especifica na porta com o comando ‘**fwrite ()** ’. Depois, o programa fecha a porta de comunicação com o comando ‘ **fclose ()** ’, para confirmar que o comando foi enviado e habilitar a porta para receber uma próxima requisição do '**experiencia1.html**'. (NIEDERAUER, 2008). Na Figura 26 é apresentada a comunicação entre o servidor e a placa através dos fluxogramas dos programas ‘**arduinoa.php**’ (servidor) e ‘**PWMLED.c**’ (Arduino).

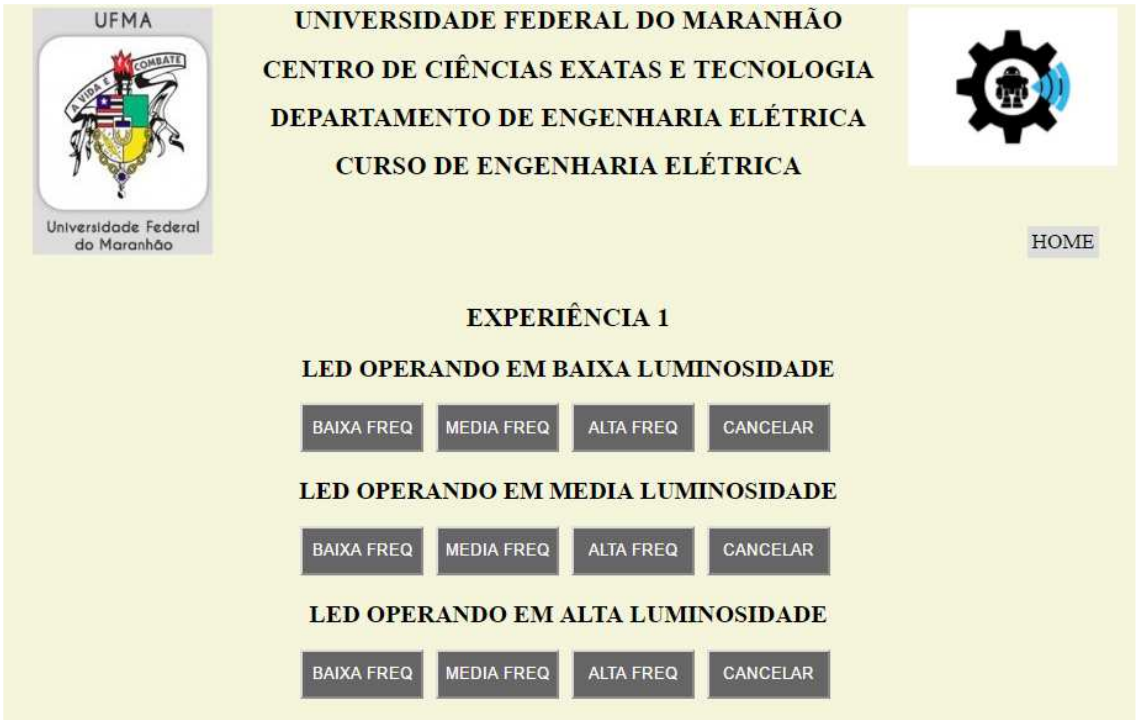
A Figura 27 apresenta a interface que o usuário tem na máquina cliente quando escolher a opção na página principal para operar o experimento 1, armazenado no servidor com o nome ‘**experiencia1.html**’. Ao escolher uma ação nesta página, esse programa enviará sua requisição para o programa em PHP ‘**arduinoa.php**’, este último executando no servidor Apache, cuja arquitetura é mostrada na Figura 28.

Figura 26 – Arquitetura de processamento e de comunicação entre os programas '**arduinoa.php**' e '**PWMLED**'.



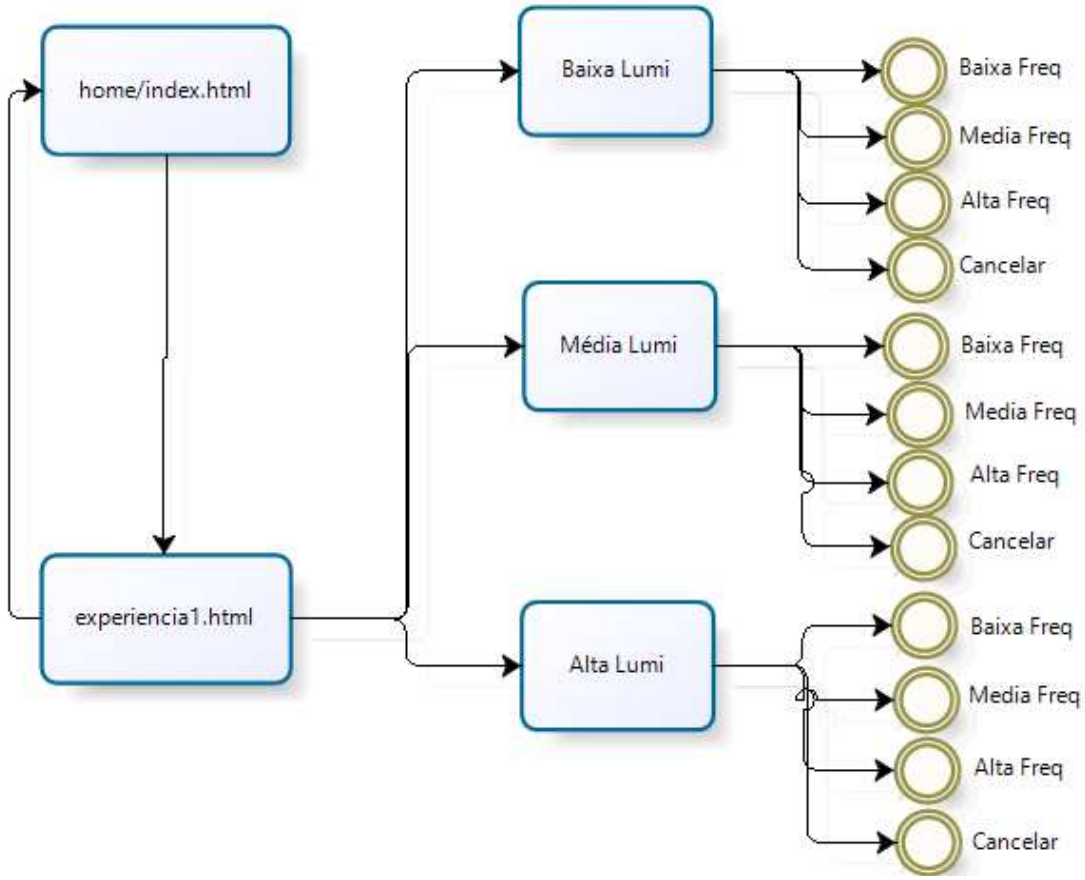
Fonte: Autor

Figura 27 – Interface do Experimento 1 – programa '**experiencia1.html**'.



Fonte: Autor

Figura 28 – Arquitetura de comando do programa '**arduinoa.php**'.



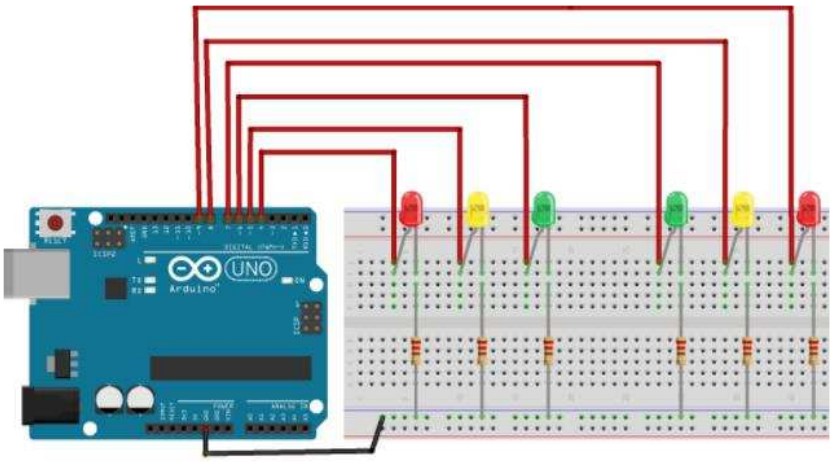
Fonte: Autor

4.2. Sistema de Acionamento e Controle de um *Array de LEDS*

Para o experimento 2 foi proposto o controle de um conjunto formado por seis *leds* no qual se busca realizar o acendimento dos mesmos considerando o sentido (direita ou esquerda), a sequência (um ou dois leds por vez) e a frequência (alta ou baixa). Na Figura 29 é apresentado a organização da montagem dos leds no *protoboard*.

O sentido de acendimento diz respeito a animação ser feita em série (1 *led* por vez) para a direita ou para a esquerda. Quando dois leds foram acendidos simultaneamente (para a direita ou para a esquerda), denomina-se acendimento paralelo. A quantidade de leds que podem ser usados e quantos são considerados no acendimento paralelo está dependente da quantidade de pinos disponíveis de propósitos gerais no Arduino. Caso seja utilizado circuitos

Figura 29 – Layout da organização dos leds no protoboard e conectados nos pinos do Arduino.

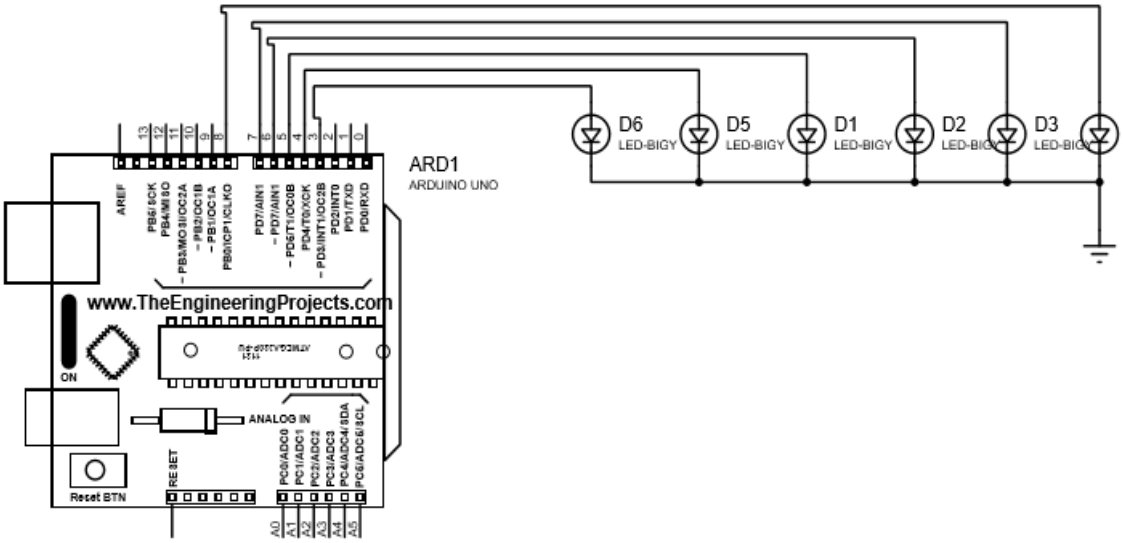


Fonte: http://fritzing.org/projects/secuencia-de-6-leds-con-arduino

de expansão de pinos que se comunica por I2C com o Arduino, como o circuito PCF8574[41], é possível ampliar a quantidade de pinos e, portanto, de leds para serem acionados.

As frequências definidas são análogas à do experimento comentado na Seção 4.1, com a diferença de que neste experimento se usará as frequências em alta e baixa. A frequência alta tem o período de acionamento de 2 milissegundos, enquanto a baixa de 5 milissegundos. Na Figura 30 é apresentada a montagem desse experimento no ambiente Proteus.

Figura 30 – Circuito do experimento 2 simulado no Proteus.



Fonte: Autor

[41] Site: http://labdegaragem.com/profiles/blogs/tutorial-sobre-pcf8574-e-como-utiliz-lo-com-arduino

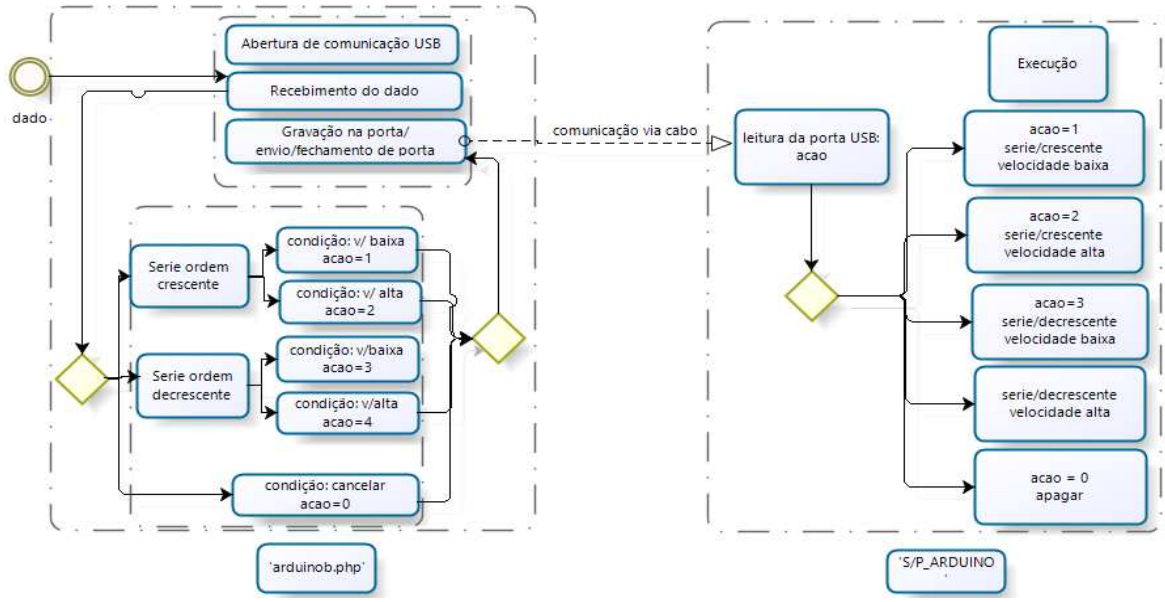
O programa **'S/P_ARDUINO'** que realiza o controle do padrão de acendimento do experimento foi gravado no Arduino. No servidor *web* tem-se para esse experimento o programa **'arduinob.php'** que opera de forma análoga ao do experimento anterior. Os códigos dos programas **'S/P_ARDUINO.c'** e **'arduinob.php'** estão nos apêndices C.2 e B.2, respectivamente. A Figura 31 apresenta a comunicação entre esses programas, representados por fluxogramas, executando no servidor (‘**arduinob.php**’) e no Arduino (**'S/P_ARDUINO.c'**).

A Figura 32 apresenta a interface que o usuário dispõe na máquina cliente quando escolher a opção na página principal para operar o experimento 2, programa denominado ‘**experiencia2.html**’. Ao escolher uma ação nesta página, esse programa enviará sua requisição para o programa em PHP ‘**arduinob.php**’, cuja arquitetura é mostrada na Figura 33.

4.3. Sistema para Acionamento de Motor CC

Neste experimento desenvolveu-se um sistema para controlar o sentido (horário e anti-horário) e a velocidade nas porcentagens de 20%, 50% e 100% da velocidade nominal de um pequeno motor DC (corrente contínua). O motor é alimentado por pilhas e formado por uma bobina que gira dentro de um campo magnético de imãs. (FITZGERALD,2006). A Figura 34 ilustra a estrutura mecânica de um motor DC genérico.

Figura 31 – Arquitetura de processamento e de comunicação entre os programas ‘**arduinob.php**’ e ‘**S/P_ARDUINO**’.

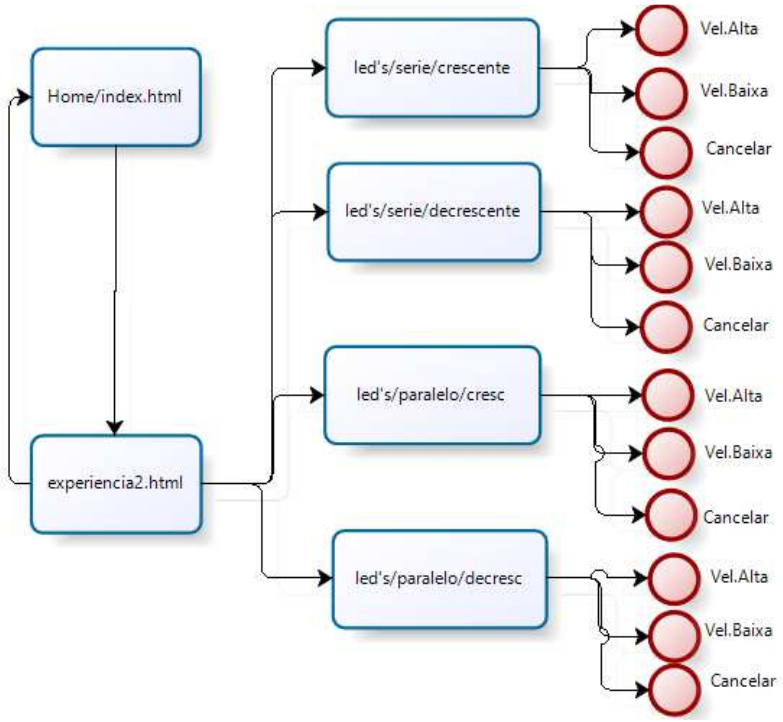


Fonte: Autor

Figura 32 – Interface do programa – '**experiencia2.html**'.

Fonte: Autor

Figura 33 – Arquitetura de comando do programa '**arduinob.php**'.



Fonte: Autor

Figura 34 – Visão da estrutura mecânica de um motor DC genérico.

Fonte: http://www.newtoncbraga.com.br/index.php/robotica/5168-mec070a

O conjunto de escovas comuta a corrente de modo que ela possa ser invertida na bobina, dando continuidade ao movimento de rotação. Seu funcionamento consiste na aplicação de uma tensão em seus terminais que criaram uma corrente circulante nas bobinas e assim produzindo movimento (FITZGERALD,2006).

A tensão do motor desse experimento é próxima a 5 V, sendo a corrente de máxima aproximadamente 200 mA, gerando uma rotação de 3.000 rpm e produzindo um pequeno torque de 5-50g.cm[42]. A corrente e o torque são diretamente proporcionais.

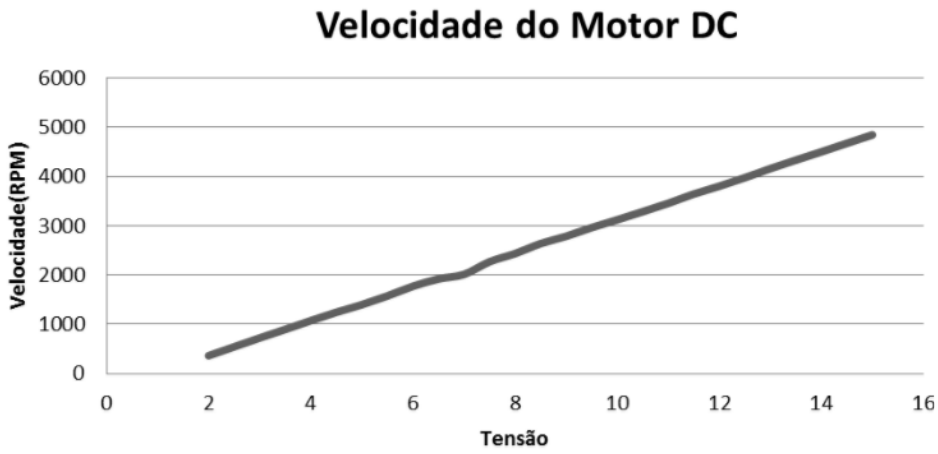
O motor usado neste trabalho pode operar em diferentes valores de tensão, possibilitando a variação da corrente, velocidade e torque. Neste experimento o motor trabalhará sem carga, ou seja, o motor gira livremente tendo sua velocidade e corrente consumida aumentando ou diminuído linearmente com a variação da tensão aplicada[43], conforme mostra o gráfico da Figura 35.

Para que se pudesse visualizar ao nível local o sentido de rotação do motor, foram colocados dois leds, um para cada sentido. O layout do circuito completo é apresentado na Figura 36, cujo esquemático foi construído no programa PROTEUS.

