UNIVERSIDADE DE PASSO FUNDO

INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

COMUNICAÇÃO BIDIRECIONAL PARA PLATAFORMA EMBARCADA DO PROTEGEMED

Maurício Antonioli Schmitz

Passo Fundo

2017

UNIVERSIDADE DE PASSO FUNDO
INSTITUTO DE CIÊNCIAS EXATAS E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

COMUNICAÇÃO BIDIRECIONAL PARA PLAFORMA EMBARCADA DO PROTEGEMED

Maurício Antonioli Schmitz

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Computação Aplicada na Universidade de Passo Fundo.

Orientador: Luiz Eduardo Schardong Spalding
Coorientador: Marcelo Trindade Rebonatto

Passo Fundo
2017

Dados Internacionais de Catalogação na Publicação (CIP)

S335c Schmitz, Maurício Antonioli
Comunicação bidirecional para plataforma embarcada do protegemed /Maurício Antonioli Schmitz. – 2017.
79 f. : il. color. ; 30 cm.

Orientador: Luiz Eduardo Schardong Spalding.
Coorientador: Marcelo Trindade Rebonatto.
Dissertação (Mestrado em Computação Aplicada) – Universidade de Passo Fundo, 2017.

1. Tecnologia apropriada. 2. Programas de computador. 3. Microcontroladores embarcados. 4. Protegemed. I. Spalding, Luiz Eduardo Schardong, orientador. II. Rebonatto, Marcelo Trindade, coorientador. III. Título.

CDU: 004.438

Bibliotecário Responsável

Marciéli de Oliveira

Número do Registro no CRB 10/2113

UPF Universidade de Passo Fundo | PPGCA Programa de Pós-Graduação em Computação Aplicada
Instituto de Ciências Exatas e Geociências - ICEG

ATA DE DEFESA DO TRABALHO DE CONCLUSÃO DE CURSO DO ACADÊMICO

MAURÍCIO ANTONIOLI SCHMITZ

Aos trinta dias do mês de março do ano de dois mil e dezessete, às 14 horas, realizou-se, no Instituto de Ciências Exatas e Geociências, prédio B5, da Universidade de Passo Fundo, a sessão pública de defesa do Trabalho de Conclusão de Curso **"Comunicação Bidirecional para Plataforma Embarcada do Protegemed"**, de autoria de Maurício Antonioli Schmitz, acadêmico do Curso de Mestrado em Computação Aplicada do Programa de Pós-Graduação em Computação Aplicada – PPGCA/UPF. Segundo as informações prestadas pelo Conselho de Pós-Graduação e constantes nos arquivos da Secretaria do PPGCA, o aluno preencheu os requisitos necessários para submeter seu trabalho à avaliação. A banca examinadora foi composta pelos doutores Luiz Eduardo Schardong Spalding, Marcelo Trindade Rebonatto, Marco Antonio Sandini Trentin e Fabiano Hessel. Concluídos os trabalhos de apresentação e arguição, a banca examinadora considerou o candidato APROVADO. Foi concedido o prazo de até quarenta e cinco (45) dias, conforme Regimento do PPGCA, para o acadêmico apresentar ao Conselho de Pós-Graduação o trabalho em sua redação definitiva, a fim de que sejam feitos os encaminhamentos necessários à emissão do Diploma de Mestre em Computação Aplicada. Para constar, foi lavrada a presente ata, que vai assinada pelos membros da banca examinadora e pela Coordenação do PPGCA.

Prof. Dr. Luiz Eduardo Schardong Spalding
Presidente da Banca Examinadora
(Orientador)

Prof. Dr. Marcelo Trindade Rebonatto
(Coorientador)

Prof. Dr. Marco Antonio Sandini Trentin
(Avaliador Interno)

Prof. Dr. Fabiano Hessel
(Avaliador Externo)

Prof. Dr. Rafael Rieder
Coordenador do PPGCA

Dedico este trabalho a minha namorada Assunta Piovesan Neta, a minha mãe Elisabete Antonioli e aos meus avós Genoveva Antonioli e Severino Antonioli (*in memorian*). Vocês são os pilares desta conquista.

AGRADECIMENTOS

Aos meus orientadores Luiz Eduardo Schardong Spalding e Marcelo Trindade Rebonatto por acreditarem no meu potencial em contribuir para o Protegemed, pelos sábios e pertinentes conselhos, pela presença em cada etapa da pesquisa, pela maestria nas sugestões e pelo profissionalismo. Os seus trabalhos e a dedicação ao ensino são invejáveis.

As amáveis mulheres presentes em minha vida, avó Genoveva Antonioli, mãe Elisabete Antonioli e namorada Assunta Piovesan Neta, pelo entendimento nos momentos de ausência, pela força e compreensão nos momentos de dificuldade e pelo apoio incondicional durante esse tempo em que estive dedicado a este trabalho. Sem vocês eu não chegaria tão longe.

Ao Instituto Federal de Educação do Rio Grande do Sul por fornecer condições e apoiar a condução de pesquisas, por permitir e incentivar a evolução acadêmica de seus servidores e por contribuir no desenvolvimento deste trabalho.

"A grande dificuldade não é você ter tudo o que quer,
mas saber querer tudo o que tem.”

COMUNICAÇÃO BIDIRECIONAL PARA PLATAFORMA EMBARCADA DO PROTEGEMED

RESUMO

O uso da energia elétrica conduz a sociedade moderna, principalmente quando utilizada para alimentar equipamentos eletrônicos. Esses equipamentos são essenciais para a vida das pessoas, sendo utilizados nas mais diversas áreas. Dentre elas, a área médica, onde os equipamentos salvam vidas quando em funcionamento normal. Porém, podem vir a falhar e encaminhar parte da energia que consomem para o paciente em contato com eles. Por isso, são necessárias formas de monitorar falhas nesses equipamentos. Uma delas é o projeto Protegemed, que verifica a corrente elétrica dos equipamentos eletromédicos (EEM) durante cirurgias e, caso detecte alguma anomalia, captura os conjuntos de valores instantâneos dessas correntes e os encaminha via conexão de rede para um servidor, onde são analisados por um software de apoio. Contudo, o Protegemed necessita de uma nova funcionalidade, que o torne capaz de modificar variáveis no *firmwa*re do microcontrolador embarcado no painel de tomadas elétricas que alimenta os EEM em uma sala de cirurgia. Esta modificação deve ocorrer, via conexão de rede, cada vez que um EEM é conectado na tomada. A modificação necessária refere-se a duas variáveis, que hoje contêm valores fixos, determinados na instalação do *firmware*, para todos os EEM ligados ao painel. Estas variáveis contêm os limites mínimos de valor eficaz da corrente de fuga e da corrente de alimentação para cada EEM que esteja sendo monitorado. Este limite mínimo, se ultrapassado, dá inicio ao procedimento de alerta de perigo. É um limite que é próprio de cada aparelho e, atualmente, o Protegemed não consegue mudar esse valor no *firmware* do microcontrolador embarcado no painel, usando, inadequadamente, o mesmo valor para, por exemplo, um monitor cardíaco e para um desfibrilador. O Protegemed não consegue fazer esta alteração porque opera com apenas um sentido para o tráfego desses dados, partindo das unidades que monitoram os equipamentos (módulos embarcados no painel de tomadas) para um servidor. Diante disso, este trabalho tem como objetivo ampliar a via de comunicação em um sentido, capacitando o Protegemed com comunicação bidirecional. Para isso, utiliza o protocolo WebSocket, que proporciona comunicação bidirecional, full-duplex e persistente. Com o uso da comunicação em duas vias foi possível enviar comandos para os módulos embarcados. Os resultados dos testes demonstram que a implementação da comunicação bidirecional utilizando o protocolo WebSocket fornece estabilidade e confiança, além de tramitar mensagens entre clientes da comunicação bidirecional dentro de intervalos de tempo compatíveis com a necessidade do Protegemed.

Palavras-chave: bidirecional, comunicação, Protegemed, websocket

BIDIRECTIONAL COMMUNICATION FOR EMBEDDED PLATFORM OF THE PROTEGEMED

ABSTRACT

The use of electricity leads modern society, especially when used to power electronic equipment. These equipment's are essential for the life of the people, being used in the most diverse areas. Among them, the medical area, where the equipment save lives when in normal operation. However, they may fail and direct some of the energy they consume to the patient in contact with them. Therefore, ways to monitor such equipment failures are needed. One of them is the Protegemed project, which checks the electric current of the electromedical equipment (EEM) during surgeries and, if it detects an anomaly, capture the sets of instantaneous values of these currents and forwards them via network connection to a server where they are analyzed by supporting software. However, the Protegemed needs a new functionality that makes it able to modify variables in the firmware of microcontroller embedded in the electrical outlets panel that feeds the EEM in a surgery room. This change must occur, via a network connection, every time an EEM is plugged in. The necessary modification refers to two variables, which today contain fixed values, determined in the firmware installation, for all the EEM connected to the panel. These variables contain the minimum effective value of leakage current and supply current for each EEM being monitored. This minimum limit, if exceeded, initiates the hazard alert procedure. It is a limit that is specific to each device, and the Protegemed project cannot currently change this value in the microcontroller firmware embedded in the panel, using, improperly, the same value, for example, for a heart monitor and for a defibrillator. The Protegemed cannot make this change because it operates with only one direction for the traffic of this data, from the units that monitor the equipment (modules embedded in the outlets panel) to a server. Therefore, this study aims to extend the single way of communication, providing a bidirectional communication for the Protegemed. To do this, it uses the WebSocket protocol, which provides bidirectional, full-duplex and persistent communication. With the use of the bidirectional communication, it was possible to send commands to the embedded modules. Test results demonstrate that the implementation of the bidirectional communication using the WebSocket protocol provides stability and reliability, as well processing messages between clients of the bidirectional communication within time intervals compatible with the need for the Protegemed.