Na Figura 37 é mostrado uma placa eletrônica de ponte H que é um circuito de potência que converte uma fonte fixa de corrente continua em uma tensão de corrente continua com valor que pode ser variável através do controle (sinal PWM). O sinal PWM e o sinal que

[42] https://www.alibaba.com/product-detail/5v-mini-dc-motor-3000rpm_716727213.html
[43] http://www.newtoncbraga.com.br/index.php/como-funciona/2829-mec060

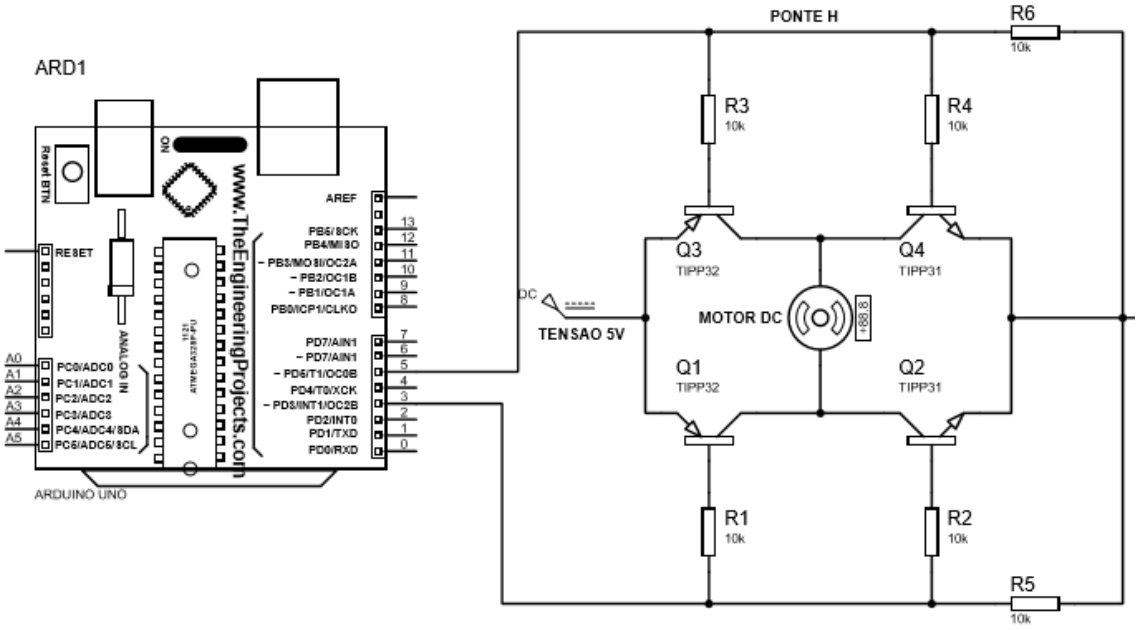
Figura 35 – Relação da tensão com a velocidade.



Fonte: https://engc49ufba20132mgf.wordpress.com/2014/02/13/medicao-de-velocidade-angular/

informa o sentido do giro do motor são fornecidos para o drive ponte HL298N, o qual pode acionar dois motores DC de forma independente, definindo a velocidade e o sentido de giro do motor[44].

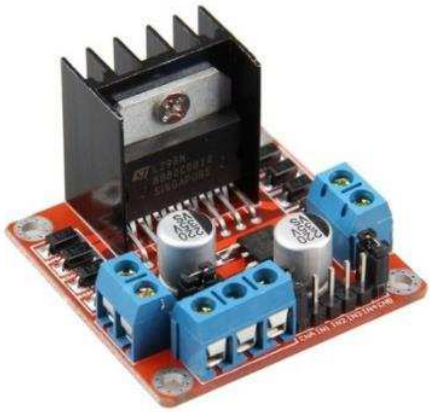
Figura 36 – *Layout* do circuito do experimento 3 implementado no ambiente Proteus.



Fonte: Autor

[44] Site: https://pt.wikipedia.org/wiki/Ponte_H

Figura 37 – Drive H L298N.



Fonte: http://www.filipeflop.com/pd-6b80a-driver-motor-ponte-h-l298n.html

O drive HL298N é um modulo projetado para controlar cargas indutivas como relés, motores DC e motores de passo. Conforme mencionado, esse circuito permite não somente o controle do sentido de rotação, mas também a velocidade com o uso do PWM[45]. As especificações desse módulo são apresentadas na Tabela 1.

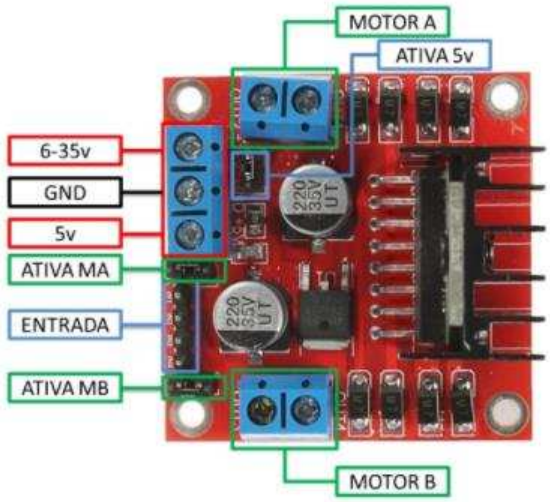
A Figura 38 ilustra as conexões presentes no circuito ponte HL298N e a Tabela 2 as correspondências entre essa pinagem e suas finalidades.

Tabela 1 – Características técnicas do drive ponte HL298N.

Características	**Especificações técnicas**
Tensão de operação	4-35 V
Controle	2 motores DC e motor de passo
Corrente de operação máxima	2A por canal ou 4A máx.
Tensão lógica	5V
Corrente lógica	0 – 36mA
Limite de temperatura	-20 a 135°C
Potência máxima	25W
Dimensões	43x43x27 mm
Peso	30g

Fonte: http://www.arduinoecia.com.br/2014/08/ponte-h-l298n-motor-de-passo.html

Figura 38 – Conexões do drive ponte HL298N.



Fonte: http://blog.filipeflop.com/motores-e-servos/motor-dc-arduino-ponte-h-l298n.html

Tabela 2 – Conectores e finalidades do drive ponte HL298N.

RECURSO / PINAGEM	FINALIDADE
Motor A e Motor B	Conectores utilizados para ligação de 2 motores DC ou 1 motor de Passo.
Ativa MA e Ativa MB	Ligados aos pinos PWM do Arduino para controle de velocidade do motor.
Ativa (5V) e (5V)	Com um regulador de tensão integrado, quando o driver estiver operando entre 6-35V o regulador oferece uma saída regulada de +5V no pino (5V) da placa Arduino para uso externo (com jumper). O pino (5V) só será uma entrada quando estiver controlando um motor com tensão de 4-5.5V (sem jumper).
(6-35V) e GND	Pinos de alimentação externa quando estiver controlado um motor que opere na tensão de 6-35V.
Entrada	Barramento composto por IN1,IN2,IN3 e IN4, são responsáveis pela rotação do motor. A (IN1 e IN2) e B (IN3 e IN4)

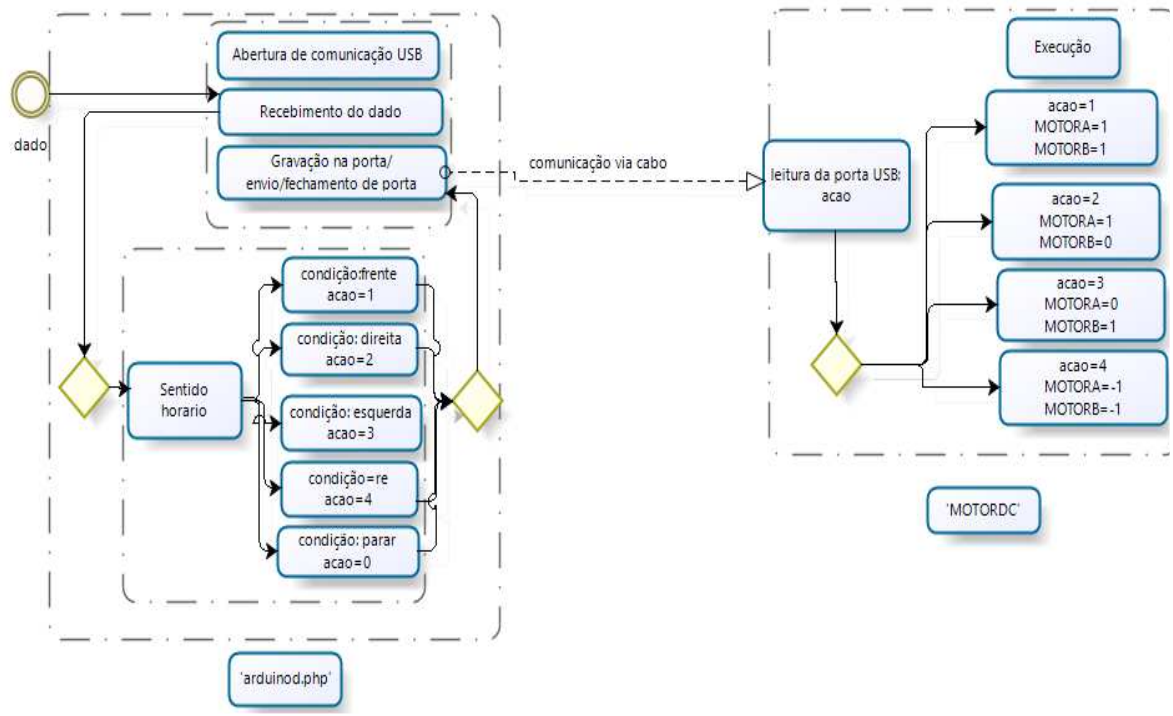
Fonte: http://www.arduinoecia.com.br/2014/08/ponte-h-l298n-motor-de-passo.html

O programa IDE do Arduino para o experimento 4 é o '**MOTORDC.c**' e seu programa em PHP é o '**arduinoc.php**' que se propõem à mesma funcionalidade no Arduino e servidor, respectivamente, aos experimentos apresentados anteriormente. Na Figura 39 apresenta-se a arquitetura de comunicação do programa '**arduinoc.php**' com o programa '**MOTORDC.c**'. Os códigos desses dois programas são apresentados nos apêndices B.3 e C.3, respectivamente.

Conforme Tabela 3, os pinos IN1 e IN2 definem o sentido de giro do eixo do motor denominado A, enquanto os pinos IN3 e IN4 o do motor B, além da condição de parada de seus eixos.

A Figura 40 apresenta a interface que o usuário acessa na máquina cliente quando escolher a opção na página principal para operar o experimento 3, programa '**experiencia3.html**' armazenado no servidor. Ao escolher uma ação nesta página, esse programa enviará sua requisição para o programa em PHP '**arduinoc.php**', cuja arquitetura é mostrada na Figura 41.

Figura 39 – Arquitetura de comunicação entre os programas '**arduinoc.php**' e '**MOTORDC**'.



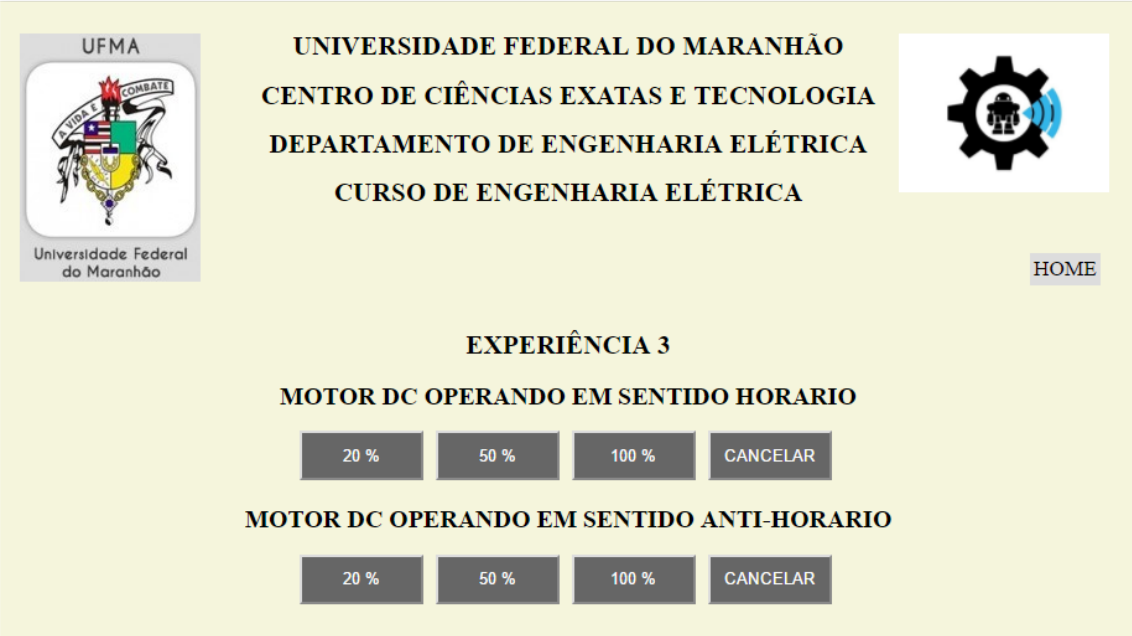
Fonte: Autor

Tabela 3 – Condições dos eixos dos motores definidas para o drive HL298N.

Motor	**IN1**	**IN2**
HORÁRIO	LIGADO	DESLIGADO
ANTI-HORÁRIO	DESLIGADO	LIGADO
PONTO MORTO	DESLIGADO	DESLIGADO
FREIO	LIGADO	LIGADO

Fonte: http://blog.vidadesilicio.com.br/arduino/modulo-ponte-h-l298n-arduino/

Figura 40 – Interface do programa – '**experiencia3.html**'.

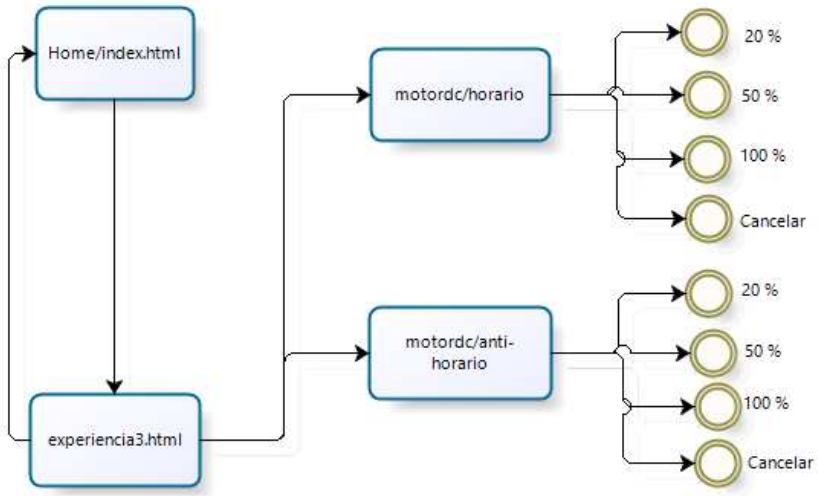


Fonte: Autor

4.4. Sistema de Controle do Robô Móvel Arduino

Para o experimento foi adquirido um pequeno robô móvel de baixo custo que utiliza como sistema embarcado uma placa Arduino. Duas ações possíveis foram programadas para o controle do robô: o tipo de movimento (frente, ré, direita e esquerda) e a leitura de um sensor sonar (onda ultrassônica) para adquirir informações de distâncias do robô a objetos localizados a sua frente, que se localizam dentro de seu cone de medição.

Figura 41 – Arquitetura do comando do programa '**arduinoc.php**'.



Fonte: Autor

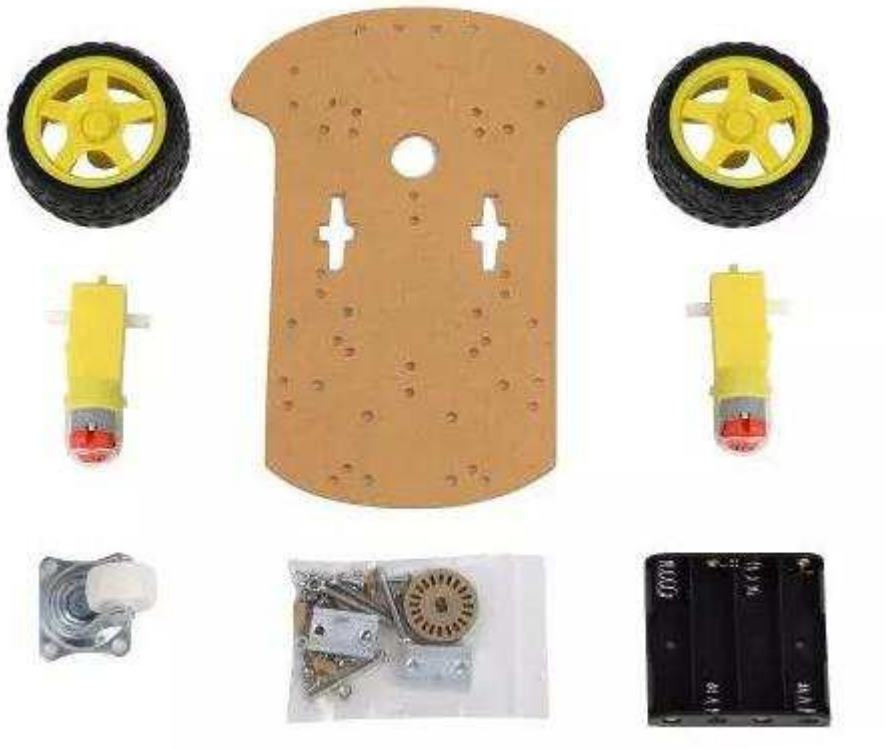
Na Tabela 4 são listados os materiais que foram usados na construção mecânica do robô e a Figura 42 mostra a imagem dos materiais que o compõe (a exceção do sonar).

Tabela 4 – Listagem dos materiais que compõem o robô móvel.

Material	**Qtd**	**Característica**
Chassi de acrílico	1	Dimensões: 22x14,7 cm
Motores DC	2	Tensão: 3-6V Eixo duplo Redução: 1:48 Peso: 30 g Corrente sem carga: ≤200mA (6V) e ≤150mA(3V) Velocidade sem carga: 200RPM (6V) e 90RPM(3V)
Roda de borracha	2	Diâmetro: 68mm Largura: 26mm Furo central: 5,3x3,66cm Peso:50 g
Rodízio giratório Roda boba	1	Diâmetro da roda: 30 mm Altura: 34 mm Peso: 32 g
Disco esconder	2	Quantidade de furos: 20 Diâmetro: 26 mm Peso: 1g
Suporte para pilhas	1	4 pilhas AA
Jogo de parafusos	1	-----

Fonte: http://www.filipeflop.com/pd-9dd47-kit-chassi-2wd-robo-para-arduino.html

Figura 42 – Materiais para montagem do robô móvel.

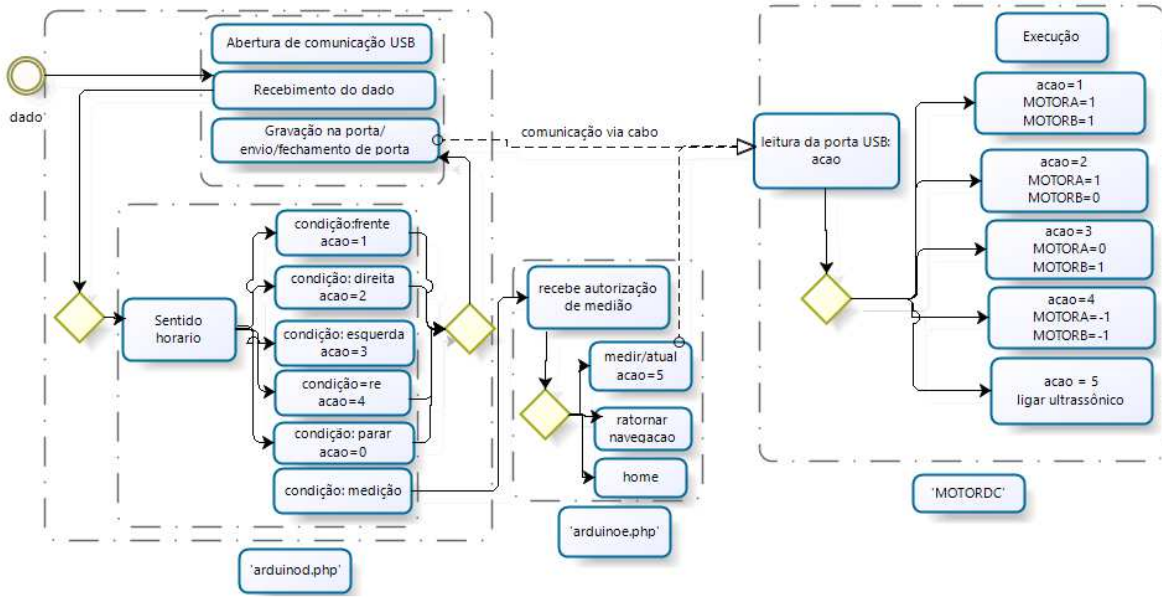


Fonte: http://produto.mercadolivre.com.br/MLB-854824174-kit-rob-segue-faixa-arduino-chassi-2wd-_JM

A direção do robô será determinada pelo cliente via interface desenvolvida pelo programa '**experiencia4.html'**, apresentada mais adiante. Os comandos do cliente serão repassados para o programa '**arduinod.php**' no servidor (ambos os códigos no apêndice B.4). O software embarcado do robô na placa Arduino é o '**ROBO_ULTRASSONICO.c**' (apêndice C.4). Para o controle da velocidade e rotação dos motores de tração DC do robô usou-se o mesmo drive do experimento 3 (Seção 4.3). Na Figura 43 é apresentada na forma de fluxograma a arquitetura de comunicação entre os códigos do servidor e do Arduino embarcado no robô.

Para o sistema de medição foi desenvolvido o programa '**arduinoe.php**' (código no apêndice B.5) para execução no servidor. O código no Arduino é o mesmo de movimento do robô, que também procede a pedidos de leitura de distância pelo sonar. O *software* embarcado no Arduino faz o cálculo da distância entre o sensor (localizado na frente do robô) e o obstáculo que detecta através do tempo de voo da onda ultrassônica. A detecção do obstáculo é possível, caso o mesmo se encontre dentro do cone sólido do trem de pulso ultrassônico emitido pelo sonar, situação em que o sinal é retornado ao transdutor. A Academia de Ciências Parisiense propôs que a velocidade do som no SI (Sistema Internacional de Unidades) é de 343

Figura 43 – Arquitetura do controle navegação com o sistema de medição do sensor ultrassônico.



Fonte: Autor

m/s a uma temperatura de 20°C. A Figura 44 apresenta a velocidade do som em alguns meios ambientes em determinadas temperaturas[46].

O cálculo da distância do sonar ao obstáculo é baseado na Eq. 1. O Fator 2 no denominador da referida equação considera que o tempo de ida e retorno da onda ultrassônica são os mesmos[47]:

$$distância = \frac{velocidade\ do\ som * tempo\ de\ voo}{2} \qquad \text{(Eq. 1)}$$

Existem várias opções de sensores ultrassônicos disponíveis no mercado. O dispositivo escolhido foi HC-SR04 pelo fator econômico e por atender as condições para o trabalho desenvolvido pelo robô Arduino nessa monografia. A Figura 45 mostra o sensor o ultrassônico HC-SR04 e a Figura 46 ilustra o envio e a recepção do sinal ultrassónico do sensor HC-SR04 em 2D. A Tabela 5 cita as suas especificações técnicas.

[46] Site: http://www.estudopratico.com.br/a-velocidade-do-som/
[47] Site: http://blog.fazedores.com/sensor-ultrassonico-com-arduino/

Figura 44 – Velocidade de propagação do som em alguns materiais.

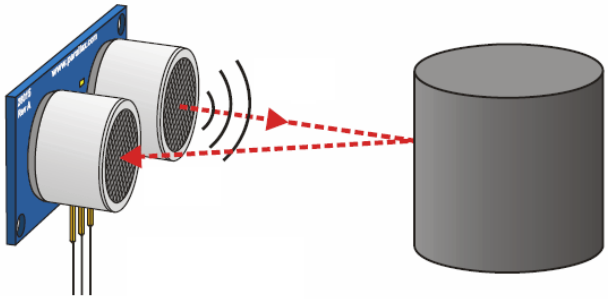
MATERIAL	VELOCIDADE DE PROPAGAÇÃO DO SOM (m/s)
Ar (10°C)	331
Ar (20°C)	343
Ar (30°C)	350
Oxigênio	317
Dióxido de Carbono	250
Água	1480
Água do Mar	1522
Borracha	54
Alumínio	4420
Aço	6000
Betão	5000
Latão	3500

Fonte: http://www.estudopratico.com.br/a-velocidade-do-som/

Figura 45 – Sensor ultrassônico HC-SR04.

Fonte: http://produto.mercadolivre.com.br/MLB-746568590-sensore-ultrassnico-de-distancia-p-arduinopic-hc-sr04- JM

Figura 46 – Ilustração do envio e da recepção do sinal ultrassónico do sensor HC-SR04.



Fonte: http://blog.vidadesilicio.com.br/arduino/sensor-ultrassom-hc-sr04/

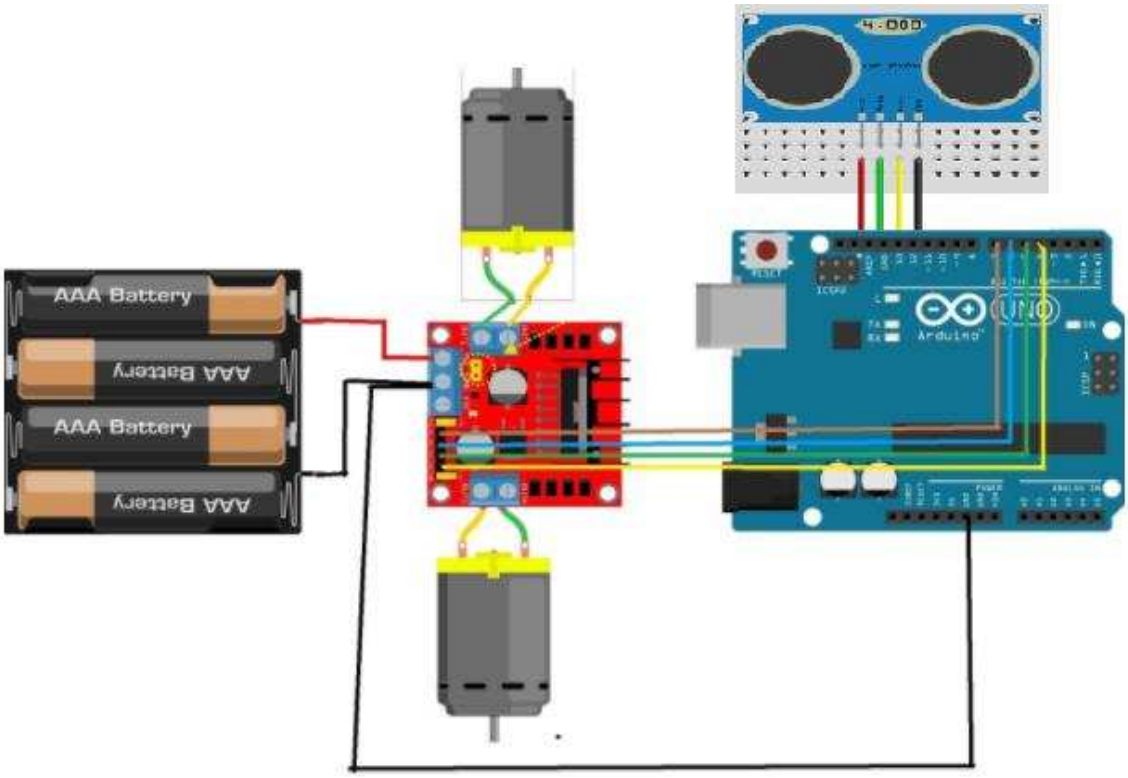
Tabela 5 – Especificações técnicas do sonar HC-SR04.

Características	**Especificações técnicas**
Tensão	5V
Corrente	2mA
Leitura	PWM
Frequência ultrassônica	50Hz
Faixa de detecção	2cm a 5 cm
Resolução	3mm

Fonte: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit

Na Figura 47 montou-se um esquema eletrônico do robô Arduino para um melhor entendimento das conexões de seus componentes.

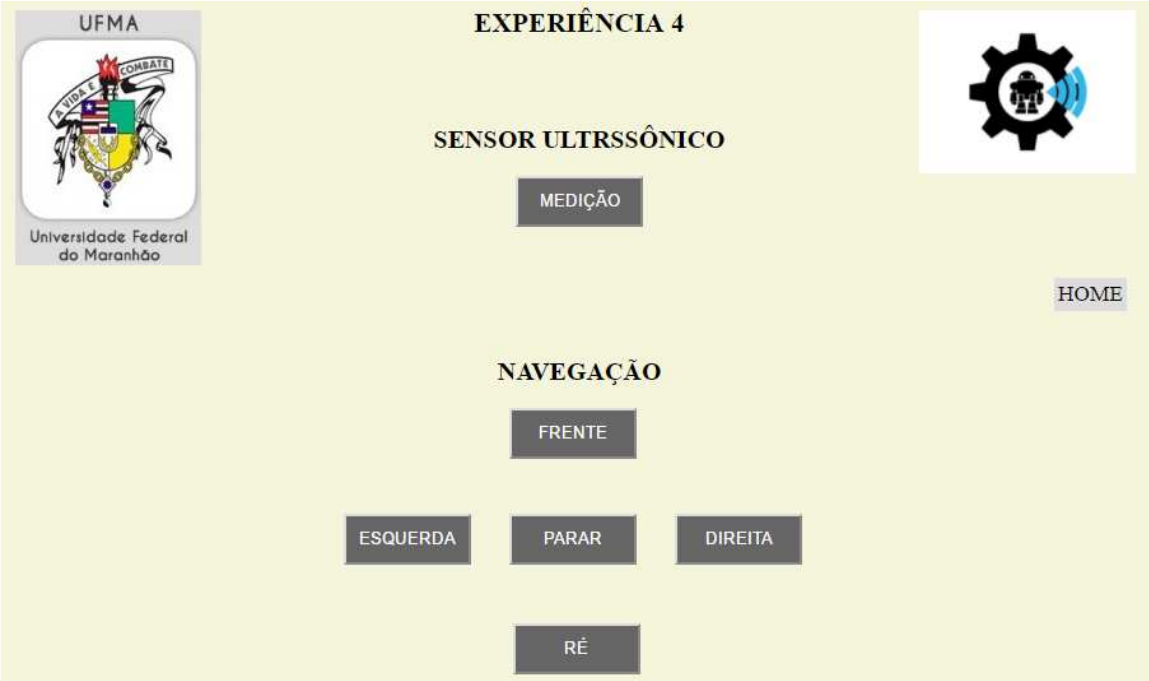
Figura: 47 – Estrutura eletrônica do robô móvel.



Fonte: Autor

A experiência 4 realiza o controle de movimento do robô Arduino, através de uma interface apropriada (*joystick* digital) que comandará os quatro possíveis movimentos do robô: frente, ré, direita e esquerda, conforme pode-se perceber na Figura 48. Um outro botão, ao centro da interface, faz o movimento em andamento cessar. Na posição acima do controle de navegação, é disponibilizado o botão que acessa a medição de distância com o sensor ultrassônico HC-RS04.

Figura 48 – Interface do programa - '**experimento4.html**'.



Fonte: Autor

Quando for acionado, uma página de interface auxiliar (Figura 49) é apresentada ao usuário, contendo os botões medir, retornar navegação (página origem) e home (página inicial). Para solicitar e atualizar a medição da distância pelo sonar localizado a frente do robô, basta acionar o botão medir.

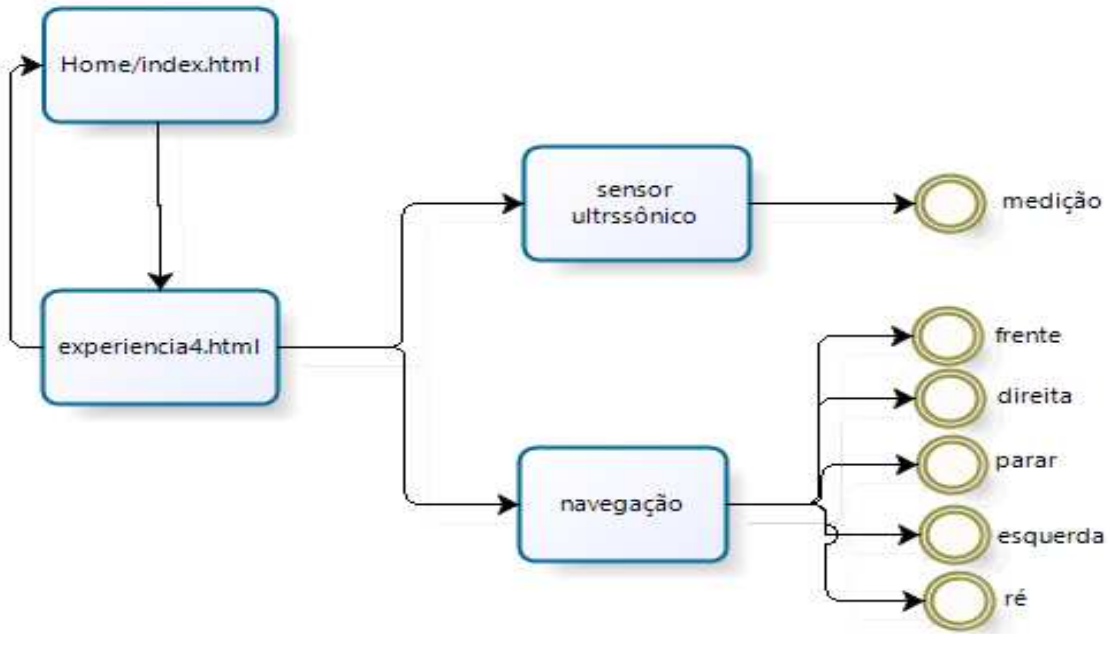
O programa que executa no servidor é o '**arduinod.php**' (Figura 50) que processará as informações da interface dos botões do controle de navegação e enviará a ação correspondente para os motores do robô se movimentar na direção pedida.

Para o processo de medição com o sensor ultrassônico, o programa executor é o '**arduinoe.php**' que atuará na solicitação da página auxiliar para medição, conforme fluxograma mostrado na Figura 51.

Figura 49 – Interface do programa auxiliar para medição (‘**arduinoe.php**’).

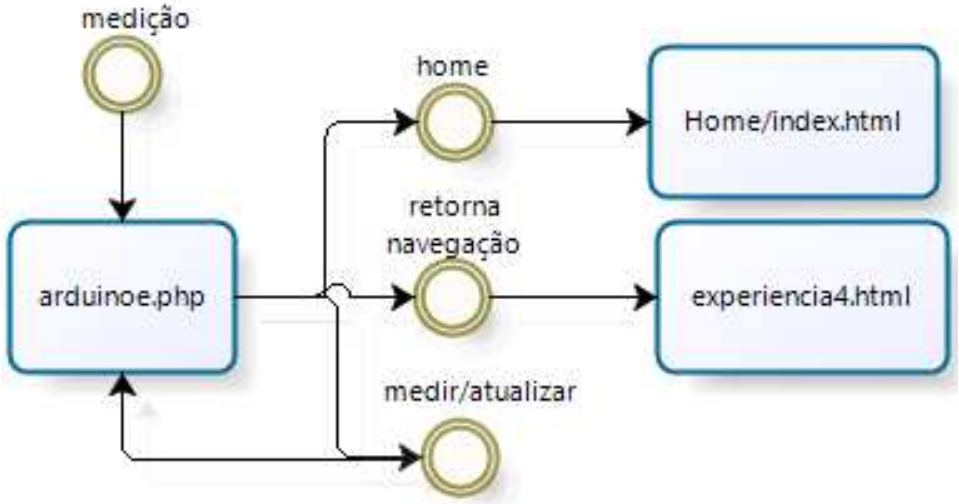
Fonte: Autor

Figura 50 – Arquitetura do programa ‘**arduinod.php**’.



Fonte: Autor

Figura 51 – Arquitetura do comando botão medição para o programa '**arduinoe.php**'.



Fonte: Autor

5. EXPERIMENTOS E RESULTADOS

Para a realização e apresentação dos testes no controle e monitoração de sistemas embarcados através da topologia cliente-servidor são apresentados os experimentos propostos separadamente em cada seção desse capítulo, considerando a seguinte estrutura:

a) A configuração cliente-servidor-experimento;
b) Testes de acesso ao sistema via página cliente, considerando a rede local com conexão apenas do experimento (sem nenhuma outra conexão), referido como tráfego sem carga ou 'rede livre'; e
c) Teste de acesso ao sistema via página cliente, considerando a rede local com atividades regulares, intensificado seu tráfego através da realização de *downloads* e acesso à sites com apresentação de vídeo *online* por outras máquinas do laboratório, referido como 'tráfego intenso'.

Na configuração é possível visualizar, por fotos cada experimento testado, incluindo as máquinas cliente e servidor, além do experimento que faz uso do sistema embarcado Arduino, conforme detalhados nos Capítulos 4 e 5.

Os testes forma elaborados com objetivo de avaliar o desempenho da rede local, no caso a que possui um SSID (*Service Set IDentifier*) denominado LMI no roteador *wifi* existente no LRC (Laboratório de Robótica Móvel e Comunicação Sem Fio). Esses testes foram divididos em dois grupos para cada experimento proposto, diferenciando no que diz respeito à intensidade do trafego na rede local. Foram escolhidas para os testes duas situações consideradas: apenas com conexão do experimento e com a rede com tráfego intensificado. No primeiro teste, apenas o servidor estava conectado à rede em um dia em que a UFMA estava sem atividades e, portanto, com o tráfego livre. O segundo tipo de teste ocorreu em um dia normal de trabalho na UFMA, com as atividades de rede com fluxo rotineiro, tendo sido conectadas outras máquinas à rede LMI para fazer *downloads* e acesso à sites com apresentação de vídeo *online*, todas essas atividades feitas em site de alto desempenho (youtube).

Para a obtenção do tempo de resposta observados em ambos os tipos de testes, foi usado um cronômetro que era disparado na ação pedida na página cliente (botão acionado) e parado na observação visual do controle efetuado no sistema embarcado (ação de controle) ou no retorno à página cliente (ação de monitoramento). Foram feitos 25 testes para cada botão dos quatro experimentos e extraídos o tempo médio e o desvio padrão usando os comandos

‘**std(tempo)**’ e ‘**mean(tempo)**’, respectivamente, existentes no MATLAB. Ambos os valores para cada botão das páginas dos experimentos foram disponibilizados em formato de tabelas.

Antes da apresentação dos tempos obtidos para os quatro experimentos, é importante destacar que em todos os testes houve a ação no sistema embarcado prevista a partir da ação no respectivo botão da página cliente, não apresentando nenhuma falha para os dois tipos de testes. A exceção ocorreu na medição do sensor ultrassônico que, em raros casos, não retornou a distância medida quando pedida. Esse problema está relacionado a ajustes no código C no Arduino que faz a leitura do dispositivo.

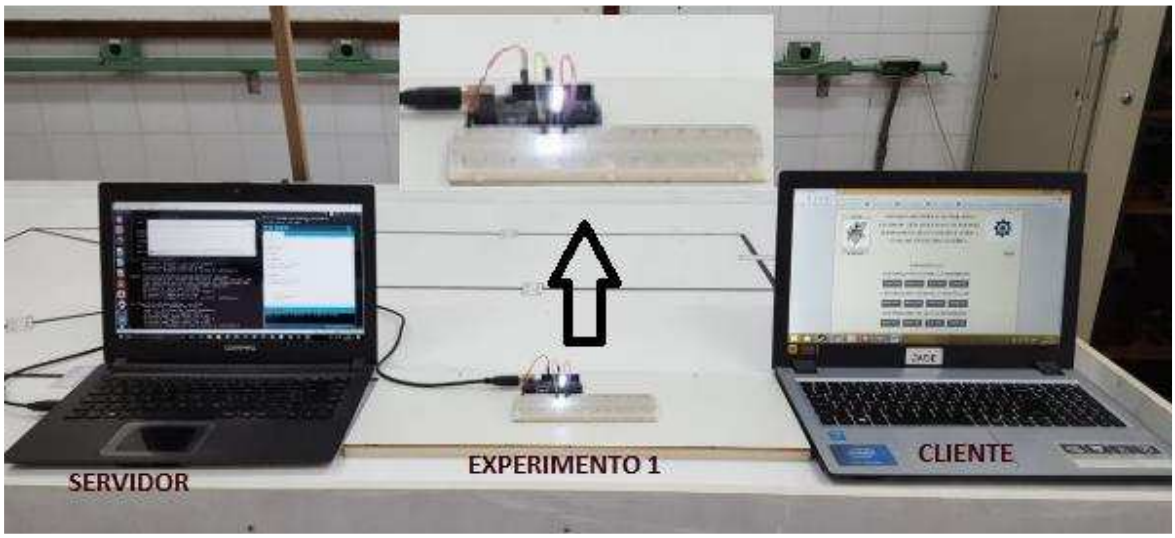
5.1. Resultados do Acionamento Liga/Desliga de um *LED*

A configuração do experimento 1 é mostrada na foto da Figura 51, onde observam-se as máquinas cliente e servidor, além do sistema embarcado Arduino (ligado ao servidor via cabo), controlando o acendimento de um *led* por sinal PWM.

No primeiro tipo de teste, onde apenas a configuração do experimento esteve conectada à rede local LMI, foram obtidos os resultados apresentados na Tabela 6.

No segundo tipo de teste, onde além do experimento estiveram conectadas à mesma rede LMI outras máquinas intensificando o tráfego, foram obtidos os resultados mostrados na Tabela 7.

Figura 52 – Experimento 1: Liga/Desliga *led*.