Keywords: bidirectional, communication, Protegemed, websocket

Lista de FIGURAS

Lista de TABELAS

LISTA DE ABREVIATURAS

A – Amperes

Ajax – Asynchronous JavaScript and XML

ANSI – American National Standards Institute

CSS – Cascading Style Sheets

DDP – Diferença de Potencial

EEM – Eletromédicos

FO – Forma de Onda

GUID – Globally Unique IDentifier

HTML – Hyper Text Markup Language

HTTP – HyperText Transfer Protocol

Hz – Hertz

ISO – International Organization for Standardization

JS – JavaScript

NVT – Network Virtual Terminal

RFID – Radio-Frequency Identification

RMS – Root Mean Square

SGBD – Sistema de Gerenciamento de Banco de Dados

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

V – Volt

SUMÁRIO

1. INTRODUÇÃO

A energia elétrica é utilizada nos mais diversos ramos, o uso correto é essencial para a vida das pessoas. Dentre as áreas que aproveitam esta forma de energia, está a medicina, mais propriamente em equipamentos que auxiliam os profissionais deste segmento. Denominados Equipamentos Eletromédicos (EEM), estão presente desde o diagnóstico de pacientes, até procedimentos cirúrgicos complexos, salvando vidas. No entanto, a energia elétrica pode exercer o papel contrário, seu uso incorreto ou a ocorrência de falhas em EEM podem ocasionar danos severos em pacientes, principalmente os que enfrentam procedimentos cirúrgicos, pois estão em situação de vulnerabilidade.

Nesse sentido, o projeto Protegemed [1] foi concebido para fornecer suporte a equipes médicas e profissionais ligados a área médica. A principal finalidade do Protegemed é supervisionar a corrente elétrica utilizada pelos EEM dentro dos centros cirúrgicos. Pois, em eventual problema nesses equipamentos, pode produzir um choque elétrico no paciente.

O Protegemed é composto de três unidades principais, uma que realiza o monitoramento da rede elétrica e captura dados, denominada módulo; outra que recebe os dados obtidos e os armazena em um banco de dados, chamada de servidor; e a última que consulta o banco de dados e exibe as informações capturadas em um software web, conhecida como software de apoio.

A forma de conexão entre o módulo e o servidor é em apenas um sentido (unidirecional), partindo do módulo para o servidor. Este tipo de comunicação supre o envio dos dados capturados pelos módulos. No entanto, apresenta limitações ao projeto por não fornecer a via oposta de comunicação, não permitindo o envio de informações aos módulos.

A falta de comunicação com os módulos traz dificuldades para o projeto. Não é possível, por exemplo, que um usuário do Protegemed, suspeitando de comportamento anormal de um dos equipamentos que operam em determinado centro cirúrgico, realize capturas de dados instantâneos do equipamento para análise. Entretanto, o principal problema é que este usuário não consegue encaminhar parâmetros de configuração para os módulos. Esta é a principal limitação que há no Protegemed.

É necessário fazer uma melhoria no Protegemed que o torne capaz de modificar o arquivo de configurações que está no microprocessador, embarcado no painel de tomadas elétricas que alimenta os EEM em uma sala de cirurgia. Esta modificação do arquivo deve ocorrer, via conexão de rede, cada vez que um EEM é conectado na tomada. A principal

modificação neste arquivo refere-se a duas variáveis que hoje contêm valores fixos, determinados na instalação do *firmware*, para todos os EEM ligados ao painel. Estas variáveis contêm os limites mínimos de valor eficaz da corrente de fuga e da corrente de alimentação para cada EEM que está sendo monitorado. Este limite mínimo, se ultrapassado, dá inicio ao procedimento de alerta de perigo. É um limite que é próprio de cada aparelho e, atualmente, o Protegemed não consegue mudar este valor no *firmware* embarcado no painel, usando, inadequadamente, o mesmo valor para um monitor cardíaco e para um desfibrilador, por exemplo. Esta limitação impede que o Protegemed possa ser adotado como um equipamento de segurança aos pacientes nas salas de cirurgias dos hospitais.

O presente trabalho tem como finalidade a ampliação da comunicação do Protegemed, proporcionando a ele uma comunicação bidirecional (em dois sentidos), substituindo ou incrementando a comunicação unidirecional existente. Com isso será possível o envio de dados para os módulos, eliminando, assim, as limitações da via unidirecional de comunicação.

O desenvolvimento do trabalho constitui diversas etapas, estando separadas da seguinte forma: no Capítulo dois, o Protegemed é apresentado, caracterizando as principais grandezas elétricas, as versões do projeto, os componentes, as funcionalidades e as limitações; no Capítulo três, os conceitos de sistemas embarcados são apresentados, com foco na plataforma MBED, utilizada pelo Protegemed; o Capítulo quatro descreve modelos de comunicação em rede, expondo suas camadas e protocolos; no Capítulo cinco, são descritos dois modelos de solução para comunicação bidirecional; no Capítulo seis, a implementação dos modelos é relatada, bem como as tecnologias utilizadas para elaborar a comunicação em duas vias; no Capítulo sete, são conduzidos testes com as soluções propostas; por fim, o Capítulo oito apresenta as conclusões obtidas, elencando as contribuições para o Protegemed e trabalhos futuros.

2. PROTEGEMED

O Protegemed une hardware e software para detecção de falhas elétricas em equipamentos eletromédicos ligados a pacientes em procedimentos cirúrgicos. Baseia-se na medida das correntes elétricas de alimentação e diferencial de cada EEM [1].

Neste sentido, a abordagem sobre conceitos de eletricidade é importante e será detalhada na seção seguinte, bem como nas próximas seções um relato sobre o Protegemed vai ser apresentado, visando demonstrar a estrutura física e lógica, o *firmware* e as versões do software de apoio do projeto.

2.1. CARACTERÍSTICAS DE ALGUMAS GRANDEZAS ELÉTRICAS

A energia elétrica é fundamental para o convívio social nos dias atuais. Segundo Fowler [2] a eletricidade é uma forma de energia, responsável por manter nossa sociedade em funcionamento adequado quando controlada adequadamente, porém, pode ser muito danosa quando fora de controle.

A produção de eletricidade pode ocorrer em grandes ou pequenas usinas, utilizando para isso geradores. Os geradores de energia elétrica têm por finalidade produzir uma diferença de potencial entre condutores elétricos de metal. Esta diferença de potencial (DDP), conhecida também por tensão elétrica, tem como unidade de medida o Volt (V). Quando aplicada esta DDP em condutores metálicos (normalmente ligas de cobre ou de alumínio) surge um campo elétrico no interior destes condutores. Este campo produz um movimento ordenado de elétrons que varia de acordo com a DDP aplicada. Tal movimento ordenado, chamado de corrente elétrica, é mensurado em Amperes (A).

A tensão e a corrente elétrica podem ser representadas de forma gráfica, exprimindo a relação destas medidas em função do tempo, para essa relação é dado o nome de Forma de Onda (FO). Há uma grande variedade de FO, as principais são: senoidal, quadrada, dente de serra e triangular. A FO da tensão entregue aos hospitais é a senoidal. A FO da corrente depende do tipo de equipamento que está sendo utilizado.

Para o estudo das FO alguns parâmetros devem ser observados, como período, frequência, amplitude e valor eficaz. Tais medidas podem ser verificadas no exemplo de uma FO senoidal na Figura 1 e possuem as seguintes características:

- Período (T) – intervalo de tempo para ocorrer um ciclo completo de uma FO;

- Frequência (f) – é a quantidade de ciclos completos de uma FO em determinado espaço de tempo. Se este tempo for expresso em segundos sua unidade de medida é o Hertz (Hz). No Brasil é adotada a frequência de 60 Hz, ou, 60 ciclos por segundo;
- Amplitude (Up) – é o valor máximo (de pico) que uma FO possui. Há também uma forma de expressar a amplitude considerando a diferença entre os valores máximo e o valor mínimo, é chamada de "amplitude pico a pico" [3];
- Valor Eficaz (Vrms ou Irms)– também conhecido como RMS (do inglês *Root Mean Square*), é a raiz quadrada da média aritmética dos quadrados dos valores instantâneos em determinado espaço de tempo [4]. Os valores RMS da tensão e da corrente são utilizados para conhecermos a potência e a energia consumida por um equipamento, seja qual for a sua FO.

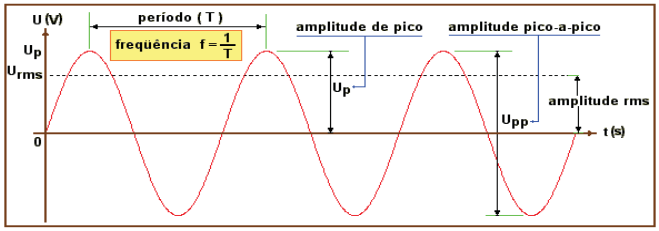


Figura 1 – Forma de onda senoidal, extraído e modificado de [5].

O foco do projeto Protegemed é proporcionar maior segurança aos pacientes submetidos a uma cirurgia quando há um risco de choque elétrico. Este risco, mesmo sendo pequeno, deve ser supervisionado. No caso de cirurgias é mais preocupante o microchoque, pois não é percebido pelos médicos. Segundo alguns autores, como Kindermann, o microchoque é aquele de pequeno valor e que ocorre nos órgãos internos do corpo humano [6], nem sempre perceptíveis às pessoas às voltas da pessoa vitimada. No Protegemed, o microchoque é caracterizado pela passagem de uma corrente pelo corpo humano com um valor menor do que 2,0 miliamperes.

Os microchoques resultam normalmente das correntes de fuga, que são correntes que seguem um fluxo diferente do esperado, podendo atingir um indivíduo em contato com um equipamento elétrico. Segundo Webster [7], os EEM são uma fonte frequente das correntes de fuga. Para identificar os microchoques, o projeto Protegemed utiliza componentes para detecção das correntes e um computador para processar as informações desta corrente.

2.2. ESTRUTURA FÍSICA DO PROTEGEMED

A primeira versão do Protegemed, instalada em uma sala cirúrgica, foi concebida em 2009, sendo que sua funcionalidade é executar, de forma automática, o método para detecção do risco de microchoque através da supervisão da corrente em EEM durante procedimentos cirúrgicos [8]. Para tal, necessitava da presença de um computador dentro do centro cirúrgico, que através da porta serial realiza a comunicação com um dispositivo que monitora o valor da corrente consumida por um EEM e permite monitorar o valor da corrente de fuga do equipamento. O Protegemed gerava alertas para a equipe médica quando estas correntes ultrapassam valores definidos como de risco. A Figura 2 ilustra essa versão inicial do projeto.

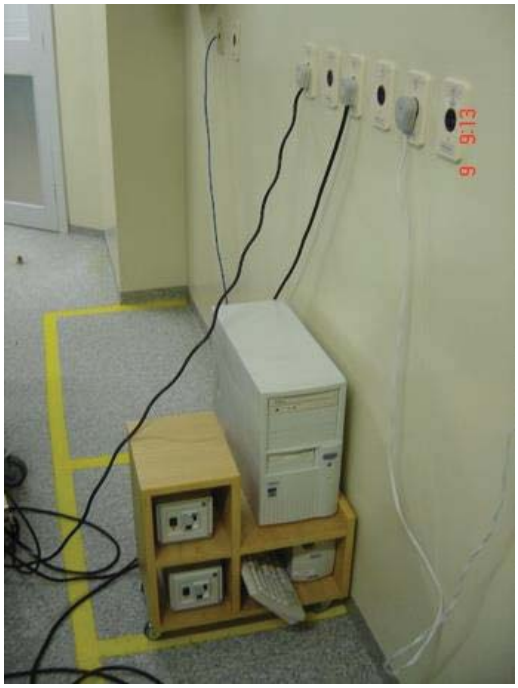


Figura 2 – Protegemed em 2009 [1].

No ano de 2015 uma nova versão do Protegemed foi proposta, denominada Protegemed2 [9]. A nova versão identificou limitações que a versão inicial do projeto apresentava. Dentre elas estão:

- Baixo número de equipamentos monitorados, estes definidos pela quantidade de conexões seriais do computador;
- Presença de um computador dentro do centro cirúrgico, no chão e com alguns cabos de energia mal posicionados;
- Supervisão da corrente de fuga notificando apenas a equipe médica no local, informando que um possível problema estava ocorrendo;

- Inexistência de mecanismos de análise para as FO de corrente diferencial, como escala de periculosidade, por exemplo.

A versão do Protegemed de 2015 pode ser visualizada na Figura 3(a), que apresenta a visão frontal do painel de gases das salas de cirurgia. Já a Figura 3(b) ilustra o aspecto interno do projeto, exibindo os componentes inseridos dentro do painel. Para que o computador fosse adicionado ao painel de gases foi necessário reduzir o seu tamanho físico, a opção, neste caso, foi utilizar computação embarcada.

Figura 3 – Protegemed em 2015, visão frontal (a) e interna (b) [10].

A versão proposta por Rebonatto busca aperfeiçoar a anterior e sugere formas de eliminar as limitações acima descritas. Além de indicar novas funcionalidades para o projeto, como:

- Identificação do EEM através do uso da tecnologia de identificação por radiofrequência RFID (do inglês "*Radio-Frequency Identification*");
- Escalas de periculosidade baseadas no valor da corrente, no espectro de frequência e na similaridade [11] entre FO simultâneas;
- Plataforma de referência projetada para funcionar embarcada no painel de gases e tomadas dos centros cirúrgicos;
- Software de apoio para armazenar os eventos produzidos em um banco de dados, possibilitando a visualização instantânea e consultas futuras dos eventos capturados, o mesmo utiliza as escalas de periculosidade para classificar os eventos.

No estágio atual o Protegemed consegue capturar FO de fuga, gerar alertas de risco de microchoque, calcular a periculosidade deste risco utilizando: valor eficaz da corrente, valor eficaz da corrente para cada frequência entre 60Hz e seus harmônicos até 720Hz e análise da similaridade das formas de onda. A captura dos dados é realizada por um componente embarcado (módulo) que possuí um *firmware* (conjunto de instruções) para monitoraros valores

de corrente elétrica. Os alertas, o cálculo da periculosidade e a análise da similaridade são executados em um software de apoio.

2.3. FIRMWARE

A cada segundo a tensão e a corrente elétrica completam sessenta ciclos, pois a frequência da rede elétrica no Brasil é 60 Hz. O Protegemed opera na análise de cada ciclo individualmente, ou seja, a cada 16,67 ms (1 segundo / 60 ciclos) a corrente é verificada em busca de falhas. Esse trabalho é realizado pelo *firmware*.

Para executar o *firmware* o projeto utiliza o MBED [12], uma plataforma que contém um microcontrolador para realizar o processamento das instruções contidas no *firmware.*

Para possibilitar essa apuração em pouco espaço de tempo, o *firmware* foi desenvolvido utilizando a linguagem de programação C, que segundo Ritchie [13], o autor da linguagem, a sua forma genérica a torna mais conveniente e eficaz que muitas linguagens.

Ao detectar uma medida fora dos limites no valor da corrente o *firmware* captura os dados do equipamento naquele instante e os encaminha para um servidor, que contém um banco de dados, através de uma conexão de rede. Os dados armazenados são examinados, então, por outro mecanismo, denominado software de apoio.

2.4. SOFTWARE DE APOIO

Abstraindo a forma como os dados são obtidos neste momento, o software de apoio é o elemento de fundamental importância para o projeto, pois é nele que a análise dos dados capturados é realizada.

Assim como o projeto Protegemed, o software de apoio possui duas versões. Uma implementada utilizando a linguagem Java e a mais atual concebida em conjunto com a segunda versão do projeto, utilizando linguagens web, principalmente PHP.

2.4.1. Versão em Java

Ao passo que a primeira versão do projeto foi proposta, constatou-se a necessidade de um software para visualização dos dados obtidos dos equipamentos eletromédicos. Esta

etapa foi concebida em um projeto de graduação [14]. Na época a opção foi pelo uso da linguagem de programação Java.

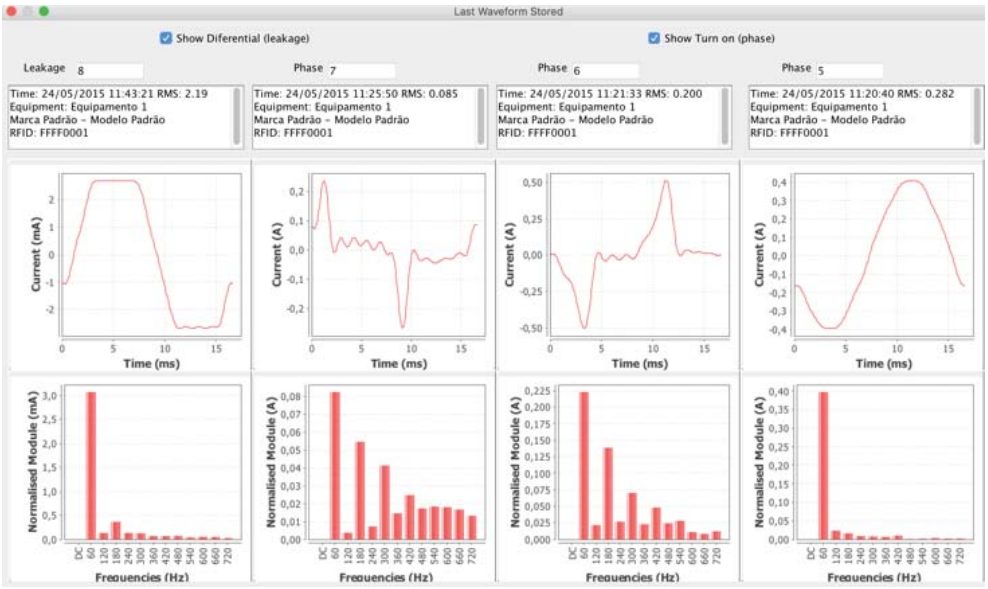


Figura 4 – Software de apoio em Java.

A Figura 4 exibe, como exemplo, uma das telas da versão em Java. Nesta imagem podem ser visualizadas informações das capturas obtidas de um banco de dados. Como a FO, data e hora, equipamento, entre outros dados. Ainda é possível separar as FO de fuga e as de fase.

2.4.2. Versão Web

Com o passar do tempo, a versão utilizando a linguagem Java se mostrou deficiente em alguns aspectos, principalmente por permitir apenas o acesso local dos dados. Dessa forma, em outro trabalho final de graduação [15], foi proposta uma alternativa para a interface inicial do projeto.