Fonte: Autor

Tabela 6 – Testes do experimento 1 com a rede local livre.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Luminosidade Baixa – Frequência Baixa	308	38,9
Luminosidade Baixa – Frequência Média	238	48,9
Luminosidade Baixa – Frequência Alta	224	29,3
Luminosidade Média– Frequência Baixa	212	30,6
Luminosidade Média – Frequência Média	178	4,1
Luminosidade Média – Frequência Alta	180	0
Luminosidade Alta– Frequência Baixa	170	20,4
Luminosidade Alta – Frequência Média	160	25,0
Luminosidade Alta – Frequência Alta	170	32,9

Fonte: Autor

Tabela 7 – Testes do experimento 1 com a rede local com tráfego intenso.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Luminosidade Baixa – Frequência Baixa	356	167,9
Luminosidade Baixa – Frequência Média	372	106,1
Luminosidade Baixa – Frequência Alta	310	102,1
Luminosidade Média– Frequência Baixa	420	169,2
Luminosidade Média – Frequência Média	442	233,0
Luminosidade Média – Frequência Alta	318	155,6
Luminosidade Alta– Frequência Baixa	588	206,1
Luminosidade Alta – Frequência Média	472	159,5
Luminosidade Alta – Frequência Alta	352	74,6

Fonte: Autor

Uma primeira constatação nos experimentos feitos neste e demais experimentos para o caso do primeiro tipo de teste, conforme serão vistos nas tabelas correspondentes, o tempo médio da reação da ação ao pedido feito da máquina cliente é muito baixo, quase desprezível, quando se considera que a operação do cronometro foi feita manualmente. Esse

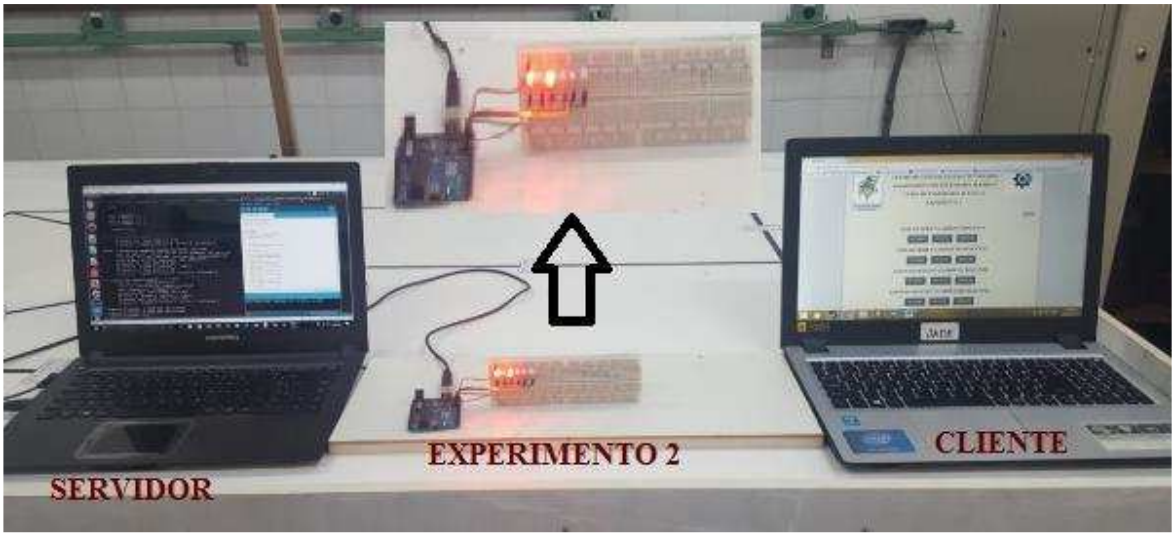
resultado já era esperado para todos os experimentos, já que esses testes foram realizados em um dia sem atividades na rede da UFMA, portanto estando conectado basicamente o experimento. O valor do desvio padrão, ainda que em alguns testes chegaram a ser mais de 20% do tempo médio de atraso e em um deles nulo, pode ser explicado pela ação manual do cronômetro.

No segundo tipo de teste, com a rede da UFMA operando com uma carga de tráfego intenso, os tempos mostraram-se um pouco maiores. Nesse tipo de teste, realizado para todos os experimentos, foi feita uma verificação da taxa de transferência de *download* que apresentou o valor próximo de 1M Bits/s, o que explica os tempos de atrasos maiores obtidos, porém ainda realistas para atividades práticas do uso dessa proposta em um laboratório. Essa performance da rede do LMI nem sempre acontece, pois em momentos anteriores de atividades do laboratório já haviam sido percebidos atrasos bem mais significativos, tanto em *downloads* quanto em atividades práticas da disciplina que utiliza a placa Raspberry Pi 3 (recurso wifi) nas aulas práticas da disciplina TEEE – SISTEMAS EMBARCADOS.

5.2. Resultados do Acionamento e Controle de um *Array* de *LEDS*

A configuração do experimento 2 é mostrada na foto da Figura 52, onde observa-se as máquinas cliente e servidor, além do sistema embarcado Arduino controlando o acendimento de um conjunto de 6 *leds* por acionamento simples (liga/desliga).

Figura 53 – Experimento 2: Acionamento de *array* de *leds.*



Fonte: Autor

No primeiro tipo de teste, onde apenas a configuração do experimento esteve conectada à rede local LMI, foram obtidos os resultados apresentados na Tabela 8.

Tabela 8 – Testes do experimento 2 com a rede local livre.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Leds em serie ordem crescente: Velocidade Baixa	178	29
Leds em serie ordem crescente: Velocidade Alta	178	4
Leds em serie ordem decrescente: Velocidade Baixa	188	44
Leds em serie ordem decrescente: Velocidade Alta	188	18
Leds em paralelo ordem crescente: Velocidade Baixa	188	16
Leds em paralelo ordem crescente: Velocidade Alta	180	0
Leds em paralelo ordem decrescente: Velocidade Baixa	204	20
Leds em paralelo ordem decrescente: Velocidade Alta	168	20

Fonte: Autor

No segundo tipo de teste (trafego da rede intensificado), foram obtidos os resultados mostrados na Tabela 9.

As mesmas justificativas apresentadas para ambos os testes 1 e 2 feitas no experimento 1 são válidas para os tempos deste experimento.

Tabela 9 – Testes do experimento 2 com a rede local com tráfego intenso.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Leds em serie ordem crescente: Velocidade Baixa	554	196
Leds em serie ordem crescente: Velocidade Alta	404	72
Leds em serie ordem decrescente: Velocidade Baixa	520	108
Leds em serie ordem decrescente: Velocidade Alta	562	210
Leds em paralelo ordem crescente: Velocidade Baixa	386	78
Leds em paralelo ordem crescente: Velocidade Alta	440	77
Leds em paralelo ordem decrescente: Velocidade Baixa	412	70
Leds em paralelo ordem decrescente: Velocidade Alta	464	81

Fonte: Autor

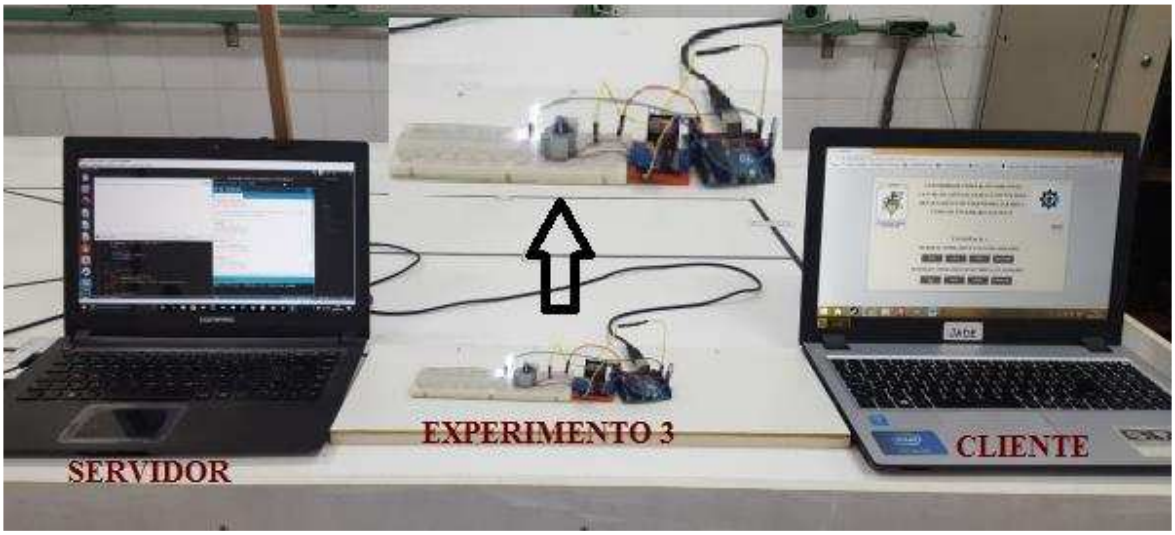
5.3. Resultados do Acionamento do Motor CC

A configuração do experimento 3 é mostrada na foto da Figura 53, onde observa-se as máquinas cliente e servidor, além do sistema embarcado Arduino controlando o acionamento do giro e velocidade de um motor DC.

No primeiro tipo de teste, onde apenas a configuração do experimento esteve conectada à rede local LMI, foram obtidos os resultados apresentados na Tabela 10.

No segundo tipo de teste foram obtidos os resultados mostrados na Tabela 11.

Figura 54 – Experimento 3: Acionamento do Motor DC.



Fonte: Autor

Tabela 10 – Testes do experimento 3 com a rede local livre.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Motor DC operando em sentido horário: 20% de velocidade	214	51
Motor DC operando em sentido horário: 50% de velocidade	188	16
Motor DC operando em sentido horário: 100% de velocidade	206	36
Motor DC operando em sentido anti-horário: 20% de velocidade	192	16
Motor DC operando em sentido anti-horário: 50% de velocidade	188	16
Motor DC operando em sentido anti-horário: 100% de velocidade	186	18

Fonte: Autor

Tabela 11 – Testes do experimento 3 com a rede local com tráfego intenso.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Motor DC operando em sentido horário: 20% de velocidade	704	123
Motor DC operando em sentido horário: 50% de velocidade	408	71
Motor DC operando em sentido horário: 100% de velocidade	632	13
Motor DC operando em sentido anti-horário: 20% de velocidade	502	36
Motor DC operando em sentido anti-horário: 50% de velocidade	524	62
Motor DC operando em sentido anti-horário: 100% de velocidade	492	12

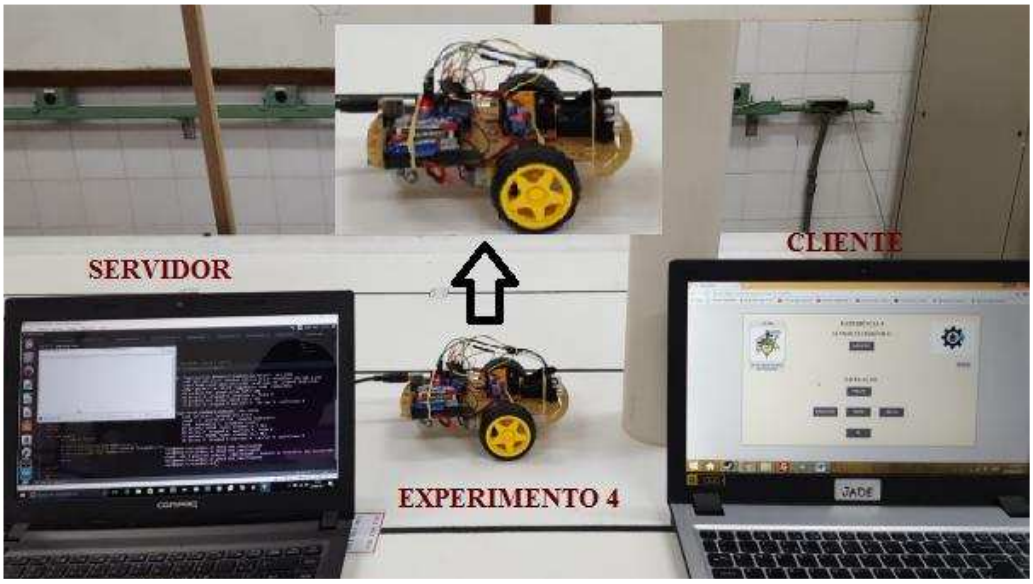
Fonte: Autor

Novamente os resultados dos testes do tipo 1 (rede livre) e do tipo 2 (rede com tráfego intensificado) foram coerentes com o esperado. Uma observação a ser feita neste experimento é que a observação visual do início do giro do eixo do motor foi usada para interromper a contagem do tempo. Caso fosse um motor de potência bem maior que o utilizado, devido ao tempo para vencer a inércia, os tempos medidos apresentariam aumentos. Como o motor DC empregado foi de potência muito baixa e, assim, o efeito foi o mesmo do acendimento de um led.

5.4. Resultados com Robô Móvel Arduino

A configuração do experimento 4 é mostrada na foto da Figura 54, onde observa-se as máquinas cliente e servidor, além do sistema robótico móvel com placa Arduino, onde os elementos são destacados por setas.

Figura 55 – Experimento 4: Robô Móvel Arduino.



Fonte: Autor

No primeiro tipo de teste (rede livre), foram obtidos os resultados apresentados na Tabela 12.

Tabela 12 – Testes do experimento 4 com a rede local livre.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Robô navegando para a frente	274	77
Robô navegando para a direita	242	83
Robô navegando para a esquerda	192	24
Robô navegando para trás	222	51
Medição do sensor ultrassônico	850	10

Fonte: Autor

No segundo tipo de teste (tráfego intensificado) foram obtidos os resultados mostrados na Tabela 13.

A única diferença nos tempos observados para ambos os tipos de testes, deu-se na requisição de medição da distância do robô ao obstáculo à sua frente feita pelo sonar, já que a resposta para o retorno dessa informação à página cliente depende do tempo de espera do Arduino pela onda ultrassônica. Em ambos os tipos, quando comprados com botões de ação de

Tabela 13 – Testes do experimento 4 com a rede local com tráfego intenso.

Comando do Cliente	**Tempo médio resposta (ms)**	**Desvio Padrão (ms)**
Robô navegando para a frente	570	62
Robô navegando para a direita	528	111
Robô navegando para a esquerda	852	75
Robô navegando para trás	572	187
Medição do sensor ultrassônico	1236	190

Fonte: Autor

movimento, eles são maiores, como era esperado. Além disso, percebe-se uma diferença percentual correspondente entre a leitura do sensor nos testes tipos 1 e 2.

De uma forma geral, os resultados apresentados, considerando-se o pior caso do tráfego da rede (tese tipo 2), os tempos ainda são aceitáveis para os propósitos de experimentos laboratoriais, os quais não possuem requisitos tempo real, como são os casos dos experimentos implementados neste trabalho. Mesmo que fossem envolvidos tais requisitos, há que se considerar se esses tempos estariam relacionados diretamente às ações disparadas no *browser*, já que uma rede por mais veloz que possa responder, terá um retardo que poderia comprometer o tempo requisitado.

6. CONCLUSÕES E TRABALHOS FUTUROS

No mundo moderno, as ações de controle e monitoração de sistemas nas mais diversas áreas como a indústria, o comércio, os serviços, o entretenimento e principalmente a doméstica, estão se tornando cada vez mais intensas com o avanço e oferta de produtos tecnológicos utilizados na implementação de tais sistemas. Além disso, com o crescimento acentuado das tecnologias empregadas na estruturação de redes de computadores mundiais (internet) essas ações, antes locais e fisicamente executadas, passaram a ser uma realidade feita remotamente e, portanto, fora do local onde se encontram os sistemas.

A proposta implementada neste trabalho de monografia possibilitou realizar essas ações de controle e monitoração de sistemas embarcados (Arduino) em uma rede local, usando a topologia cliente-servidor. Ainda que restrita a uma rede local, o trabalho desenvolvido permitiu a estruturação modular de programas para ser usado no ensino práticos nos laboratórios locais de universidades, em especial o LRC existente no Curso de Engenharia Elétrica da UFMA.

No intuito de validar a proposta foram realizados dois tipos de testes em situações consideradas como extremas em relação ao tráfego da rede local disponível no LRC. Em ambos os testes, a partir de uma máquina cliente foram mensurados tempos entre a ação disparada na máquina cliente e a reação visual ao pedido nos quatros experimentos desenvolvidos no sistema Arduino. O primeiro teste, considerado como de rede livre, basicamente o servidor Apache esteva conectado. No segundo, considerado como de tráfego intensificado, foram feitas outras conexões à mesma rede local (*downloads* e acesso a sites *online*).

. Os resultados desses tempos ocorrerão todos dentro do previsto. No primeiro tipo de teste os tempos foram desprezíveis, o que já era esperado. No segundo tipo, os tempos ainda que maiores que os do anterior, são considerados baixos (menor que 1 segundo) em função da rede apresentar bom desempenho, mesmo tendo seu tráfego sido intensificado com outras atividades. Testes de *downloads* indicaram uma taxa próxima a 1 M Bits/s nesta condição de rede.

Alguns trabalhos futuros que podem ser citados para serem conduzidos de forma imediata, a partir do atual estágio alcançado nas implementações realizadas neste trabalho, são:

a) A inserção dos quatros experimentos implementados, executando concomitantemente no sistema embarcado Arduino, devendo o servidor informar inicialmente ao Arduino qual é o experimento que entrará em execução e prosseguir com o seu controle/monitoração como proposto. Esta possibilidade é viável quanto a recursos no Arduino, tanto ao nível de *hardware* quanto de *software*;
b) A expansão da topologia cliente-servidor proposta para operar com experimentos com maior grau de complexidade, tano ao nível do controle quanto da monitoração, usando o Arduino ou um sistema embarcado mais robusto em recurso de *hardware* e de capacidade de processamento, como é o caso do Raspberry Pi 3; e
c) A expansão da estrutura de controle para acesso remoto, estando a máquina cliente fora da rede do servidor (outra rede local), fazendo uso da internet.

Referências Bibliográficas

ALBUQUERQUE, Fernando. **TCP/IP Internet Protocolo e Tecnologia**, 2001. 3ª ed.

ANDROUTSELLIS, Theotokis, S.; SPINELLIS, D. **A Survey of Peer-To-Peer Content Distribution Technologies**, 2004.
Disponível em: <https://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.pdf>. Acesso em: 10/maio/2017.

BOCHENSKI, Barbara. **Implementando Sistemas Cliente/Servidor de Qualidade,** 1995. 1ª ed.

COMER, Douglas E. **Computer Networks and Internets with Internet Applications,** 2009. Disponível em: <https://web.njit.edu/~bm245/IT420/Computer-Networks-and-Internets-Fifth-Edition.pdf>. Acesso em: 18/maio/2017

CANÊDO, Igor Berquó. **Sistema WEB para os Cursos de Extensão,** 2015.
Disponível em:<http://monografias.poli.ufrj.br/monografias/monopoli10014571.pdf>. Acesso em: 15/abril/2017

CARVALHO, João Antônio. Informática para Concursos,2006. 3ª ed.

CAVALCANTE, S. **Introdução aos sistemas embarcados**, 2005. Apostila. UFPE.
Disponível em:<http://www.cin.ufpe.br/~vba/periodos/8th/s.e/aulas/STP%20-20Intro%20Sist%20Embarcados.pdf >. Acesso em: 12/jun/2017.

CÔRTE, Leandro. **Método para a Avaliação de Servidores WWW no Ambiente Corporativo**, 2002. Disponível em:< http://www.lume.ufrgs.br/handle/10183/3375>. Acesso em: 17/fev/2017.

CUNHA, A. **O que são sistemas embarcados?,** 2013. Apostila. Disponível em: <http://www.techtraining.eng.br/files/uploads/2013/04/19/artigo-sist-emb.pdf>. Acesso em: 28/jan/2017.

CHASE, O. **Sistemas embarcados**, 2007. Disponível em: <http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>. Acesso em: 20/jan/2017.

FOROUZAN, Behrouz A. Comunicação de Dados e Rede de Computadores, 2010. 4ª ed.
.
FRANCISCATTO, Roberto. **Redes de computadores**, 2014. Disponível em: < http://estudio01.proj.ufsm.br/cadernos/cafw/tecnico_informatica/redes_computadores.pdf>.Acesso em: 14/fev/2017.

KUROSE, J. F; ROSS, K. W. **Rede de Computadores e a Internet: uma abordagem top-down.** 2013. 6ª ed.

LINHARES, Edson Sousa. **Tópicos Avançados. Comparativos Entre Os Servidores Web: APACHE E IIS**, 2014. Disponível em:< http://revistas.ung.br/index.php/computacaoaplicada/article/view/1950/1545>. Acesso em: 12/jan/2017.

MARTINS, Ronei Ximenes. **Introdução a Redes de Computadores**, 2002. Disponível em:< http://bertozi.com/adb/PITAGORAS/Sistemas/Gerencia_de_Redes/Introducao%20Redes%20Computadores.pdf>. Acesso em: 3/mar/2017.

MORIMOTO, Carlos E. **Redes e Servidores Linux: Guia Prático**, 2005. Disponível em:< http://www.jovemdigital.virtual.ufc.br/downloads/redes/apostila_redes.pdf>. Acesso em: 17/jan/2017.