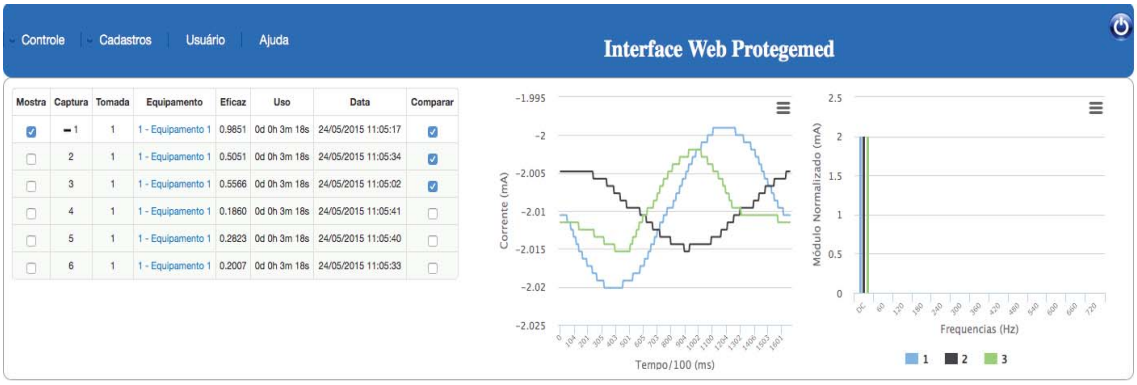


Figura 5 – Software de apoio versão web.

Nesta nova versão, ilustrada na Figura 5, foram utilizadas tecnologias voltadas para web. Nela foram implementadas funcionalidades da versão em Java, como exibir as FO e compará-las ou consultar dados das capturas. Porém, paralelo a esta pesquisa, está em curso, em outro TCC, a inclusão de novas funcionalidades para o software web, como escalas de periculosidades. Ainda a estrutura de serviços (servidor web e banco de dados) está sendo migrada para uma solução virtual, possibilitando a distribuição do projeto em formato de máquina virtual.

O ponto principal do trabalho desenvolvido por Perego [15] foi a possibilidade de centralizar o software de apoio em um servidor, permitindo acesso ao mesmo através de uma conexão de rede. Eliminando, assim, a necessidade de um computador dedicado para o software de apoio, proporcionando, inclusive, o acesso através de dispositivos móveis ao software.

Este avanço foi possível devido as tecnologias utilizadas no desenvolvimento. Dentre elas estão: HTML [16], HTTP [17], PHP [18], CodeIgniter [19], JavaScript [20], jQuery [21], MySQL [22], CSS [23], Ajax [24], HighCharts [25] e BootStrap [26].

3. SISTEMAS EMBARCADOS

O uso de equipamentos eletrônicos é cada vez mais comum, o contato com esse tipo de tecnologia alcança uma vasta gama da população, principalmente o uso de computadores pessoais, pois eles foram desenvolvidos para serem equipamentos de uso geral e genéricos.

No entanto, além dos computadores de propósito geral, há dispositivos que sozinhos não tem a intenção de serem genéricos, mas contém um computador que é programável para determinada aplicação e são conhecidos como sistemas embarcados [27].

3.1. DEFINIÇÃO

Marwedel [28] apresenta outra definição para sistemas embarcados, como sistemas de processamento de informações que são incorporados em um produto maior e que normalmente não são diretamente visíveis para o usuário. Ainda Marwedel lista algumas características comuns desse tipo de sistema, como a frequente conexão dos sistemas embarcados a sensores e atuadores, a necessidade dos sistemas fornecerem confiabilidade e eficiência, além de atenderem a requisições em tempo real.

No quesito confiabilidade, muitos sistemas embarcados são críticos para segurança e por isso necessitam ser confiáveis. Como exemplo de sistemas críticos para segurança estão as usinas nucleares, que, pelo menos, são parcialmente controladas por software e a confiabilidade nesse caso é extremamente importante para segurança. Uma das principais razões para a confiabilidade dos sistemas é que esses estão diretamente ligados ao ambiente em que estão inseridos e produzem um impacto imediato nesse ambiente. Diante disso, a confiabilidade de um sistema abrange os seguintes aspectos de um sistema:

- Confiança: é a probabilidade de que um sistema não irá falhar;
- Manutenção: é a probabilidade de que um sistema com falha possa ser reparado dentro de um determinado período de tempo;
- Disponibilidade: é a probabilidade de que um sistema esteja disponível;
- Segurança (*safety*): descreve a propriedade de que um sistema com falha não causará nenhum dano;
- Segurança (*security*): descreve a propriedade de que os dados confidenciais permaneçam confidenciais e de que a comunicação autêntica seja garantida.

Com relação a eficiência, algumas métricas podem ser utilizadas quando se deseja avaliar se um sistema embarcado é eficiente. Assim sendo, abaixo segue relação contendo alguns exemplos dessas métricas:

- Energia: muitos sistemas embarcados obtêm a sua energia através de baterias, nesse caso o uso da energia deve ser eficiente para proporcionar maior tempo de execução ao sistema [28], bem como um maior consumo de energia produz mais calor e gera a necessidade de aumento na dissipação desse calor [27];
- Tamanho do código: geralmente o código executado pelos sistemas embarcados precisa ser armazenado junto com o sistema, em uma memória integrada. Com isso, a memória deve ser usada da forma mais eficiente e o tamanho do código deve ser o menor possível;
- Tempo de execução: a quantidade mínima de recursos deve ser utilizada para implementar a funcionalidade desejada. Somente os componentes de hardware necessários devem estar presentes, componentes que não melhoram o tempo de execução no pior cenário devem ser omitidos;

A respeito do sistema embarcado atender a requisições em tempo real, caso isso não ocorra, a qualidade do sistema está comprometida. Podendo resultar em uma catástrofe, por exemplo, se o sistema estiver presente em carros, trens ou aviões e não operar da forma esperada.

Em muitos sistemas embarcados o hardware embarcado é utilizando em um loop, onde informações do ambiente são coletadas através de sensores e tipicamente são analógicas, passando por um conversor de analógico para digital (A/D) para serem processadas digitalmente. Os resultados gerados desse processamento podem ser exibidos em uma tela (*display*) ou utilizados para controlar o ambiente através de atuadores, como alguns atuadores são analógicos, há necessidade de converter o sinal digital para analógico (D/A). Um exemplo de hardware de um sistema embarcado é exibido na Figura 6 [28].

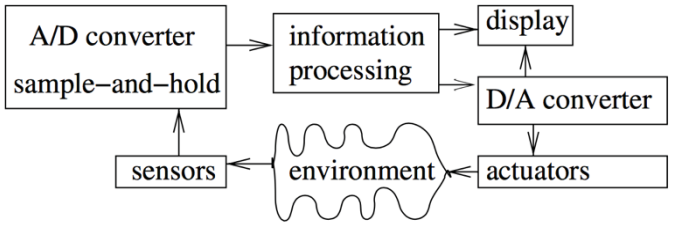


Figura 6 – Exemplo de hardware de um sistema embarcado, extraído de [28].

O exemplo de hardware do sistema embarcado acima pode ser implantado em apenas um chip, essa integração é conhecida como microcontrolador e será o tema da próxima seção.

3.2. MICROCONTROLADOR

Microcontrolador é um computador em apenas um chip, contendo memórias, processador e componentes de entrada e saída. Sua funcionalidade é executar programas dedicados para determinada aplicação. Segundo Toulson [29], um microcontrolador é composto de três partes principais, a CPU, as memórias e os periféricos.

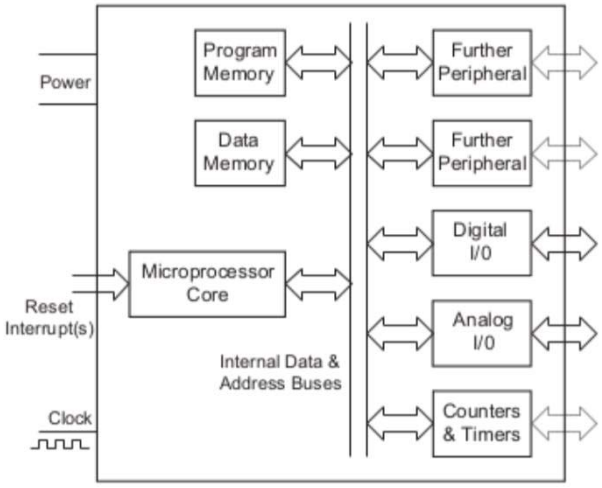


Figura 7 – Exemplo de microcontrolador [29].

Na Figura 7 pode ser visualizado um exemplo de microcontrolador e seus componentes internos, uma parte deles são os periféricos de entrada e saída. Os componentes periféricos permitem a comunicação com elementos exteriores, como componentes de entrada e saída digitais e analógicos, contadores, portas seriais, entre outros [29].

Dentre os mais diversos microcontroladores existentes, o Protegemed utiliza um em específico, o LPC1768. Desenvolvido pela empresa NXP Semiconductors, ele utiliza a arquitetura ARM Cortex-M3 [30] de 32 bits, rodando a 96 MHz, conta com 32 KB de memória RAM e 512 KB de memória flash, além de possuir como principais interfaces de comunicação os padrões USB e Ethernet [12]. O LPC1768 é usado no Protegemed pois é o microcontrolador que equipa a plataforma embarca MBED, utilizada pelo projeto.

3.3. MBED

O MBED é uma plataforma de desenvolvimento para rápida prototipagem, possui um compilador online, bibliotecas de código e uma área contendo documentação. Isso torna o

MBED um ambiente completo de desenvolvimento para sistemas embarcados, permitindo que usuários desenvolvam com simplicidade, eficiência e rapidez.

A placa que compõe a plataforma, exibida na Figura 8, possui 5,3 cm de comprimento por 2,6 cm de largura. A imagem da esquerda da figura abaixo, que ilustra a parte frontal da placa, conta com um conector USB no padrão Mini-B na parte inferior, um botão para reiniciar na parte central e o Microcontrolador LPC1768 na parte superior.

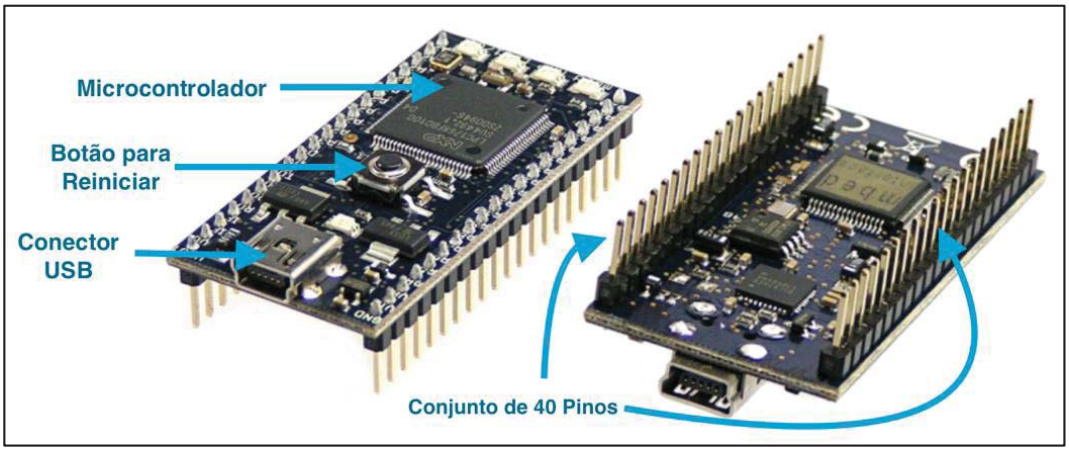


Figura 8 – Plataforma MBED, extraído e modificado de [31].

Nas laterais da figura acima, a direita, podem ser visualizados um conjunto de quarenta pinos, vinte em cada extremidade, a distância entre cada pino é expressa em polegadas, sendo de 0,1 polegadas, esse distanciamento é um padrão em componentes eletrônicos. Os pinos são utilizados em sua maioria para conexões de entrada e saída, como, por exemplo, a conexão de rede no padrão Ethernet

A versão atual do MBED possui um *firmware* implementado utilizando a linguagem de programação C, a codificação foi realizada com auxílio do compilador online da plataforma.

4. MODELOS DE COMUNICAÇÃO EM REDE

Em uma sociedade cada vez mais integrada, a necessidade de uma comunicação eficaz entre as pessoas é primordial. Para tal, o uso de equipamentos eletrônicos se mostra como uma solução, principalmente com auxílio da internet, conectando a população sem a necessidade de deslocamento físico. Esta seção visa apresentar alguns conceitos ligados a forma como essa comunicação é desenvolvida, com base na comunicação exercida pelo Protegemed.

Para Forouzan, redes e sistemas de comunicação de dados são umas das tecnologias com maior crescimento na atual cultura, observados pelo crescente número de profissões e cursos ligados a essas tecnologias [32]. Uma definição de rede é apresentada por Tanenbaum como sendo um conjunto de equipamentos eletrônicos autônomos interconectados por uma única tecnologia [33].

O interesse para este estudo é a forma com que esse tráfego de informações será realizado. Para isso, o desmembramento de um modelo genérico de comunicação para redes, que apenas informa que um dado foi transmitido entre dois equipamentos, em modelos de referência baseados em camadas serão apresentados.

4.1. MODELOS OSI E TCP/IP

Dois modelos de referência se destacam, os modelos OSI e TCP/IP. Para Tanenbaum, os protocolos associados ao modelo OSI são raramente utilizados, porém o modelo e as características de suas camadas são muito importantes. Já o modelo TCP/IP apresenta características inversas, com o modelo não sendo muito utilizado e os protocolos largamente difundidos.

Ambos os modelos apresentam como definição para uma comunicação em rede o princípio de camadas, onde cada uma delas trata uma parcela da informação, permitindo assim garantir que se for enviado o valor 0 (zero) por um equipamento, o outro irá receber esse valor e não o valor 1 (um), por exemplo.

O modelo OSI possui sete camadas, que vão desde o tratamento do sinal elétrico na camada física até a camada de aplicação, responsável por efetuar a interface entre as aplicações e os protocolos. O modelo TCP/IP é mais abstrato, apresentando quatro camadas apenas, a primeira de enlace, que ilustra uma interface entre o computador e a rede; a segunda denominada internet, essa garante que os dados enviados trafeguem até o destino; a terceira de

transporte, que mantém uma conexão entre a origem e o destino; a quarta de aplicação, que contém todos os protocolos de alto nível.

Os modelos possuem algumas semelhanças, como o conceito de pilha de protocolos independentes, onde os dados seguem o fluxo de baixo para cima. As pilhas, contendo as camadas dos dois modelos, podem ser visualizas na Figura 9.

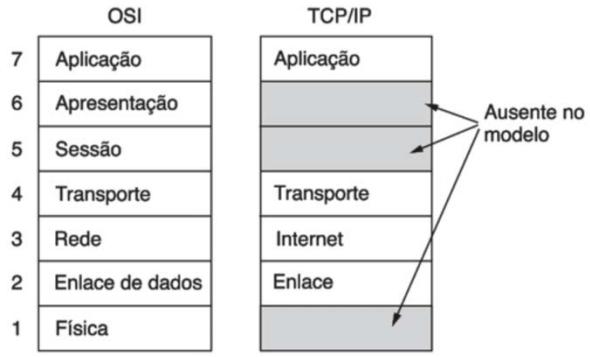


Figura 9 – Modelos de referência OSI e TCP/IP [33].

Outra afinidade entre os modelos é que as camadas possuem as mesmas funcionalidades. Por exemplo, os dois possuem uma camada de transporte e uma de aplicação, para fornecer, respectivamente, um serviço de transporte ponta a ponta e uma camada de aplicação que fará uso da camada de transporte [33]. Estas duas camadas são o foco deste estudo.

4.1.1. Camada de Transporte

A camada de transporte fornece comunicação lógica, e não física, entre processos de aplicações. Estes processos utilizam a comunicação lógica para trocar mensagens entre si, eliminando a preocupação com detalhes físicos da infraestrutura [34].

Para possibilitar a troca de mensagens é necessário que remetente e destinatário utilizem um padrão de comunicação. Similar ao idioma humano, onde um chinês não será compreendido por um brasileiro. Em uma rede a troca de informação deve utilizar uma forma conhecida por ambos os lados. Esse padrão de comunicação é conhecido como protocolo.

Segundo Kurose [34], “um protocolo define o formato e a ordem das mensagens trocadas entre duas ou mais entidades comunicantes, bem como as ações realizadas na transmissão e/ou no recebimento de uma mensagem ou outro evento”.

A camada de transporte possuí dois protocolos para troca de informações. Um deles é o UDP (Protocolo de Datagrama de Usuário, do inglês *User Datagram Protocol*), que fornece

um serviço não confiável e não orientado para conexão. O outro é o TCP (Protocolo de Controle de Transmissão, do inglês *Transmission Control Protocol*), que fornece o oposto do UDP, um serviço orientado para conexão e confiável [34].

4.1.2. Camada de Aplicação

As camadas anteriores a camada de aplicação tem por finalidade fornecer um serviço de transporte, entretanto não executam nenhuma tarefa para o usuário. Para tal, a troca de informações faz uso de protocolos da camada de aplicação [33].

Para realizar a troca de dados a camada de aplicação utiliza uma interface de software com a camada de transporte. Esta interface é conhecida como *socket*, sendo composta por um endereço IP, que identifica um equipamento na rede, em conjunto com um número de porta, que define o protocolo utilizado [34]. A informação da porta é necessária pois sobre um único endereço IP podem ser executados diferentes tipos de protocolos, cada qual com seu número de porta específico.

Os protocolos da camada de aplicação mais populares, como e-mail, acesso a terminais remotos, a Web e a transferência de arquivos utilizam o protocolo TCP para transporte de dados. Essa decisão ocorre, principalmente, devido ao TCP oferecer confiança na entrega dos dados, garantindo a chegada da informação ao seu destino, diferente do UDP, que não fornece tal segurança [34].

Neste estudo há necessidade de se confiar na entrega da informação, com isso o foco da pesquisa foi em protocolos da camada de aplicação que transportam os dados sobre TCP, tais protocolos serão apresentados na seção seguinte.

4.2. PROTOCOLOS ESTUDADOS

Um dos protocolos mais utilizados atualmente é o HTTP (Protocolo de Transferência de Hipertexto, do inglês *HyperText Tranfer Protocol*), especificado na RFC 2616 [17], que opera sobre a troca de mensagem entre cliente e servidor. O HTTP fornece alguns métodos para o envio de mensagens, com os principais sendo o GET, utilizado por um cliente para requisição de mensagens de um servidor, e o POST, usado para enviar mensagens de um cliente para um servidor [34].