NEDELCU, Clément. **Nginx HTTP Server**, 2013. Disponível em:< http://www.gocit.vn/files/Nginx.HTTP.Server-www.gocit.vn.pdf>. Acesso em: 27/abr/2017.

NIEDERAUER, Juliano. **PHP para quem conhece PHP**. 2008. 1ª ed.

REZENDE, Evandro da Silva. **Modelo Estrutural para Compartilhamento de Arquivos Peer-to-Peer**, 2009. Disponível em:< https://repositorio.unesp.br/handle/11449/98698>. Acesso em: 25/jan/2017.

ROCHA, Carlos Andre de Sousa. **Análise de Desempenho em Ambientes Cliente/Servidor 2-camadas e 3-camadas**, 2002. Disponível em:< https://repositorio.ufsc.br/bitstream/handle/123456789/84180/190171.pdf?sequence=1>. Acesso em: 17/jan/2017.

TANENBAUM, A. S. **Redes de Computadores**, 2011.5 ed.

TORRES, Gabriel. **Rede de Computadores**, 2014. 2ª ed.

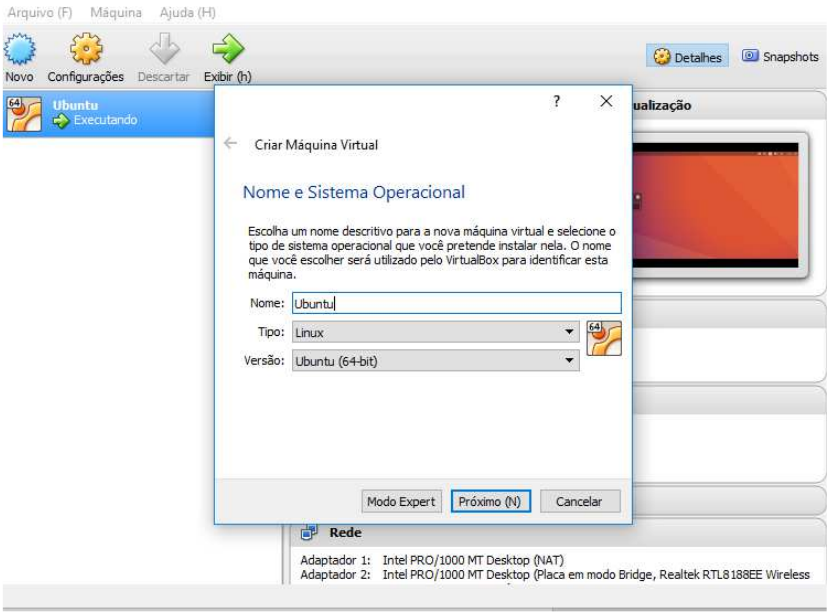
APÊNDICE A – INSTALAÇÃO E CONFIGURAÇÃO DO SERVIDOR APACHE

A.1 - INSTALAÇÃO DA MAQUINA VIRTUAL

Para o desenvolvimento desse trabalho de monografia foi necessária a instalação de um servidor virtual Linux usando o Ubuntu Desktop como servidor *web*, sendo a versão usada a 16.10, baixada gratuitamente em '**www.ubuntu.com**'**.** O software usado para a virtualização foi o VirtualBox, cujo download livre é feito de '**www.virtulbox.com**'.

Com o VirtualBox instalando no terminal, clicou-se na opção '**Novo**' para instalar o servidor virtual. Nessa opção (Figura A.1) foi colocado o nome do servidor e o sistema operacional, no caso, Linux de 64 bits.

Figura A.1: Processo de instalação do Ubuntu no VirtualBox



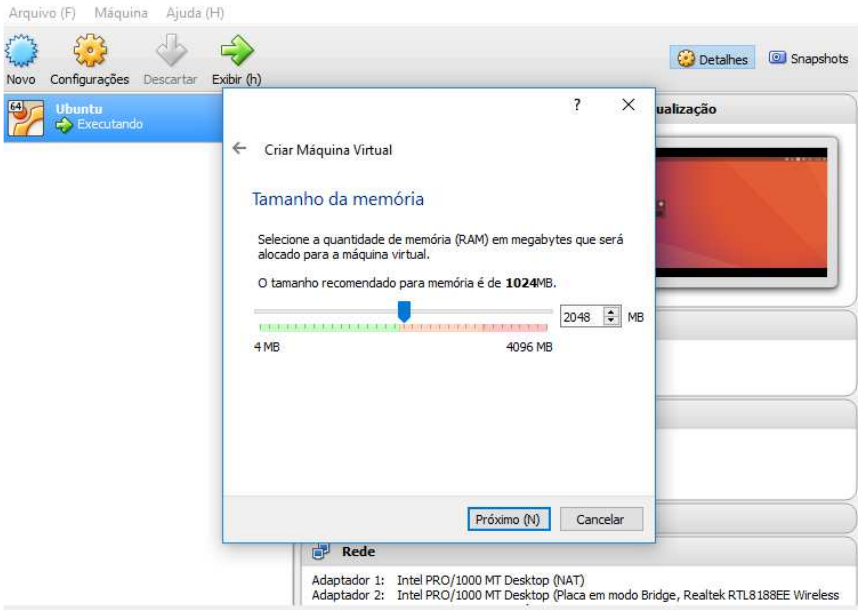
Fonte: Autor

Na Figura A.2 definiu-se o tamanho da memória RAM do servidor.

A Figura A.3 mostra a configuração do espaço do disco rígido que foi disponibilizado para a máquina virtual operar.

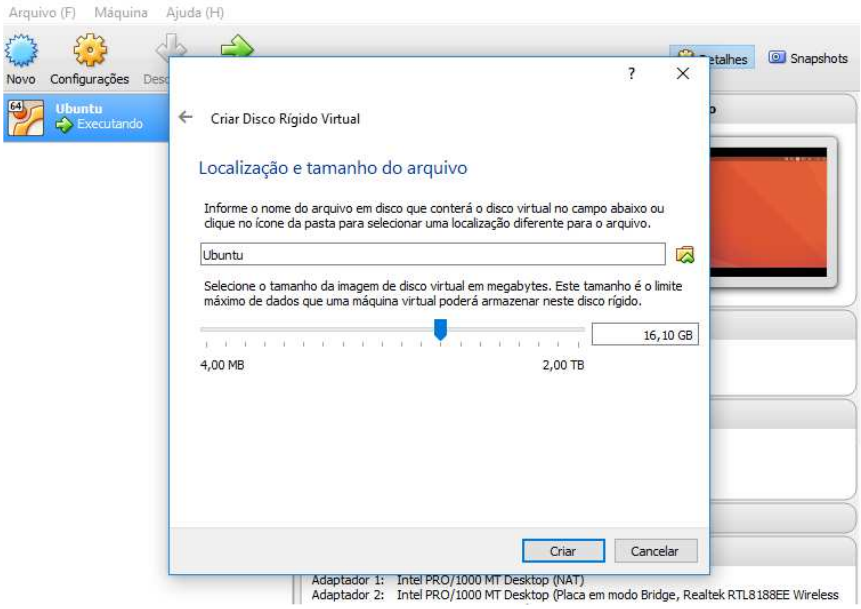
Em seguida foram feitas as alterações na configuração de rede, pelo terminal em configuração, na opção '**Rede**' (Figura A.4).

Figura A.2: Configuração da memória RAM



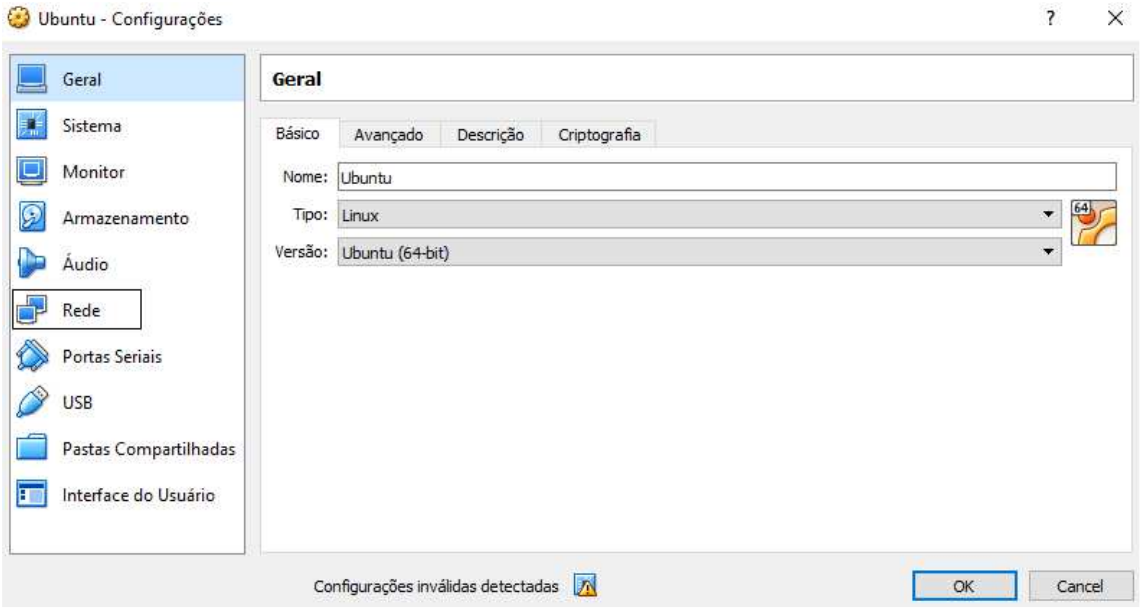
Fonte: Autor

Figura A.3: Configuração do espaço do disco rígido



Fonte: Autor

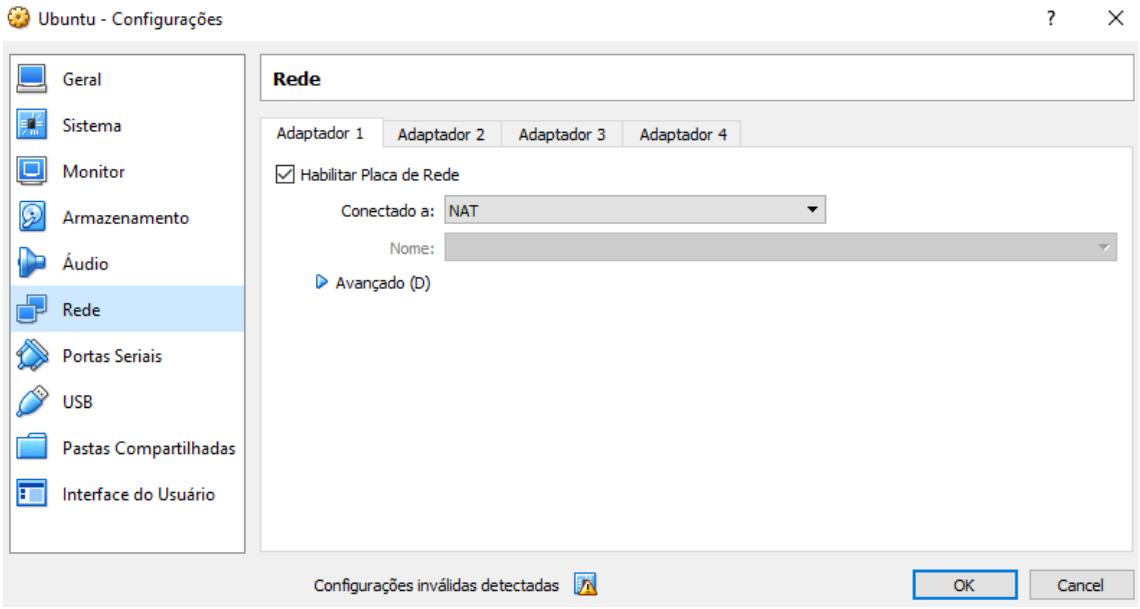
Figura A.4: Configuração das interfaces da Rede



Fonte: Autor

O primeiro adaptador ficou na placa NAT (*Network Address Translation*), fazendo o VirtualBox agir como roteador para que o servidor se comunique com a internet de maneira isolada (Figura A.5)

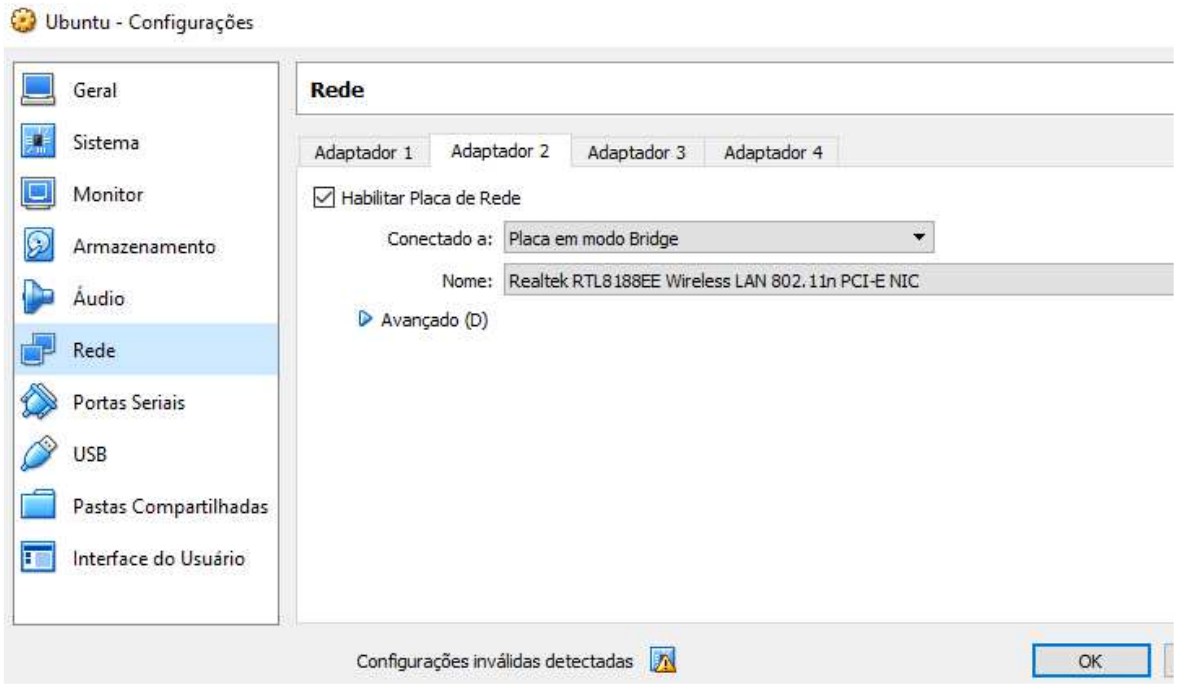
Figura A.5: Adaptador 1 configurado como NAT



Fonte: Autor

O adaptador foi configurado para o modo Bridge para que o servidor possa ter a comunicação de dados diretamente de uma rede externa, recebendo um IP (Figura A.6).

Figura A.6: Adaptador 2 configurado como Bridge

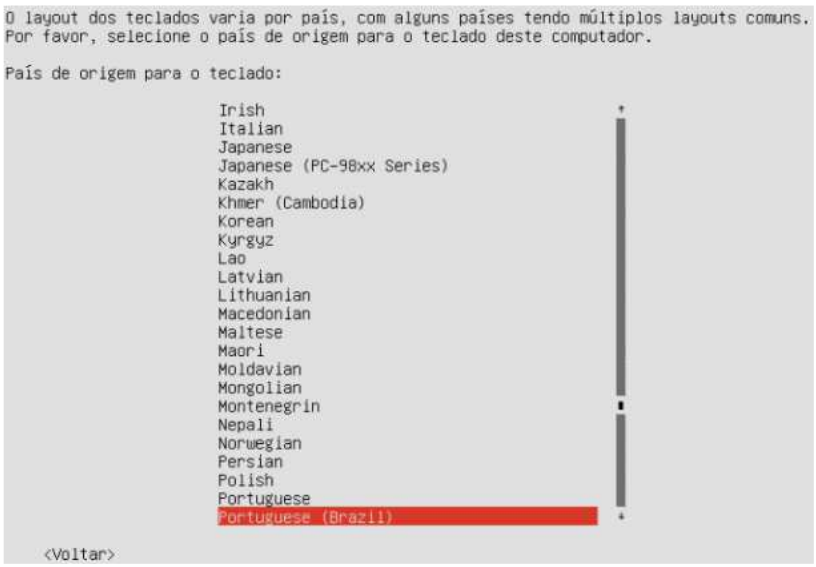


Fonte: Autor

A.2 - INSTALAÇÃO DO UBUNTU

Primeiramente, foi escolhido o idioma Português do Brasil (Figura A.7).

Figura A.7: Escolha do idioma para configuração do teclado

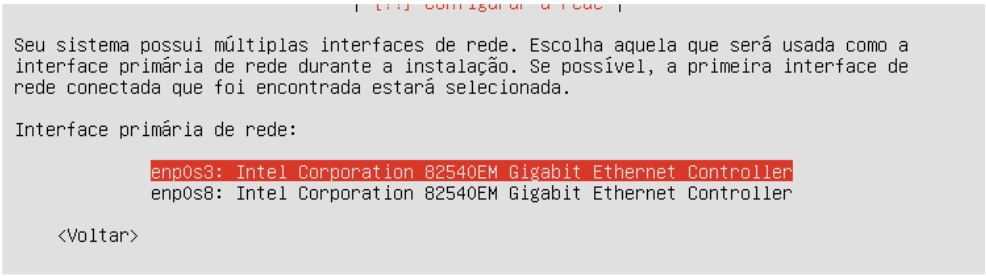


Fonte: Autor

Após o carregamento da primeira configuração, definiu-se quais foram os módulos das placas de redes enp0s3 e enp0s8 que, nesse caso NAT e Bridge, respectivamente (Figura A.8).

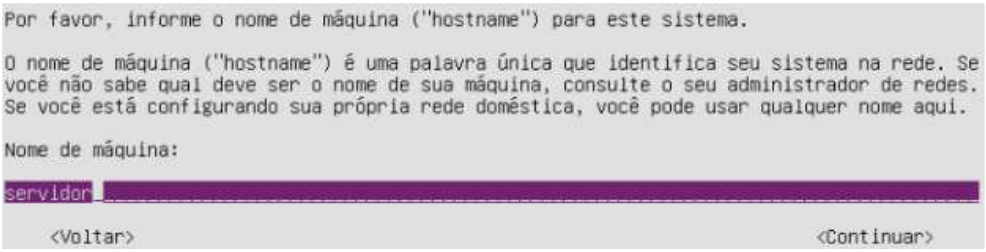
Em seguida foi escolhido o nome do *hostname* do servidor, que nesse caso foi '**servidor**' (Figura A.9).

Figura A.8: Reafirmação das configurações das interfaces da máquina



Fonte: Autor

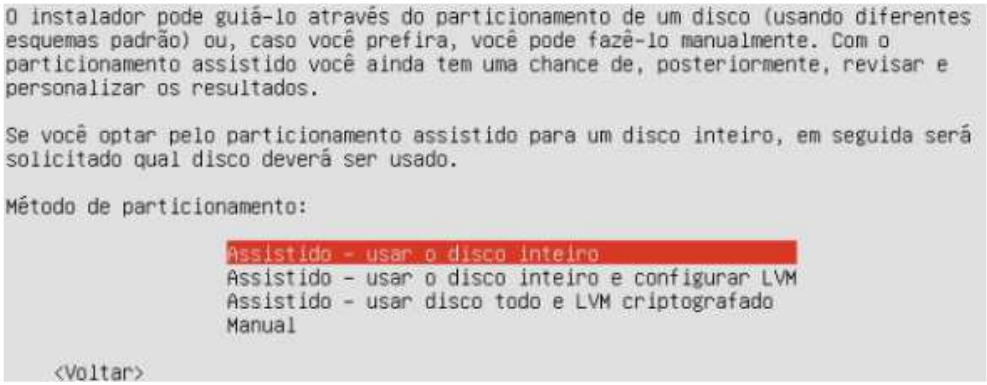
Figura A.9: Escolha do *hostname* da máquina



Fonte: Autor

Na Figura A.10 teve-se a opção de criptografia do disco, caso se deseje. Na configuração do relógio foi posicionado o fuso da cidade de São Paulo. Não foi utilizado LVM, mas o disco inteiro.

Figura A.10: Escolha de uso de disco inteiro



Fonte: Autor

Selecionando o disco primário SCS13 com mudanças no disco, após instalação a configuração do Proxy HTTP ficou em branco.

Na parte de instalações automáticas, marcou-se "**Instalar atualizações de segurança automática**". Depois o programa de instalação perguntou quais programas se desejava instalar, onde foram marcados DNS server, LAMP server, Sampa file server, Standard system utilities e OpenSSH (Figura A.11).

Figura A.11: Seleção dos programas que serão instalados

Fonte: Autor

A.3 – CONFIGURAÇÃO DO UBUNTU

Para começar a configuração foram utilizados os pacotes do Ubuntu com os comandos '**sudo apt-get update**' e '**sudo apt-get upgrade**'.