O método POST é a base para o envio de mensagem do Protegemed, pois a informação é capturada pelo módulo e enviada para um servidor contendo um banco de dados através desse método.

Em um trabalho de graduação, Fucilini [35] realizou uma comparação entre o protocolo HTTP (que envia por POST), com o uso de um protocolo de comunicação próprio, buscando o que apresentasse melhor desempenho para o Protegemed. Ao final concluiu que ambos mostraram desempenho similar e defendeu a manutenção do protocolo HTTP, por ser um padrão de comunicação em servidores web.

Na época o aluno já comentara sobre a necessidade desta comunicação ser ampliada, pois ela opera em sentido unidirecional (apenas uma direção), com o equipamento embarcado enviando dados para o servidor. Porém, ressaltou que não era o objetivo do trabalho desenvolvido incrementar esta conexão [35].

Um conceito para comunicação nos dois sentidos, ou comunicação bidirecional, é expresso por Kurose como a transmissão de pacotes entre um remetente e um destinatário em ambas as direções [34]. Tal forma de comunicação também pode ser definida como dois canais unidirecionais operando em sentidos opostos.

O desenvolvimento de uma forma de conexão bidirecional é o alvo deste trabalho e se guiará através do uso de modelos cliente-servidor, visando substituir ou incrementar a atual comunicação HTTP.

4.2.1. Servidor HTTP

Devido ao módulo utilizar uma via de comunicação, encaminhando os dados para um servidor HTTP. O incremento dessa via, fornecendo um canal unidirecional oposto, pode suprir a necessidade de envio de dados para o módulo.

Para prover esse segundo canal de comunicação, um servidor HTTP, recebendo conexões, pode ser inserido dentro do módulo. Este servidor inicia um *socket* em uma porta específica e permanece verificando se dados foram encaminhados para essa porta a todo instante. Kurose [34], entende como servidor HTTP um programa que escuta o lado servidor do HTTP e o define como servidor web.

O servidor HTTP, ao receber mensagem, identifica e executa o comando relativo a mensagem. Que pode ser uma solicitação de captura, o qual o servidor direcionará para a captura ser realizada, ou um comando de parâmetros para o módulo, que o servidor procederá com a alteração das configurações informadas.

4.2.2. Telnet

O protocolo Telnet passou por uma série de RFC, iniciando em 1969 na RFC 15 [36], onde era descrito como um subsistema para interligar *hosts*. Depois em 1971 na RFC 137 [37], no qual já era citado como protocolo de rede para interligar sistemas, operando um terminal de um *host* remoto como sendo um terminal local.

Alguns anos passaram e em 1983, através da RFC 854 [38], o Telnet foi especificado com um padrão de comunicação na internet. O seu propósito é prover uma forma de comunicação padrão, bidirecional e de 8 bits. Operando sobre o protocolo TCP, o Telnet foi construído baseado no conceito principal de ser uma rede de terminal virtual NVT (do inglês *Network Virtual Terminal*).

Uma NVT é um dispositivo imaginário que provê um padrão bidirecional (*half-duplex*) para comunicação de rede através de caracteres, permitindo que clientes e servidores se comuniquem sem que saibam as suas características.

Para estabelecer uma conexão Telnet é necessário informar o endereço IP de destino e uma porta, por padrão a porta utilizada é a 23. O transito de dados entre os *hosts* é controlado por caracteres de controle, ou comandos padronizados para indicar o que se deseja realizar. Existem uma série de comandos Telnet, sendo os principais:

- WILL – indica desejo de iniciar, ou confirma que está executando a opção;
- WON'T – indica recusa da opção;
- DO – indica que o outro *host* pode executar, ou confirma que está esperando a execução da opção;
- DON'T – indica que o outro *host* deve parar de executar, ou confirma que não irá mais executar a opção;
- IAC – *Interpret as Command* – deve preceder todo comando Telnet, informa que o próximo byte será um comando Telnet.

Após conectado ao *host*, para que o mesmo saiba que um caractere é de comando, deve sempre ser encaminhando ao menos dois comandos. Primeiro o IAC, para informar que o próximo byte é de comando, na sequência o comando a ser executado. Toda comunicação deve seguir este principio [38].

4.2.3. WebSocket

Oposto da longa trajetória do protocolo Telnet, o WebSocket é relativamente recente, mas já vem apresentando resultados para comunicação bidirecional e persistente entre clientes e servidores. Como no trabalho exposto por Varela [39], que realizou a comparação entre os protocolo WebSocket e HTTP, concluindo que o uso do protocolo WebSocket é mais rápido, mais econômico e permite mais opções que o protocolo HTTP.

O protocolo foi especificado em 2011 através da RFC 6455 [40], o WebSocket provê, através de apenas uma conexão TCP, comunicação bidirecional, full-duplex e persistente. Operando sobre o princípio de cliente-servidor, fornece compatibilidade com infraestruturas atuais e segurança na troca de dados.

Um dos objetivos de seu desenvolvimento foi suprir a dificuldade de realizar comunicação em duas vias utilizando aplicações web. Este tipo de comunicação web necessitava uma série de requisições HTTP para manter cliente e servidor em contato. Dentre os problemas que o protocolo busca eliminar estão:

- O servidor é forçado a usar um número de diferentes conexões TCP para cada cliente, uma para enviar informações ao cliente e uma nova a cada mensagem recebida de cada cliente;
- Sobrecarga de tráfego na rede, pois a cada nova mensagem de clientes, esta deve incluir o cabeçalho HTTP, além do conteúdo da mensagem, informando os dados do servidor a cada envio de dados para este;
- O lado cliente necessita executar scripts para manter um mapa de conexões de saída e de entrada, visando identificar respostas.

Estes empecilhos podem ser evitados utilizando uma única conexão TCP em ambas as direções. O protocolo WebSocket é proposto pensando neste tipo de situação, pois mantém uma conexão TCP persistente e suporta o envio de dados em ambas as direções.

Para realizar este modelo de conexão, o protocolo possui duas partes. A primeira responsável pelas conexões de clientes, utilizando *handshakes* para iniciar ou encerrar conexões. Na segunda parte ocorre a troca de dados.

Para ser possível o envio de informações, o cliente necessita estar conectado ao servidor. A forma de conexão utilizada pelo WebSocket é através de uma requisição do protocolo HTTP. Esta definição busca tornar compatível o uso do protocolo em servidores baseados no protocolo HTTP, mesmo que tais servidores operem utilizando *proxy*, pois faz uso das portas 80 e 443, utilizadas por padrão para conexões HTTP e HTTPS, respectivamente.

A solicitação inicial de conexão utiliza o método GET do protocolo HTTP, informando nesta requisição que se deseja realizar o upgrade do protocolo para WebSocket. Além de informar na requisição o *Host*, a chave de segurança, dentre outras informações. Os dados principais do cabeçalho HTTP para iniciar uma conexão são:

- Host – endereço do servidor WebSocket;
- Upgrade – indica o protocolo a ser adotado;
- Connection – informa atualização de protocolo;
- Sec-WebSocket-Key – chave de segurança no formato Base64 para validar a conexão;
- Sec-WebSocket-Version – versão do protocolo WebSocket.

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Figura 10 – *Handshake* solicitação de conexão, extraído de [40].

Com os dados acima é possível encaminhar uma solicitação de conexão ao servidor, como pode ser visualizado no exemplo da Figura 10. No entanto, para que a conexão seja realizada, o servidor precisa encaminhar ao cliente a confirmação da conexão.

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Figura 11 – *Handshake* confirmação de conexão, extraído de [40].

Para isto, o servidor retorna com um cabeçalho HTTP de atualização de protocolo, o qual pode ser conferido na Figura 11. Sendo mais simples do que o *handshake* do cliente, porém duas informações são essenciais:

- Código estado HTTP – retorna um código de estado da requisição HTTP, dentre inúmeros, como o conhecido 404 (página não encontrada), no caso do WebSocket e a requisição de Upgrade de protocolo, deve retornar o valor 101, informando o sucesso na troca de protocolo. Caso retorne outro código a conexão não será estabelecida;
- Sec-WebSocket-Accept – chave codificada em base64 para confirmar a conexão. Esta chave é baseada na chave enviada pelo cliente, onde é concatenada no servidor com uma chave global única (GUID, do inglês *Globally Unique IDentifier*), aplicada uma função

de dispersão criptográfica SHA-1 e codificada em base64. Este procedimento é realizado também no lado cliente, para poder validar a chave, caso seja divergente a conexão não será estabelecida.

Após validada a conexão entre cliente e servidor, o cliente está apto a encaminhar mensagens. A partir deste momento não é necessário o envio do cabeçalho HTTP novamente, pois a conexão é persistente. Para direcionar mensagens, denominadas pelo protocolo como *frames* (quadros), o cliente necessita informar o tipo de dados que está encaminhando utilizando para isso *opcodes* e a informação em si.

Um *opcode* possui quatro bits e determina a interpretação do servidor para os dados enviados. Definindo se os dados são do tipo texto ou binário, se é a continuação de um *frame*, entre outros. Dentre os principais *opcodes* estão:

- %x0 – denota um *frame* de continuação;
- %x1 – denota um *frame* de texto;
- %x2 – denota um *frame* binário;
- %x8 – denota um *frame* de encerramento de conexão.

Para compor o frame, além do *opcode*, o protocolo prevê a área de *Payload data*, espaço definido para encaminhar os dados. Cada frame pode ser formado por até 18.446.744.073.709.551.615 bytes de dados, valor de um *unsigned integer* de 64-bit [40].

4.3. CONSIDERAÇÕES SOBRE OS PROTOCOLOS

Após o estudo dos protocolos de rede, buscando protocolos que realizassem a comunicação entre cliente e servidor sobre a camada de aplicação, foram encontradas três opções que atenderiam a necessidade, os protocolos WebSocket, Telnet e HTTP.

No entanto, na pesquisa foi verificado que a adoção de um servidor HTTP embarcado no MBED é inviável por questões de recursos. Dessa forma, a etapa seguinte, de implementação dos protocolos, onde serão codificadas suas definições, contará apenas com os protocolos Telnet e WebSocket. Eliminando, assim, o protocolo HTTP, através de um servidor embarcado, da etapa de implementação.

Antes da etapa de implementação com uso dos protocolos WebSocket e Telnet, serão apresentados, na próxima seção, modelos de solução com o uso de ambos os protocolos, visto que cada um possuí características específicas para sanar a necessidade deste trabalho.

5. MODELOS DE SOLUÇÃO

Ao definir a sequencia da pesquisa sobre os protocolos WebSocket e Telnet, o uso de um modelo abstrato para ilustrar o problema a ser solucionado por estes protocolos se faz necessário.

O diagrama de atividades da Figura 12 ilustra o procedimento atual do Protegemed, separado em três blocos, apresentando os passos desde a aquisição dos dados, no *firmware* do MBED, até a sua verificação posterior, no software de apoio.

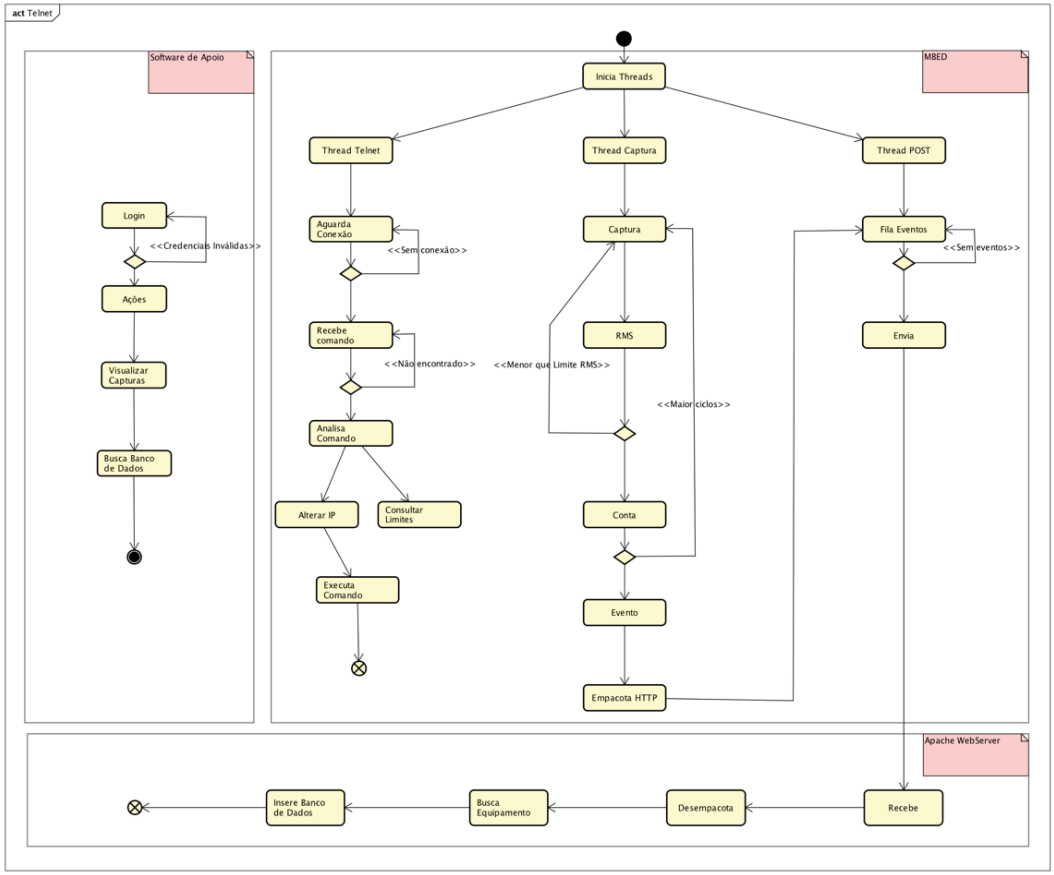


Figura 12 – Diagrama de atividade do Protegemed [10].

No primeiro bloco, parte esquerda da imagem, está o software de apoio. Ele é responsável, por exemplo, pela exibição das capturas realizadas para o usuário. Para isso, realiza comunicação com o banco de dados, apresentando ao usuário os dados em formato de Formas de Onda, espectro de frequência e tabulados.

O *firmware* do MBED está no segundo bloco, parte direita da imagem, nele são executados a cada 16,67 ms uma sequência de testes para definir se os dados obtidos podem representar perigo. Um fator importante no *firmware* é a quantidade de memória RAM

disponível, pois no estágio atual o Protegemed utiliza boa parte da capacidade total desta, apresentando assim um desafio para o desenvolvimento da comunicação bidirecional.

O *firmware* inicia pela captura dos dados elétricos, após a captura o primeiro passo é verificar se o valor RMS é superior ao definido como máximo, em caso positivo um contador é iniciado, cada ciclo contínuo que estiver acima do limite RMS é somado a este contador. O uso de contador é necessário devido a oscilação da rede elétrica, onde podem ocorrer um ou mais ciclos acima do limite RMS e não caracterizar um perigo se não forem persistentes. Ao atingir o número de ciclos contínuos acima do limite definido, um evento é gerado, sendo adquirido o RFID do equipamento, realizado um cálculo matemático, empacotado os dados no formato HTTP e enviado para um servidor web.

No terceiro bloco, parte inferior da imagem, está o servidor apache (servidor web), este recebe os dados coletados pelo MBED, os desempacota, busca o equipamento associado ao RFID e os insere em um banco de dados.

O funcionamento atual do Protegemed apresenta uma característica importante e que limita o projeto. Como pode ser visualizado na Figura 12, a forma de comunicação adotada fornece apenas um caminho para tráfego das informações, do módulo para o servidor. Dessa forma, o envio de dados para o módulo não é possível.

Na sequência, dois modelos de solução para a comunicação com o módulo embarcado serão apresentados, um baseado no protocolo Telnet e outro no protocolo WebSocket.

5.1. COMUNICAÇÃO COM TELNET

Um dos protocolos mais antigos e conhecidos na área de redes, a escolha do Telnet como possível solução para o problema foi definida devido ao *firmware* do módulo já contar com uma implementação prévia de um servidor Telnet para mensagens de controle.

Porém, a comunicação com o atual servidor Telnet embarcado é realizada através dos meios padrões para a tecnologia, necessitando o uso de terminal no cliente e a expertise do operador para interagir com o MBED através de linhas de comando. Ainda, a versão possuí apenas comandos de controle implementados, não contando, por exemplo, com a realização de capturas instantâneas.

Diante disso e da necessidade da comunicação com o módulo ser realizada através do software de apoio (web), um modelo de solução baseado no protocolo Telnet pode ser

visualizado na Figura 13. Visando possibilitar a comparação visual, foi adaptado o fluxograma do Protegemed, ilustrado na Figura 12, para inserir a segunda via de comunicação.

A figura abaixo apresenta três blocos, o software de apoio, o módulo (MBED) e o servidor web (Apache). Este último se manteve inalterado, recebendo os dados provenientes do MBED e inserindo em um banco de dados. No entanto, para suportar a comunicação, os dois outros elementos precisam sofrer alterações.