Na sequência foi instalado o programa '**Nano**' para editar os textos que foram utilizados no processo de configuração com o comando '**sudo apt-get install nano**'.

Na configuração do DHCP foram feitas alterações nas interfaces da rede que podem ser observadas com o comando '**sudo ifconfig -a | more**' (Figura A.12).

Na Figura A.12 podem-se observar as três interfaces enp0s3, enp0s8 e lo, sendo:

a) enp0s3: placa de rede em NAT

b) enp0s8: placa de rede Bridge

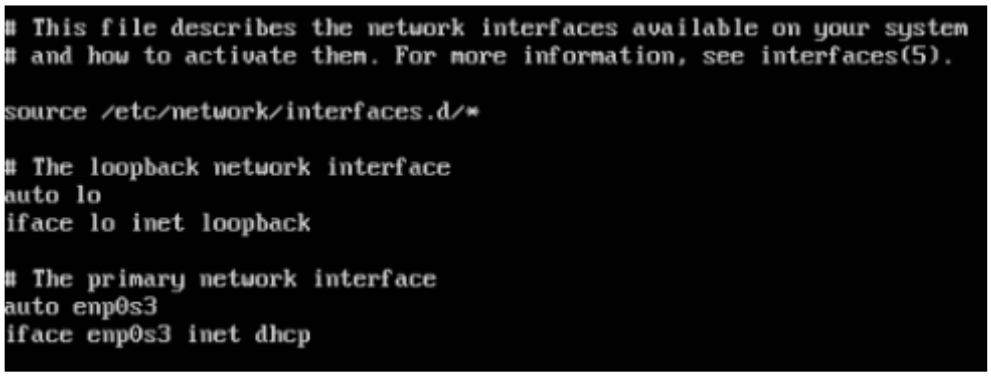
c) lo: placa de rede loopback, (comunica-se) com ela mesma.

Editou-se a rede enp0s8 com o comando '**sudo nano /etc/network/interfaces**' (Figura A.13).

Figura A.12: Visualização das interfaces de Rede

Fonte: Autor

Figura A.13: Área interface enp0s8 para recebimento de IP



Fonte: Autor

Na Figura A.14 mostra a configuração feita para o address IP, o *netmask*, a *network* e o *broadcast*.

Feitas as alterações, foi reiniciada a rede com o comando '**sudo invoke-rc.d networkig restart**'.

Depois editou-se o arquivo '**/etc/sysctl.conf**' que permitiu o encaminhamento ao ipv4. Aberto o arquivo, o comentário '#' na linha "**#inet.ipv4.ip_forward=1**" foi removido

(Figura A.15). Em seguida foi usado o comando '**sudo systcl -w net.ipv4.ip_forward=1**' para aplicar as alterações.

Figura A.14: Configuração do *address, netmask, network* e *broadcast*

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet static
address 192.168.0.100
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
```

Fonte: Autor

Figura A.15: Retirada do comentário da linha '**net.ipv4.ip_forward=1**'

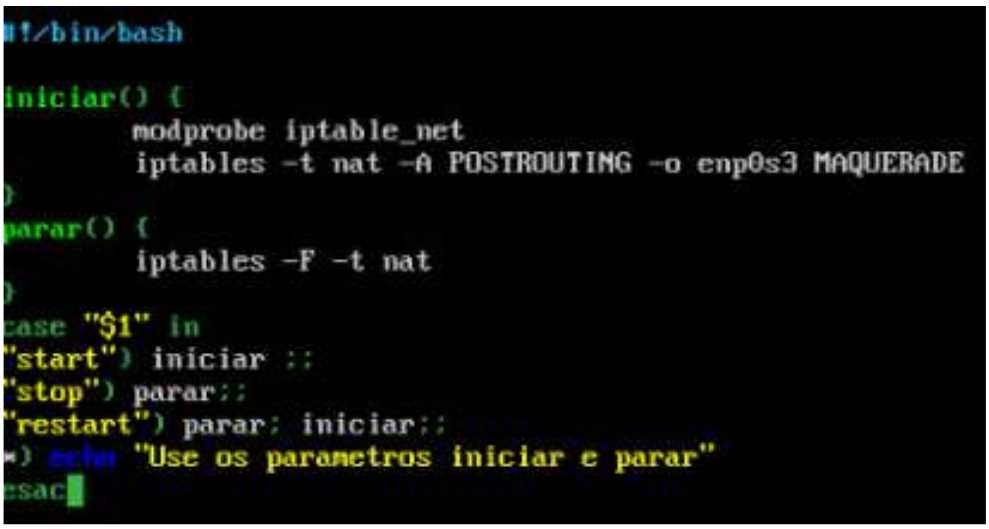
```
#########################################################################3
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1

# Uncomment the next line to enable packet forwarding for IPv6
#  Enabling this option disables Stateless Address Autoconfiguration
#  based on Router Advertisements for this host
```

Fonte: Autor

Foi criado um script para gerenciar o serviço de internet com o comando '**sudo nano /etc/init.d/internet**' (Figura A.16).

Foi usado o comando '**sudo chmod 777 /etc/init.d/internet**' para permissão para execução ao arquivo e, em seguida, foi reiniciado o serviço por '**sudo invoke-rc.d internet restart**'.

Figura A.16: Script de gerenciamento de serviço de internet

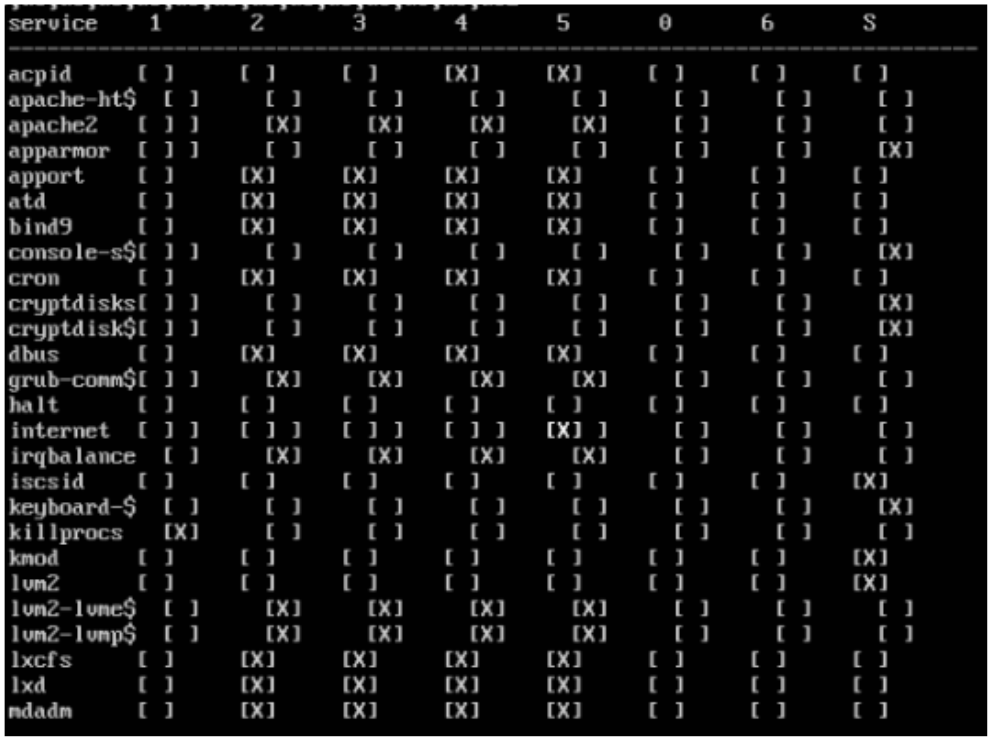


```
#!/bin/bash

iniciar() {
        modprobe iptable_net
        iptables -t nat -A POSTROUTING -o enp0s3 MAQUERADE
}
parar() {
        iptables -F -t nat
}
case "$1" in
"start") iniciar ;;
"stop") parar;;
"restart") parar; iniciar;;
*) echo "Use os parametros iniciar e parar"
esac
```

Fonte: Autor

Para que o serviço se reinicie junto com o servidor editou-se o arquivo '**sysv-rc-conf**' que é instalado por '**sudo apt-get install sysv-rc-conf**' e executado por '**sudo sysv-rc-conf**'. O '**sysv-rc-conf**' é o programa que permite alterar o *runlevel* de serviços e *scripts*. No Linux o *runlevel* são os níveis do processo de *boot* percorre. Para o *script* desenvolvido utilizou-se os níveis do *runlevel* de 2 a 5 (Figura A.17).

Figura A.17: Seleção dos serviços que ficaram ativos

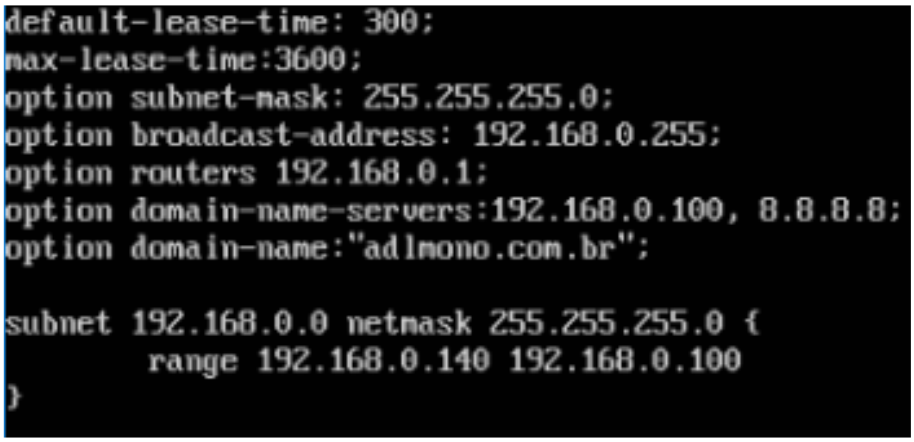


service	1	2	3	4	5	0	6	S
acpid	[ ]	[ ]	[ ]	[X]	[X]	[ ]	[ ]	[ ]
apache-ht$	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
apache2	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
apparmor	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
apport	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
atd	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
bind9	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
console-s$	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
cron	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
cryptdisks	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
cryptdisk$	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
dbus	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
grub-comm$	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
halt	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
internet	[ ]	[ ]	[ ]	[ ]	[X]	[ ]	[ ]	[ ]
irqbalance	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
iscsid	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
keyboard-$	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
killprocs	[X]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]
kmod	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
lvm2	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[ ]	[X]
lvm2-lvme$	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
lvm2-lvmp$	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
lxcfs	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
lxd	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]
mdadm	[ ]	[X]	[X]	[X]	[X]	[ ]	[ ]	[ ]

Fonte: Autor

Após essas alterações passou-se à configuração do DHCP, instalando-o com '**sudo apt-get install isc-dhcp-server**' e depois editando o arquivo '**/etc/dhcp/dhcpd.conf**' com o comando '**sudo nano /etc/dhcp/dhcpd.conf**' (Figura A.18).

Figura A.18: Configuração do DHCP

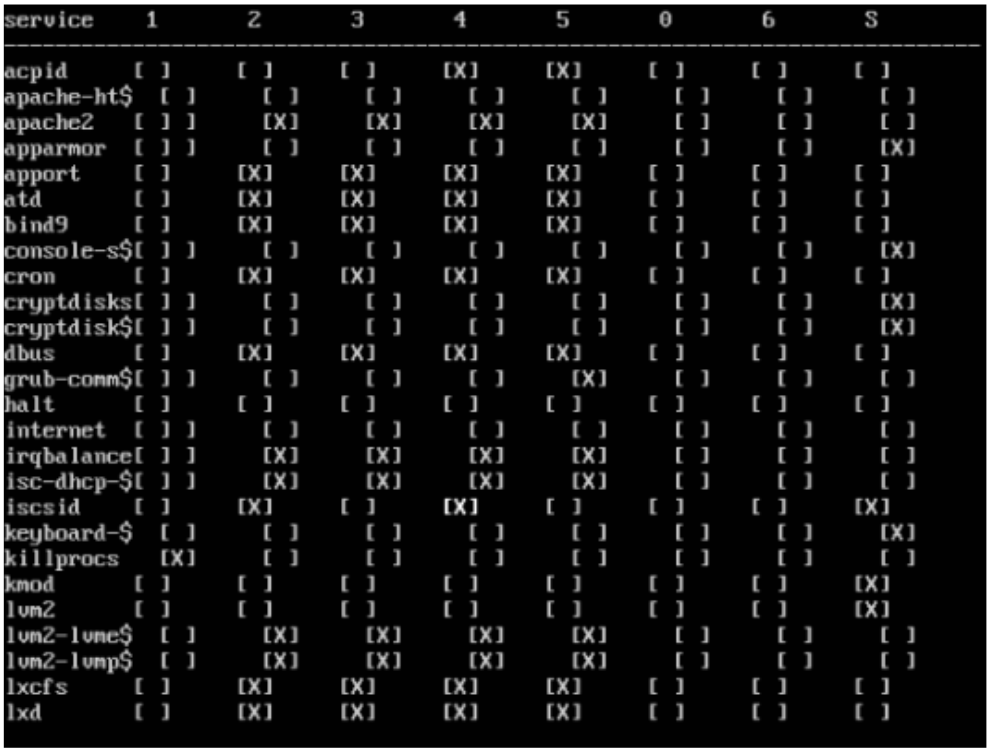


Fonte: Autor

Depois inseriu-se o enp0s8 no *script* do arquivo, trocando a linha '**INTERFCE=””**' por '**INTERFACE=”enp0s8”**'.

Abrindo o '**sysv-rc.conf**' com **'sudo sysv-rc.conf'**, procurou-se pela linha do '**isc-dhcp-server**' para se marcar as colunas 2 e 4 (Figura A.19).

Figura A.19: Marcação dos serviços do DHCP



Fonte: Autor

Na sequência, reiniciou-se o serviço por '**sudo service isc-dhcp-server restart**'.

Para instalar o servidor FTP usou-se o comando '**sudo apt-get install vsftpd**'. Após a instalação, abriu-se o arquivo de configuração do vsftpd com o comando '**sudo nano /etc/vsftpd.conf**'. No arquivo, em algumas linhas removeu-se os comentários (Figura A.20) e outros foram adicionados (Figura A.21).

Figura A.20: Remoção dos comentários '#'

```
write_enable = YES
local_umask = 022
chroot_local_user = YES
```

Fonte: https://pedroteixeira.io/criando-um-servidor-virtual-virtual-host-com-apache-2-no-linux/

Figura A.21: Linhas adicionadas

```
allow_writeable_chroot = YES
pasv_enable = YES
pasv_min_port = 40000
pasv_max_port = 40100
```

Fonte: https://pedroteixeira.io/criando-um-servidor-virtual-virtual-host-com-apache-2-no-linux/

Foram adicionadas as configurações mostradas na Figura A.22

Figura A.22: Visão de toda a configuração para o FTP

```
# This string is the name of the PAM service vsftpd will use.
pam_service_name=vsftpd
#
# This option specifies the location of the RSA certificate to use for SSL
# encrypted connections.
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=NO

#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES
allow_writeable_chroot = YES
pasv_enable = YES
pasv_min_port = 40000
pasv_max_port = 40100_

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Fonte: Autor

Configurando e instalando o '**bind9/DNS**' para obtenção de serviço de interpretação de endereços IPs legíveis e memoráveis. A instalação foi feita com '**sudo apt-get install bind9**' e para fazer a configuração foi necessário encontrar os arquivos de configuração do DNS (que se encontram na pasta '**/etc/bind**', a qual contém o arquivo '**named.conf.default-zones**'), utilizando o comando '**sudo nano /etc/bind/named.conf. default-zones**' (Figura A.23).

Figura A.23: Configuração do DNS pelo '**bind9**'

```
// prime the server with knowledge of the root servers
zone "." {
        type hint;
        file "/etc/bind/db.root";
};

// be authoritative for the localhost forward and reverse zones, and for
// broadcast zones as per RFC 1912

zone "localhost" {
        type master;
        file "/etc/bind/db.local";
};

zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};

zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};
```

Fonte: Autor

Foram criados os arquivos '**db.adlmono**' e '**db.192**' como copias do arquivo '**db.empty**' (Figura A.24).

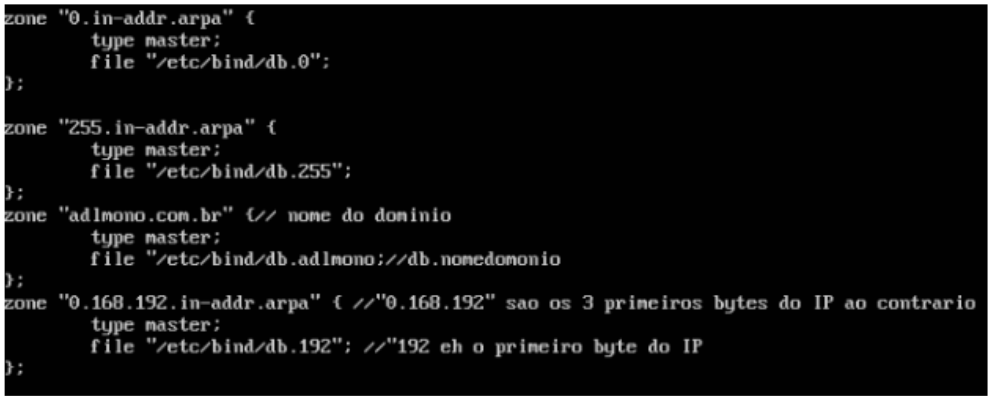
Foram editados os arquivos '**db.adlmono**' e '**db.192**' (Figuras 25 e 26).

A Figura A.27 mostra a configuração do endereço IP do servidor.

Para que o script desenvolvido reinicie junto com Linux criou-se o *script* com o comando '**sudo nano/etc/init.d/script_dns.sh**' e escrevendo '**echo nameserver 192.168.0.100' > /etc/resolv.conf**'. Depois foram dadas as permissões de execução com os comandos da Figura A.28.

O serviço foi reiniciado com o comando '**sudo service bind9 restart**'.

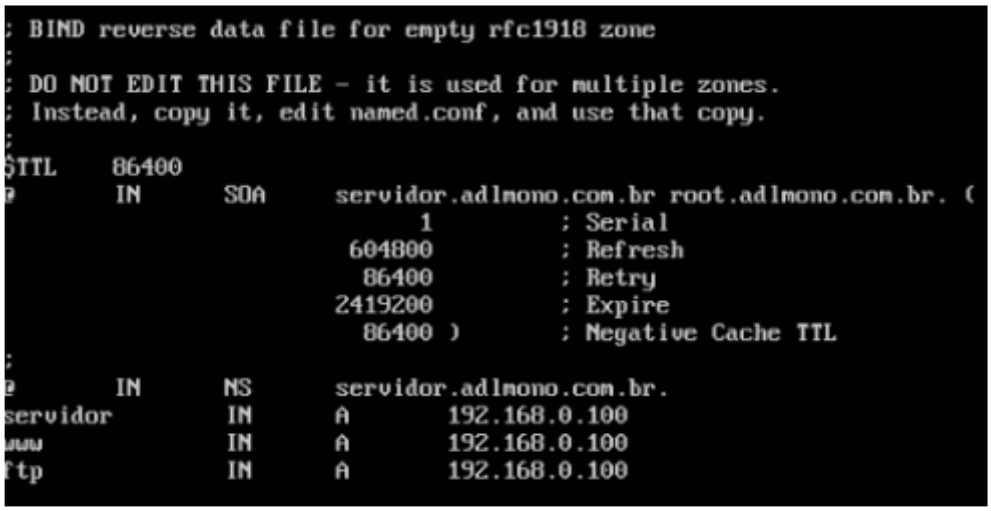
Figura A.24: Criação dos arquivos '**db.adlmono**' e '**db.192**'



```
zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};

zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};
zone "adlmono.com.br" {// nome do dominio
        type master;
        file "/etc/bind/db.adlmono;//db.nomedomonio
};
zone "0.168.192.in-addr.arpa" { //"0.168.192" sao os 3 primeiros bytes do IP ao contrario
        type master;
        file "/etc/bind/db.192"; //"192 eh o primeiro byte do IP
};
```

Fonte: Autor

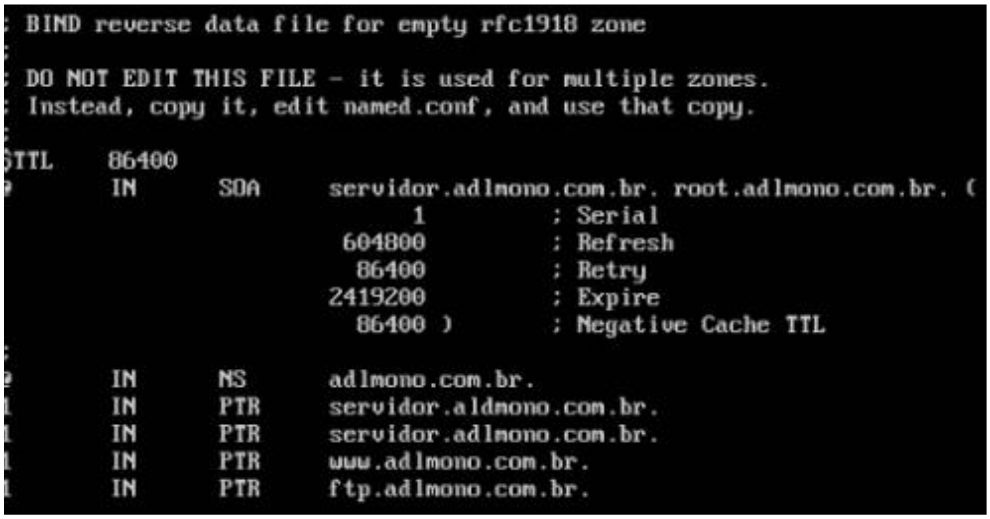
Figura A.25: Edição do '**db.adlmono**'



```
; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL    86400
@       IN      SOA     servidor.adlmono.com.br root.adlmono.com.br. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      servidor.adlmono.com.br.
servidor        IN      A       192.168.0.100
www             IN      A       192.168.0.100
ftp             IN      A       192.168.0.100
```

Fonte: Autor

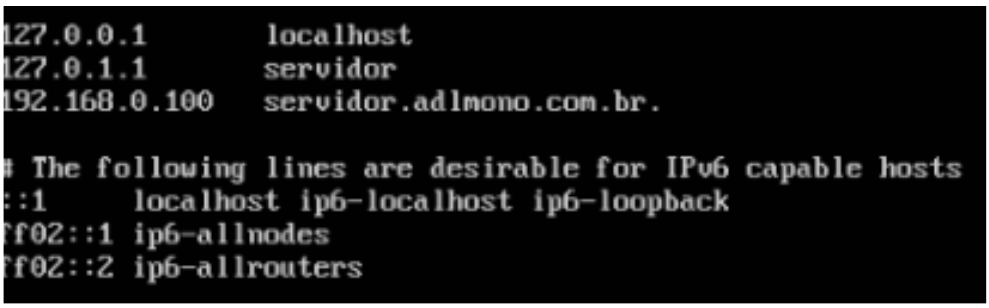
Figura A.26: Edição do '**db.192**'



```
; BIND reverse data file for empty rfc1918 zone
;
; DO NOT EDIT THIS FILE - it is used for multiple zones.
; Instead, copy it, edit named.conf, and use that copy.
;
$TTL    86400
@       IN      SOA     servidor.adlmono.com.br. root.adlmono.com.br. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                          86400 )       ; Negative Cache TTL
;
@       IN      NS      adlmono.com.br.
1       IN      PTR     servidor.aldmono.com.br.
1       IN      PTR     servidor.adlmono.com.br.
1       IN      PTR     www.adlmono.com.br.
1       IN      PTR     ftp.adlmono.com.br.
```

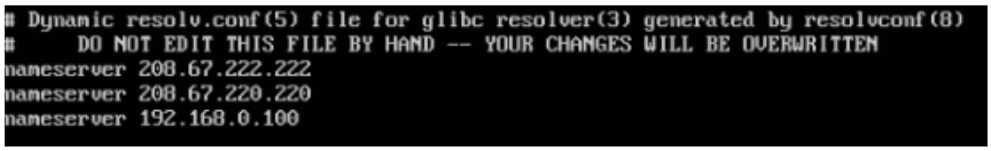
Fonte: Autor

Figura A.27: Configuração do IP do servidor



Fonte: Autor

Figura A.28: Configuração para que o servidor reinicie junto com Linux



Fonte: Autor

APÊNDICE B – CÓDIGOS DE PÁGINAS (HTML) E COMUNICAÇÃO SERVIDOR-CLIENTE E SERVIDOR-SISTEMA EMBARCADO (PHP)

Neste apêndice são disponibilizados dois tipos de código: as páginas da máquina cliente feitos na linguagem HTML e os de comunicação entre o servidor Apache entre o servidor e o sistema embarcado nos quatro experimentos implementados.

B.1. – CÓDIGO DA PÁGINA PRINCIPAL EM HTML

```
<!--tag indicacao para o navegador da versão recente de uso do HMTL-->
<!DOCTYPE html>
<!--tag de inicio de hierarquia do HTML e determinacao na busca da pesquisa do navegador no idoma em portugues do brasil-->
<html lang="pt-br">
<!--tag de infomacoes do documento, titulo (meta), codigo e conexao com CSS(edicao)-->
<head>
<!--tag de configuracao no uso de caracteres pela pagina-->
<meta charset= "UTF-8"/>
<!--titulo do documento-->
<title>Monografia</title>
 <!--conexao com o editor do documento-->
<link rel="stylesheet" href="_css/estilo.css"/> </head>
<body>
<!--tag de divisao com a identificao da interface dando um delimitacao ao site-->
<div id="interface">
<!--tag de marcao de cabecalho do site-->
<header id="cabecalho">
<!--tag de agrupacao das tags de macacao de hierarquia-->
<hgroup>
<!--tag de marcacao de titulo da primeira importancia-->
<h1>UNIVERSIDADE FEDERAL DO MARANHÃO</h1>
<h1>CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA</h1>
<h1>DEPARTAMENTO DE ENGENHARIA ELÉTRICA</h1>
<h1>CURSO  DE ENGENHARIA ELÉTRICA</h1>
<h1>ALUNO: ADENILSON LIMA DINIZ</h1>
<h1>MONOGRAFIA</h1>
<!--tag de marcacao de titulos da segunda importancia-->
<h2>Controle e Monitoramento de Sistemas via Web</h2>
<!--tag de navegao de identificacao da imagem no site-->
<nav id="figura1">
<!--tag de edicao imagens-->
<figure class="logo_ufma">
 <!--tag de definicao da imagen-->
<img src="_imagens/logo_ufma.jpg"> </figure> </nav>
<nav id="figura2">
<figure class="logo_LRC.jpg">
<img src="_imagens/logo_LRC.jpg" width="170">  </figure> </nav>
<!--tag de navegacao com identificacao do menu-->
<nav id="menu">
<!--tag de marcacao de terceira importancia-->
<h3> Menu de Experiências</h3>
<!--tag de listagem do menu de forma nao ordenada do tipo disco -->
<ul type = "disc">
```

```
<!--tag <li> determina cada item da lista e a tag <a> para ancora para criacao de um link entre as paginas sendo a pricipal a do index.html que a interface principal -->
<li><a href="index.html">Home</a></li>
<li><a href="experiencia1.html">Experiência 1</a></li>
<li><a href="experiencia2.html">Experiência 2</a></li>
<li><a href="experiencia3.html">Experiência 3</a> </li>
<li><a href="experiencia4.html">Experiência 4</a></li></ul></nav>
<!--tag de paragrafo-->
                    <p> <u>Resumo:</u> Este trabalho de mono<wbr>gráfia é uma implemen<wbr/>tação de uma estrutura cliente-servidor para controle e moni<wbr>toração de dispos<wbr>itivos inteligentes (acionados pela plataforma Arduino ligada ao ser<wbr>vidor) pela maquina cliente. Os dispositivos a serem controlados e monitorados serão um con<wbr>junto de LEDs, um motor DC e um Robô Móvel. Com essa estrutura implan<wbr>tada, espera-se demons<wbr>trar o acesso ao controle e monitoração por meio da pagina web, exe<wbr/>cutada remota<wbr/>mente na maquina cliente. Escolhedo um dos experimentos do menu, testaremos a aplicabi<wbr>lidade do sistema e sua efi<wbr>ciên<wbr>cia.</p></hgroup> </header> </div> </body></html>
```

B.1.1 – CÓDIGO DO ESTILO DAS INTERFACES (CSS)

```
/*Programa de estilo das paginas htlms conecentadas por meio de links */
@charset="UTF-8";
/*FORMATAÇÃO DA LIMITAÇÃO DO CORPO. QUADRO DO CORPO  "INTERFACE"*/
/*o comando dv server para limitar o corpo dentro de um contener que sera*/
/*uma posicao relativa fora desse contener sera absoluto*/
div#interface{
width: 900px;  /*limite da lagura da inerface*/
background-color:#F5F5DC; /*cor de fundo*/
margin: 40px auto 0px auto;/*na ordem: encima, direita, embaixo, esquerda no ajuste da margem*/
box-shadow: 0px 0px 10px rgba(0,0,0,0.5);/* sobra da margem: deslocamento latera, deslocamento vertical, tamanho da sombre e a cor da sombra*/
padding: 10px 10px 10px 10px;}/*cofiguracao da margem interna. distacia das letras da margem da interface na ordem de cima, direita,embaixo e esquerda*/
/*formatacao dos textos em h1 , h2 e h3 do resumo (p)*/
header#cabecalho h1 {
font-family: times, Times new Roman;
font-size: 16pt;
text-align:center;
padding:0px;}
header#cabecalho h2{
font-family: times, Times new Roman;
font-size:15pt;
text-align:center;
padding:0px;}
header#cabecalho h3{
font-family: times, Times new Roman;
font-size:15pt;
text-align:leaf;
padding:0px;}
header#cabecalho li {
font-family: times, Times new Roman;
font-size:13pt;
text-align:center;
padding:3px;}
header#cabecalho p {
text-align: justify;
text-indent:50px;
font-size: 15pt;
font-family: times, Times new Roman;}
```

```
body {
/*FORMATAÇÃO DO CORPO E DO RESUMO*/
background-color: #dddddd;
color: rgba(0,0,0,1);}

/*FORMATAÇÃO DO MENU*/
nav#menu{
display: block;}
nav#menu ul {
list-style: none;
text-transform: uppercase;
position: relative}
nav#menu li{
display: inline-block;
background-color: #dddddd;
margin: 20px;
transition: background-color 0.5s; }
/*EFEITOS VISUAL NO MENU*/
nav#menu li:hover{
background-color: #606060; }
nav#menu h3{
  display: none;}
nav#menu a {
  color: #000000;
  text-decoration: none;}
nav#menu a:hover{
  color:#ffffff;}
/*Formatação de imagens*/
nav#figura1 {
display: block; /*formato bloco que possibilita as imegens se movimentarem*/
position: absolute; /* possiblita se movimentarem fora da area delimitada, diferente dos texto que realiva*/
top: 50px; /* distancia do cima */
left: 200px;} /*distancia do lado esquerdo*/
nav#figura2{
display: block;
position: absolute;
top: 50px;
left:910px;}
/*Formatação do controle do experimento 1 frequencia e tempo de operação*/
nav#controlef{
  font-family: times, Times new Roman;
  font-size:15pt;
  text-align:center;
  padding:0px;}
nav#controlet{
  font-family: times, Times new Roman;
  font-size:15pt;
  text-align:center;
  padding:0px;}
nav#controlec{
  font-family: times,Times new Roman;
  font-size: 15px;
  text-align: center;
  padding: 0px; }
```

B.2. – CÓDIGOS DO EXPERIMENTO 1

```
<?php
//RECEBIMENTO DOS REQUERIMENTOS
$freq1=$_REQUEST['led1'];
$freq2=$_REQUEST['led2'];
$freq3=$_REQUEST['led3'];
$_REQUEST['nota'];
// ABERTURA E REGISTRO DA NA PORTA USB
$port=fopen("/dev/ttyACM0", "w");
//OPERACAO PARA 1 KHZ/LED1/BAIXA LUMINOSIDADE
if ($freq1=="ALTA FREQ") {
  $acao='1';}
if($freq1=="MEDIA FREQ"){
  $acao='2';}
if($freq1=="BAIXA FREQ"){
  $acao='3';}
//OPERACAO PARA 500 HZ/LED2
if ($freq2=="ALTA FREQ") {
  $acao='4';}
if ($freq2=="MEDIA FREQ") {
  $acao='5';}
if ($freq2=="BAIXA FREQ") {
  $acao='6';}
//OPERACAO PARA 200 HZ/LED3
if ($freq3=="ALTA FREQ") {
  $acao='7';}
if ($freq3=="MEDIA FREQ") {
  $acao='8';}
if ($freq3=="BAIXA FREQ") {
  $acao='9';}
if ($_REQUEST['nota']=="CANCELAR") {
  $acao="0";}
   sleep(0.5); // TEMPO DE DELAY
    fwrite($port,$acao); // GRAVACAO NA USB
    fclose($port); // FECHAMENTO DA PORTA USB
  ?>
```

B.2.1 – CÓDIGO DA PÁGINA DO EXPERIMENTO 1 EM HTML

```
<html lang="pt-br">
<head>
<meta charset= "UTF-8"/>
<title>Monografia</title>
<link rel="stylesheet" href="_css/estilo.css"/></head>
<body>
<div id="interface">
<header id="cabecalho">
<hgroup>
<h1>UNIVERSIDADE FEDERAL DO MARANHÃO</h1>
<h1>CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA</h1>
<h1>DEPARTAMENTO DE ENGENHARIA ELÉTRICA</h1>
<h1>CURSO  DE ENGENHARIA ELÉTRICA</h1>
<dir align ="right">
<nav id="menu">
 <ul type = "disc">
<li><a href="index.html">Home</a></li></ul></nav></dir>
<nav id="figura1">
```

```
<figure class="logo_ufma">
  <img src="_imagens/logo_ufma.jpg"></figure> </nav>
   <nav id="figura2">
  <figure class="logo_LRC.jpg">
  <img src="_imagens/logo_LRC.jpg" width="170"></figure></nav>
  <!--EDICAO DOS BOTOES DE COMANDO-->
  <style type="text/css">
 .mono{
 width: 100px;
  height: 40px;
 background: #666;
 color: #FFF;} </style>
<h1>EXPERIÊNCIA 1</h1>
<h2>LED OPERANDO EM BAIXA LUMINOSIDADE</h2>
<div align="center">
 <nav id="controlef">
 <!--ENDERECAMENTO E METODO DE ARMAZENAMENTO-->
<form  action="arduinoa.php" method="get" >
<!--TIPO DE ENTRADA COM IDENTIFICACAO ARMAZENAMENTO DE VALOR-->
 <input type="submit"  class="mono" name="led1" id="botao"      value="BAIXA FREQ"> 
 <input type="submit"  class="mono" name="led1" id="botao"      value="MEDIA FREQ"> 
 <input type="submit"  class="mono" name="led1" id="botao"      value="ALTA FREQ"> 
 <form action="arduinoa.php" method="get">
 <input type="submit" class="mono" name="nota" id="botao"  value="CANCELAR"> 
 </form> </form> </nav>  </div>
 <style type="text/css">
  .mono{
  width: 100px;
  height: 40px;
  background: #666;
  color: #FFF; }</style>
 <h2>LED OPERANDO EM MEDIA LUMINOSIDADE</h2>
 <div align="center">
 <nav id="controlef">
 <form  action="arduinoa.php" method="get" >
  <input type="submit" class="mono" name="led2" id="botao"       value="BAIXA FREQ"> 
 <input type="submit" class="mono" name="led2" id="botao"       value="MEDIA FREQ"> 
 <input type="submit" class="mono" name="led2" id="botao"       value="ALTA FREQ"> 
 <form action="arduinoa.php" method="get">
<input type="submit" class="mono" name="nota" id="botao0"  value="CANCELAR"> 
 </form> </form> </nav>  </div>
<h2> LED OPERANDO EM ALTA LUMINOSIDADE</h2>
<div align="center">
<nav id="controlef">
 <form  action="arduinoa.php" method="get" >
 <input type="submit" class="mono" name="led3" id="botao"     value="BAIXA FREQ"> 
 <input type="submit" class="mono" name="led3" id="botao"     value="MEDIA FREQ"> 
 <input type="submit" class="mono" name="led3" id="botao"     value="ALTA FREQ"> 
 <form action="arduinoa.php" method="get">
 <input type="submit" class="mono" name="nota" id="botao0"  value="CANCELAR"> 
 </form> </form> </nav> </div></hgroup></header></div></body></html>
```

B.3. – CÓDIGOS DO EXPERIMENTO 2

```
?php
$vel1=$_REQUEST['vel1'];
$vel2=$_REQUEST['vel2'];
$vel3=$_REQUEST['vel3'];
```

```
$vel4=$_REQUEST['vel4'];
$_REQUEST['nota'];
$port=fopen("/dev/ttyACM0", "w");
//SERIE/CRESC
if ($vel1=="VEL ALTA") {
  $acao='1';}
if ($vel1=="VEL BAIXA") {
  $acao='2';}
//SERIE/DECRESC
if ($vel2=="VEL ALTA") {
  $acao='3';}
if ($vel2=="VEL BAIXA") {
  $acao='4';}
//PARALELA/CRESC
if ($vel3=="VEL ALTA") {
  $acao='5';}
if ($vel3=="VEL BAIXA") {
  $acao='6';}
//PARALELA/DECRESC
if ($vel4=="VEL ALTA") {
  $acao='7';}
if ($vel4=="VEL BAIXA") {
  $acao='8';}
if ($_REQUEST['nota']=="CANCELAR") {
  $acao="0";}
   sleep(0.05);
    fwrite($port,$acao);
    fclose($port);
  ?>
```

B.3.1 – CÓDIGO DA PÁGINA DO EXPERIMENTO 2 EM HTML

```
<html lang="pt-br">
<head>
<meta charset= "UTF-8"/>
<title>Monografia</title>
<link rel="stylesheet" href="_css/estilo.css"/></head>
<body>
<div id="interface">
            <header id="cabecalho">
            <hgroup>
<h1>UNIVERSIDADE FEDERAL DO MARANHÃO</h1>
<h1>CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA</h1>
<h1>DEPARTAMENTO DE ENGENHARIA ELÉTRICA</h1>
<h1>CURSO  DE ENGENHARIA ELÉTRICA</h1>
<h1>EXPERIÊNCIA 2</h1>
<dir align ="right">
<nav id="menu">
<ul type = "disc">
<li><a href="index.html">Home</a></li></ul></nav></dir>
  <nav id="figura1">
  <figure class="logo_ufma">
  <img src="_imagens/logo_ufma.jpg"></figure></nav>
 <nav id="figura2">
 <figure class="logo_LRC.jpg">
<img src="_imagens/logo_LRC.jpg" width="170"></figure></nav>
 <style type="text/css">
 .mono{
 width: 100px;
```

```
 height: 40px;
 background: #666;
 color: #FFF;}</style>
<h2>LEDS EM SERIE NA ORDEM CRESCENTE</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinob.php" method="get" >
<input type="submit" class="mono" name="vel1" id="botao"   value="VEL BAIXA">  
<input type="submit" class="mono" name="vel1" id="botao"   value="VEL ALTA">  
<form action="arduinob.php" method="get">
<input type="submit" class="mono"name="nota" id="botao"   value="CANCELAR">
</form> </form> </nav></div>
<h2>LEDS EM SERIE NA ORDEM DECRESCENTE</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinob.php" method="get" >
<input type="submit" class="mono" name="vel2" id="botao"   value="VEL BAIXA">  
<input type="submit" class="mono" name="vel2" id="botao"   value="VEL ALTA">  
<form action="arduinob.php" method="get">
<input type="submit" class="mono" name="nota" id="botao"   value="CANCELAR">
</form></form> </nav></div>
<h2>LEDS EM PARALELO NA ORDEM CRESCENTE</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinob.php" method="get" >
<input type="submit" class="mono" name="vel3" id="botao"      value="VEL BAIXA">  
<input type="submit" class="mono" name="vel3" id="botao"      value="VEL ALTA">  
<form action="arduinob.php" method="get">
<input type="submit" class="mono" name="nota" id="botao"      value="CANCELAR">
</form></form> </nav></div>
 <h2>LEDS EM PARALELO NA ORDEM DECRESCENTE</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinob.php" method="get" >
<input type="submit" class="mono" name="vel4" id="botao"      value="VEL BAIXA">  
<input type="submit" class="mono" name="vel4" id="botao"      value="VEL ALTA">  
<form action="arduinob.php" method="get">
<input type="submit" class="mono" name="nota" id="botao"      value="CANCELAR">
 </form> </form> </nav> </div></hgroup></header></div></body></html>
```

B.4. – CÓDIGOS DO EXPERIMENTO 3

```
<?php
$freq1=$_REQUEST['MOTOR1'];
$freq2=$_REQUEST['MOTOR2'];
$_REQUEST['nota'];
$port=fopen("/dev/ttyACM2", "w");
//horario
if ($freq1=="100 %") {
  $acao='1';}
if($freq1=="50 %"){
  $acao='2';}
if($freq1=="20 %"){
  $acao='3';}
//anti-horario
if ($freq2=="100 %") {
  $acao='5';}
if ($freq2=="50 %") {
```

```
  $acao='6';}
if ($freq2=="20 %") {
  $acao='7';}
if ($_REQUEST['nota']=="CANCELAR") {
  $acao="0";}
   sleep(0.5);
    fwrite($port,$acao);
    fclose($port);
  ?>
```

B.4.1 – CÓDIGO DA PÁGINA DO EXPERIMENTO 3 EM HTML

```
<html lang="pt-br">
<head>
<meta charset= "UTF-8"/>
<title>Monografia</title>
<link rel="stylesheet" href="_css/estilo.css"/></head>
<body>
<div id="interface">
<header id="cabecalho">
<hgroup>
<h1>UNIVERSIDADE FEDERAL DO MARANHÃO</h1>
<h1>CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA</h1>
<h1>DEPARTAMENTO DE ENGENHARIA ELÉTRICA</h1>
<h1>CURSO  DE ENGENHARIA ELÉTRICA</h1>
<dir align ="right">
<nav id="menu">
<ul type = "disc">
<li><a href="index.html">Home</a></li></ul></nav></dir>
<nav id="figura1">
<figure class="logo_ufma">
<img src="_imagens/logo_ufma.jpg"> </figure></nav>
<nav id="figura2">
<figure class="logo_LRC.jpg">
<img src="_imagens/logo_LRC.jpg" width="170"></figure></nav>
 <style type="text/css">
.mono{
width: 100px;
height: 40px;
background: #666;
color: #FFF; } </style>
<h1>EXPERIÊNCIA 3</h1>
<h2>MOTOR DC OPERANDO EM SENTIDO HORARIO</h2>
<div align="center">
<nav id="controlef">
 <form  action="arduinoc.php" method="get" >
<input type="submit"  class="mono" name="MOTOR1" id="botao"     value="20 %"> 
<input type="submit"  class="mono" name="MOTOR1" id="botao"     value="50 %"> 
<input type="submit"  class="mono" name="MOTOR1" id="botao"     value="100 %"> 
<form action="arduinoc.php" method="get">
 <input type="submit" class="mono" name="nota" id="botao"  value="CANCELAR"> 
 </form> </form> </nav> </div>
<h2>MOTOR DC OPERANDO EM SENTIDO ANTI-HORARIO</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinoc.php" method="get" >
<input type="submit" class="mono" name="MOTOR2" id="botao"     value="20 %"> 
<input type="submit" class="mono" name="MOTOR2" id="botao"     value="50 %"> 
<input type="submit" class="mono" name="MOTOR2" id="botao"     value="100 %"> 
```

```
<form action="arduinoc.php" method="get">
<input type="submit" class="mono" name="nota" id="botao0"  value="CANCELAR"> 
 </form></form></nav> </div></hgroup></header></div></body></html>
```

B.5. – CÓDIGOS DO EXPERIMENTO 4

B.5.1 – CÓDIGO DA PÁGINA DO EXPERIMENTO 4 EM HTML

```
<html lang="pt-br">
<head>
<meta charset= "UTF-8"/>
<title>Monografia</title>
<link rel="stylesheet" href="_css/estilo.css"/></head>
<body>
<div id="interface">
<header id="cabecalho">
<hgroup>
<nav id="figura1">
<figure class="logo_ufma">
<img src="_imagens/logo_ufma.jpg"></figure></nav>
<nav id="figura2">
<figure class="logo_LRC.jpg">
<img src="_imagens/logo_LRC.jpg" width="170"></figure></nav>

  <style type="text/css">
  .mono{
  width: 100px;
  height: 40px;
  background: #666;
  color: #FFF; }</style>
<h1>EXPERIÊNCIA 4</h1>
 <br/> <br/>
<h2>SENSOR ULTRSSÔNICO</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinoe.php" method="get" >
<input type="submit"  class="mono" name="medicao" id="botao"      value="MEDIÇÃO">
 </form> </nav>
<dir align="right">
<nav id="menu">
<ul type = "disc">
<li><a href="index.html">Home</a></li></ul></nav></dir>
<h2>NAVEGAÇÃO</h2>
<div align="center">
<nav id="controlef">
<form  action="arduinod.php" method="get" >
<input type="submit"  class="mono" name="direcao" id="botao"      value="FRENTE"> 
 <br/><br/> </form>
<form  action="arduinod.php" method="get" >
<input type="submit"  class="mono" name="direcao" id="botao"
value="ESQUERDA">     
<input type="submit"  class="mono" name="direcao" id="botao"
value="PARAR">     
<input type="submit"  class="mono" name="direcao" id="botao"      value="DIREITA"> 
<br/><br/><br/>
<input type="submit"  class="mono" name="direcao" id="botao"      value="RÉ"> 
</form> </nav> </div>
  <style type="text/css">
```

```
  .mono{
  width: 100px;
  height: 40px;
  background: #666;
  color: #FFF; }</style>
</hgroup></header></div></body></html>
```

B.5.2.1 – CÓDIGO DO MOVIMENTO DO ROBÔ

```
<?php
$dir=$_REQUEST['direcao'];
$med=$_REQUEST['medicao'];
$port=fopen("/dev/ttyUSB0", "w");
//DIRECAO DA NAVEGACAO
if ($dir=="FRENTE") {
  $acao='1';}
if($dir=="RÉ"){
  $acao='2';}
if($dir=="DIREITA"){
  $acao='3';}
if($dir=='ESQUERDA'){
  $acao='4';}
//PARAR
if($dir=='PARAR'){
  $acao='0';}
 sleep(0.5);
 fwrite($port,$acao);
 fclose($port);
?>
```

B.5.2.2 – CÓDIGO DE MEDIÇÃO DO ROBÔ

```
<?php
$port=fopen("/dev/ttyUSB0", "w+");
echo ("<h2> MEDIÇÃO COM SENSOR ULTRASSÔNICO HC-SR04</h2>");
echo("<p>A distância máxima de medição do sensor é de 50 cm.</p>
<p>          O valor '0' representa a distância fora de alcance do sensor( Distância > 50 cm ) </p>");
if(!$port){
  echo("URGENTE: SEM CONEXÃO COM A PORTA SERIAL");}
echo("<p>Mediçaõ em centimentros é:</p>");
fwrite($port, 5);
sleep(0.05);
echo fgets($port); LEITURA DA PORTA USB
fclose($port);
?>
//CONTROLE DE AUXILIO PARA MEDICAO
<html lang="pt-br">
<head>
<meta charset= "UTF-8"/>
<title>Monografia</title>
<link rel="stylesheet" href="_css/estilo.css"/></head>
<body>
<nav id="menu">
<ul type = "disc">
<li><a href="arduinoe.php">Medir</a></li>
<li><a href="experiencia4.html">retorna navegação</a></li>
<li><a href="index.html">Home</a></li></ul></nav></body></html>
```

APÊNDICE C – CÓDIGOS DOS EXPERIMENTOS EMBARCADOS NO ARDUÍNO DESENVOLVIDOS EM LINGUAGEM C.

Neste apêndice são disponibilizados os códigos que encontram-se embarcados no Arduino, a depender do experimento em execução, escritos em linguagem C.

C.1 – EXPERIMENTO 1

```
//Projeto controle de tempo e frequência de luminosidade de um led
int ledPin = 9; // Variavel do pino
int acao = -5; // Variavel de leitura
float sinVal; // Variavel do sinal
int ledVal; // Variavel auxiliar
void setup(){
  pinMode (ledPin, OUTPUT); //Definido a porta como saida
  Serial.begin(9600); } // Velocidade de comunicação serial
void loop(){
    if(Serial.available()>0){ // Disponiblidade para leitura
    acao=Serial.read();} // Leitura da porta serial
 // BAIXA LUMINOSIDADE
    if(acao=='1'){
    for (int x=0 ; x< 180; x++){ // Variavel auxliar para controle dos angulos
    sinVal = (sin(x*(3.1412/180))); //  Formula do seno com a mudança de graus para radianos
    ledVal=int(sinVal*255); //  Multiplicacao com o valor alto do PWM.
    analogWrite(ledPin,(ledVal/20)); // Saida do sinal para o pino
    delay(1); }} // Tempo de repetição
    if(acao=='2'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal/20));
    delay(2);}}
    if(acao=='3'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin, ledVal/20);
    delay(5);}}
    // MEDIA LUMINOSIDADE
    if(acao=='4'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal/4));
    delay(1);}}
   if(acao=='5'){
   for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal/4));
    delay(2);}}
    if(acao=='6'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
```

```
    analogWrite(ledPin, ledVal/4);
    delay(5);}}
    // ALTA LUMINOSIDADE
    if(acao=='7'){
   for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal));
    delay(1);}}
    if(acao=='8'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal));
    delay(2);}}
    if(acao=='9'){
    for (int x=0 ; x< 180; x++){
    sinVal = (sin(x*(3.1412/180)));
    ledVal=int(sinVal*255);
    analogWrite(ledPin,(ledVal));
    delay(5);}}
    else if(acao=='0'){// Condicao de anulacao do programa
    digitalWrite(ledPin,LOW);}}}
```

C.2 – EXPERIMENTO 2

```
//ACIONAMENTO DE LEDS EM SERIE E PARALELO.
// CRESCENTE E DECRESCENTE
//CONTROLE DE VELOCIDADE

// Variaveis dos pinos
int led0=7;
int led1=8;
int led2=9;
int led3=10;
int led4=11;
int led5=12;
int acao=-5;
void setup(){
// Definicao dos pinos de saida
  pinMode(led0,OUTPUT);
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(led3,OUTPUT);
  pinMode(led4,OUTPUT);
  pinMode(led5,OUTPUT);
  Serial.begin(9600); }
 void loop(){
  if(Serial.available()>0){
    acao=Serial.read();}

    //SERIE/CRESCENTE VEL.ALTA
    if(acao=='1'){
//Variacao do Sistema de liga e desliga
    digitalWrite(led5,LOW);
    digitalWrite(led0,HIGH);
    delay(250); tempo de repeticao.
    digitalWrite(led0,LOW);
    digitalWrite(led1,HIGH);
```

```
delay(250);
digitalWrite(led1,LOW);
digitalWrite(led2,HIGH);
delay(250);
digitalWrite(led2,LOW);
digitalWrite(led3,HIGH);
delay(250);
digitalWrite(led3,LOW);
digitalWrite(led4,HIGH);
delay(250);
digitalWrite(led4,LOW);
digitalWrite(led5,HIGH);
delay(250);}
//SERIE/CRESCENTE/VEL BAIXA
if(acao=='2'){
digitalWrite(led5,LOW);
digitalWrite(led0,HIGH);
delay(500);
digitalWrite(led0,LOW);
digitalWrite(led1,HIGH);
delay(500);
digitalWrite(led1,LOW);
digitalWrite(led2,HIGH);
delay(500);
digitalWrite(led2,LOW);
digitalWrite(led3,HIGH);
delay(500);
digitalWrite(led3,LOW);
digitalWrite(led4,HIGH);
delay(500);
digitalWrite(led4,LOW);
digitalWrite(led5,HIGH);
delay(500);}
 //SERIE/DESCRECENTE ALTA
if(acao=='3'){
digitalWrite(led0,LOW);
digitalWrite(led5,HIGH);
delay(250);
digitalWrite(led5,LOW);
digitalWrite(led4,HIGH);
delay(250);
digitalWrite(led4,LOW);
digitalWrite(led3,HIGH);
delay(250);
digitalWrite(led3,LOW);
digitalWrite(led2,HIGH);
delay(250);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
delay(250);
digitalWrite(led1,LOW);
digitalWrite(led0,HIGH);
delay(250);}
//SERIE/DESCRESCENTE VEL BAIXA
if(acao=='4'){
digitalWrite(led5,HIGH);
digitalWrite(led0,LOW);
delay(500);
digitalWrite(led5,LOW);
digitalWrite(led4,HIGH);
```

```
delay(500);
digitalWrite(led4,LOW);
digitalWrite(led3,HIGH);
delay(500);
digitalWrite(led3,LOW);
digitalWrite(led2,HIGH);
delay(500);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
delay(500);
digitalWrite(led1,LOW);
digitalWrite(led0,HIGH);
delay(500);}
 // PARALELA CRESCENTE ALTA
if(acao=='5'){
digitalWrite(led3,LOW);
digitalWrite(led5,LOW);
digitalWrite(led0,HIGH);
digitalWrite(led2,HIGH);
delay(250);
digitalWrite(led0,LOW);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
digitalWrite(led3,HIGH);
delay(250);
digitalWrite(led1,LOW);
digitalWrite(led3,LOW);
digitalWrite(led2,HIGH);
digitalWrite(led4,HIGH);
delay(250);
digitalWrite(led2,LOW);
digitalWrite(led4,LOW);
digitalWrite(led3,HIGH);
digitalWrite(led5,HIGH);
delay(250);}
//PARALELA CRESCENTE BAIXA
if(acao=='6'){
digitalWrite(led3,LOW);
digitalWrite(led5,LOW);
digitalWrite(led0,HIGH);
digitalWrite(led2,HIGH);
delay(500);
digitalWrite(led0,LOW);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
digitalWrite(led3,HIGH);
delay(500);
digitalWrite(led1,LOW);
digitalWrite(led3,LOW);
digitalWrite(led2,HIGH);
digitalWrite(led4,HIGH);
delay(500);
digitalWrite(led2,LOW);
digitalWrite(led4,LOW);
digitalWrite(led3,HIGH);
digitalWrite(led5,HIGH);
delay(500);}
 // PARALELA DECRESCENTE ALTA
if(acao=='7'){
digitalWrite(led0,LOW);
```

```
digitalWrite(led2,LOW);
digitalWrite(led5,HIGH);
digitalWrite(led3,HIGH);
delay(250);
digitalWrite(led5,LOW);
digitalWrite(led3,LOW);
digitalWrite(led4,HIGH);
digitalWrite(led2,HIGH);
delay(250);
digitalWrite(led4,LOW);
digitalWrite(led2,LOW);
digitalWrite(led3,HIGH);
digitalWrite(led1,HIGH);
delay(250);
digitalWrite(led3,LOW);
digitalWrite(led1,LOW);
digitalWrite(led2,HIGH);
digitalWrite(led0,HIGH);
delay(250);}

//PARALELA/DECRESCENTE/BAIXA
if(acao=='8'){
digitalWrite(led0,LOW);
digitalWrite(led2,LOW);
digitalWrite(led5,HIGH);
digitalWrite(led3,HIGH);
delay(500);
digitalWrite(led5,LOW);
digitalWrite(led3,LOW);
digitalWrite(led4,HIGH);
digitalWrite(led2,HIGH);
delay(500);
digitalWrite(led4,LOW);
digitalWrite(led2,LOW);
digitalWrite(led3,HIGH);
digitalWrite(led1,HIGH);
delay(500);
digitalWrite(led3,LOW);
digitalWrite(led1,LOW);
digitalWrite(led2,HIGH);
digitalWrite(led0,HIGH);
delay(500);}
 if(acao=='0'){
digitalWrite(led0,LOW);
digitalWrite(led1,LOW);
digitalWrite(led2,LOW);
digitalWrite(led3,LOW);
digitalWrite(led4,LOW);
digitalWrite(led5,LOW);}}
```

C.3 – EXPERIMENTO 3

```
//BIBLIOECA DO USO DE SERVO MOTORES

#include <Servo.h>
// CONTROLE DO MOTOR DC
//VARIAVEIS 12 E 13 SÃO PARA SINALIZACAO DE DIRECAO
int led2=12;
```

```
int led1=13;
int IN3=3; //PINOS 3 E 5 DE CONEXAO DO DRIVE DA PONTE H
int IN4=5;
int Tm; //VARIAVEL DA TENSAO MEDIA
float DT; // DUTY CICLE
int acao=-5;
void setup(){
pinMode(IN3, OUTPUT);
pinMode(IN4, OUTPUT);
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
Serial.begin(9600);}
//PROGRAMA DE ROTINA
void loop(){
//CALC. DA TENSAO QUE SERA OPERADA NO PROGRAMA
//TENDO UMA CONDICAO DE TENSAO DO MOTOR DC QUE EH DE 5v
// DENTRO DO DUTY CICLE QUE O TEMPO O MOTOR ATUA EM TESAO
// MAXIMA DE 5v SOBRE O TEMPO TOTAL DE OSCILACAO
//O VALOR DO DUTY CICLE VAI DE 0 A 255 QUE EH AFORMA DO VALOR
// DE 8 bits QUE ESTA ARMAZENADO NO ARDUINO.
if(Serial.available()>0){
acao=Serial.read();}
//controle de vel por meio da tensao controlada pelo pwm
//com rotacao horaria.
if(acao=='1'){
Tm=255;
analogWrite(IN3,Tm);
digitalWrite(led1,HIGH);
digitalWrite(led2,LOW);}
if(acao=='2'){
Tm=255*0.5;
analogWrite(IN3,Tm);
digitalWrite(led1,HIGH);
digitalWrite(led2,LOW);}
if(acao=='3'){
Tm=255*0.25;
analogWrite(IN3,Tm);
digitalWrite(led1,HIGH);
digitalWrite(led2,LOW);}
//controle de velecidade por meio da tensao de saida contralada
//pelo pwm com rotacaoanti-horaria.
if(acao=='5'){
Tm=255;
analogWrite(IN4,Tm);
digitalWrite(led2,HIGH);
digitalWrite(led1,LOW);}
if(acao=='6'){
Tm=255*0.5;
analogWrite(IN4,Tm);
digitalWrite(led2,HIGH);
digitalWrite(led1,LOW);}
if(acao=='7'){
Tm=255*0.30;
analogWrite(IN4,Tm);
digitalWrite(led2,HIGH);
digitalWrite(led1,LOW);}
if(acao=='0'){ // commando de parade do motor
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,LOW);
  digitalWrite(led1,LOW);
```

```
  digitalWrite(led2,LOW);}}
```

C.4 – EXPERIMENTO 4

```
   //bibliotecas para uso do servo motor e do sensor ultrassonico
#include <MegaServo.h>
#include <Ultrasonic.h>
//VARAIVEIS DO SENSOR ULTRASONICO
int echoPin=13;
int triggerPin=12;
long duracao=0;
long distancia=0;
String sinal;
//VARIAVEIS DE CONTROLE E VELOCIDADE DO MOTOR A
int IN1=2;
int IN2=4;
int velocA=3;
//VARIAVEIS DE CONTROLE E VELOCIDADE DO MOTOR B
int IN3=6;
int IN4=7;
int velocB=5;
// VARIAVEL DE LEITURA DE COMANDO ENVIADO DA PORTA USB
int acao=-5;

void setup(){
  //VELOCIDADE DE CONEXAO
  Serial.begin(9600);
  //PINOS DOS MOTORES
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(velocA,OUTPUT);
  pinMode(velocB,OUTPUT);
  // ENTRADA DO SINAL UTRASSONICO
  pinMode(echoPin,INPUT);
  // SAIDA DO SINAL ULTRASONICO
  pinMode(triggerPin,OUTPUT);}
void loop(){
  if(Serial.available()>0){
    acao=Serial.read();}
    // ENVIO DAS CONFIGURACOES DAS VELOC. DOS MOTORES
   analogWrite(velocA,170);
   analogWrite(velocB,180);

   //NAVEGACAO DO ROBO
   //PARA FRENTE
   if(acao=='1'){
   digitalWrite(IN1,HIGH);
   digitalWrite(IN3,HIGH);
   digitalWrite(IN2,LOW);
   digitalWrite(IN4,LOW);
   delayMicroseconds(100);}
    //PARA TRAS
   if(acao=='2'){
   digitalWrite(IN1,LOW);
   digitalWrite(IN2,HIGH);
   digitalWrite(IN3,LOW);
   digitalWrite(IN4,HIGH);
```

```
 delayMicroseconds(100);}
 //PARA DIREITA
  if(acao=='3'){
 digitalWrite(IN1,HIGH);
 digitalWrite(IN2,LOW);
 digitalWrite(IN3,LOW);
 digitalWrite(IN4,HIGH);
 delayMicroseconds(100);}
 //PARA A ESQUERDA
  if(acao=='4'){
 digitalWrite(IN1,LOW);
 digitalWrite(IN2,HIGH);
 digitalWrite(IN3,HIGH);
 digitalWrite(IN4,LOW);
 delayMicroseconds(100);}
//PARAR
 if(acao=='0'){
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
 digitalWrite(IN3, LOW);
 digitalWrite(IN4, LOW);}
 // OPERACAO DO SENSOR ULTRASONICO
if(acao=='5'){
 digitalWrite(triggerPin,LOW); DISPARO DO SINAL
 delayMicroseconds(0.5);//TEMPO DE DISPARO DE SINAL
 digitalWrite(triggerPin,HIGH);//RECEBIMENTO DO SINAL
 delayMicroseconds(5);// TEMPO DE RECEBIMENTO DO SINAL
 digitalWrite(triggerPin,LOW);
 duracao=pulseIn(echoPin,HIGH);// ARMAZENAMENTO DO TEMPO
 distancia=duracao/58; // CLIBRACAO DO TEMPO
 if(distancia<50){ // DISTANCIA MAXIMA PERMITIDA PARA MEDICAO
 sinal=String(distancia);
 Serial.println(sinal) + "|"; }//ENVIO PARA A PORTA SERIAL
  else if(distancia>50){// CONDICAO DE FORA DE ALCANCE
  distancia=0;
  sinal=String(distancia);
  Serial.println(sinal)+"|";}
   delayMicroseconds(1);}}// TEMPO DE REPTICAO
```