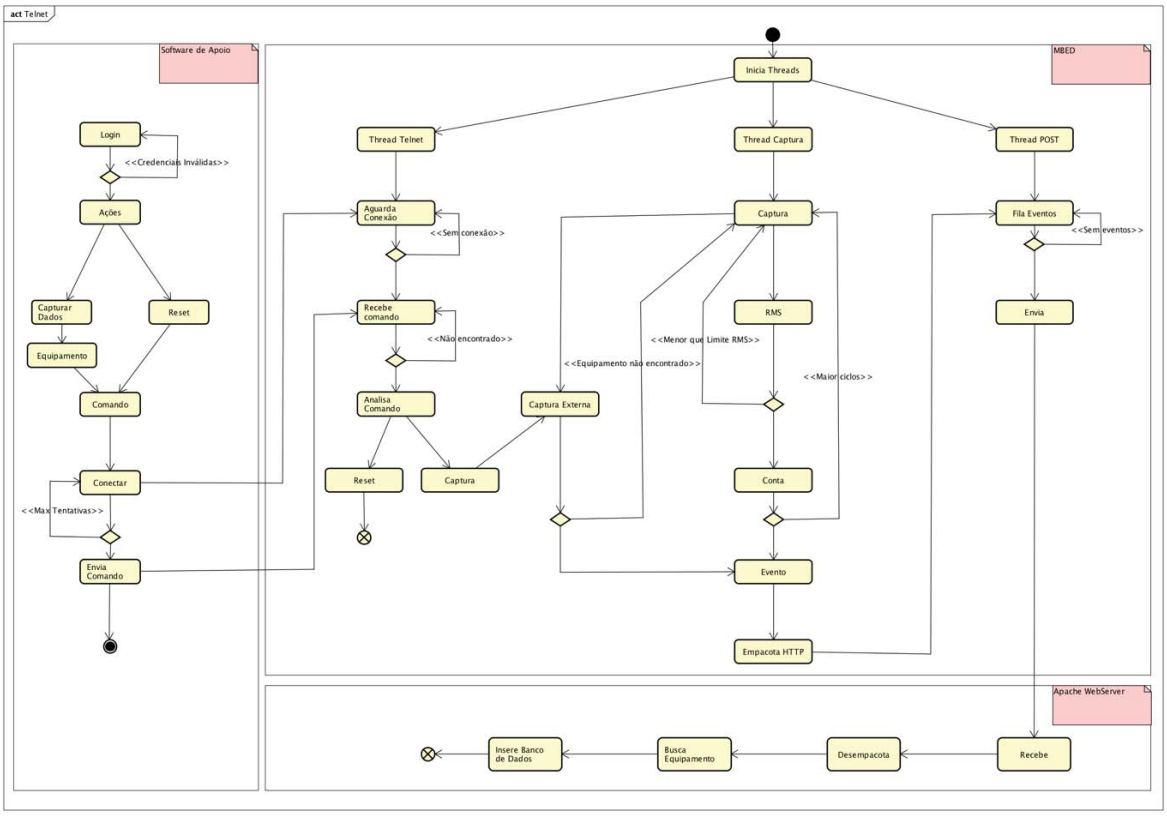


Figura 13 – Diagrama de atividade do modelo de solução Telnet.

Como a comunicação com o MBED será executada, também, pelo software de apoio, as análises dos dados realizadas por ele e demonstradas na Figura 12 foram suprimidas do bloco da Figura 13, mas ainda estão presente no software de apoio. Ao seu lugar foram inseridas as ações para ser possível a comunicação com o MBED. O objetivo, neste caso, é aprimorar a visualização da comunicação em duas vias no software de apoio.

No bloco do MBED as mudanças foram nas *Threads* Telnet e Captura, que passam a contar com ligação entre elas. Esta ligação permite que capturas sejam solicitadas via protocolo Telnet, recebidas pela *Thread* Telnet e direcionadas para *Thread* de Captura obter os dados.

A *Thread* Telnet, em ambos os diagramas, inicia um servidor Telnet no módulo. Sua função é verificar a porta 23 (padrão do protocolo) em busca de conexões de clientes,

enquanto não recebe conexão, fica aguardado. Quando uma conexão é estabelecida, passa a esperar um comando a ser executado, dentro de uma lista de comandos pré-definidos. Se o comando informado é um comando válido, o executa.

Para que o software de apoio consiga encaminhar comandos ao módulo, este deve conhecer as ações que podem ser executadas. Diante disso, após o *login* (identificação) no software de apoio, a próxima etapa é definir a ação que se deseja executar no MBED, dentro de uma lista de possibilidades apresentadas. Com o comando definido, o software inicia tentativas de conexões com o módulo. Caso a conexão ocorra, o comando é enviado e o módulo o executa, finalizando o envio de dados.

No diagrama da Figura 13 dois exemplos de comandos estão descritos, o *reset* do módulo e a captura de dados instantânea de determinado equipamento. Para o *reset* o usuário deve estar credenciado no software de apoio e encaminhar o comando *reset*, o módulo recebe o comando e realiza o *reset* do sistema. Caso a opção seja pela captura de dados, é necessário informar o equipamento ao qual a captura deve ser realizada, o *firmware*, então, recebe o comando de captura (composto do equipamento) e encaminha para *Thread* de captura, que localiza o equipamento e captura os dados, encaminhando estes para o servidor apache inserir no banco de dados através da *Thread* POST. Nesse ponto o software de apoio recebe a atualização do banco de dados e exibe a captura para o usuário.

5.2. COMUNICAÇÃO COM WEBSOCKET

O modelo de solução com WebSocket é similar ao exposto com Telnet. Ambos mantêm as *Threads* de captura e de POST e compartilham as mesmas características na *Thread* WebSocket (Telnet no modelo anterior) e na execução no software de apoio. No entanto, o modelo com WebSocket apresenta um bloco adicional para viabilizar a conexão, um servidor de WebSocket.

Diferente do modelo Telnet, em que a conexão é realizada diretamente entre o software de apoio e o módulo. O modelo com WebSocket opera com a centralização da comunicação em um servidor dedicado para troca de mensagens. Onde o software de apoio e o MBED atuam como clientes deste servidor. O modelo pode ser visualizado na Figura 14.

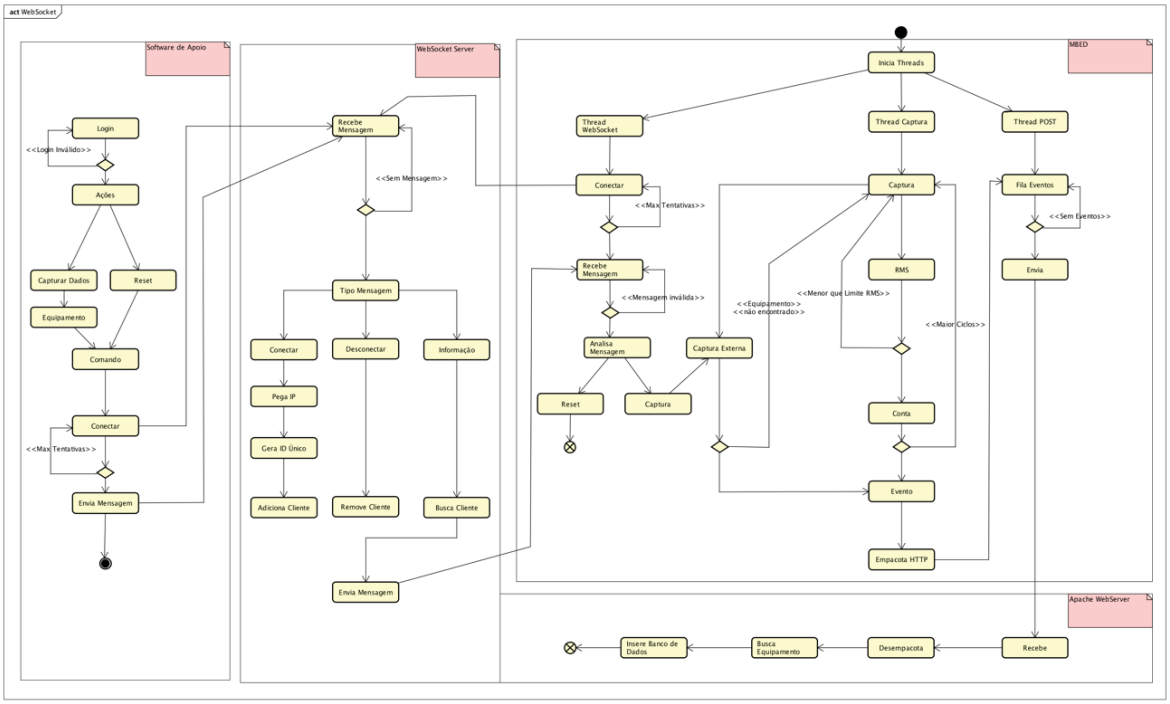


Figura 14 – Diagrama de atividades do modelo de solução WebSocket.

Com a adição do servidor WebSocket, que recebe tanto mensagens de conexão e desconexão quanto mensagem contendo informação pelo mesmo caminho, a primeira etapa do elemento é verificar o tipo de mensagem recebida. No modelo WebSocket as mensagens para o servidor de WebSocket podem ser de três tipos:

- Conexão – mensagem inicial de um novo cliente para se conectar ao servidor. Após receber a mensagem o endereço IP do cliente é armazenado, atribuído a um ID único WebSocket e o cliente é adicionado a lista de clientes conectados;
- Desconexão – mensagem para o servidor remover o cliente da lista de clientes conectados;
- Informação – mensagem contendo dados, o destinatário da informação deve ser informado. O servidor então busca o cliente informado na lista dos conectados e o encaminha a mensagem.

Os comandos exemplos do modelo Telnet estão presentes no modelo WebSocket também. A sequência para o usuário executar os comandos é a mesma. Porém, para ocorrer o envio dos comandos, o software de apoio e o módulo devem estar conectados no servidor WebSocket.

Para que o comando de *reset*, por exemplo, seja encaminhado ao MBED, este deve constar na lista de clientes conectados do servidor WebSocket. O software de apoio, então, realiza a conexão com o servidor WebSocket através do envio da mensagem de conexão para

o servidor, após conectado não há mais necessidade do envio deste tipo de mensagem, pois a conexão com o servidor é persistente. Com o sucesso da conexão o software de apoio deve encaminhar uma nova mensagem de informação, contendo o comando de *reset* e o cliente destino desta mensagem, neste caso o MBED. O servidor WebSocket busca o cliente informado e direciona o comando de *reset* para o mesmo. Este recebendo o comando e o identificando, procede com a reinicialização do sistema.

O modelo WebSocket com quatro blocos e o modelo Telnet com três blocos apresentam duas formas visualmente similares de encaminhar os dados, porém muito distintas quando implementadas. A próxima seção aborda as implementações dos modelos e suas peculiaridades.

6. IMPLEMENTAÇÕES

Após definição teórica e apresentação de dois modelos de solução para a comunicação bidirecional, o próximo passo é implementar o planejado. Para isso, serão utilizadas tecnologias já presentes no Protegemed, como HTML, MySQL e PHP, bem como ferramentas adicionais de apoio ao desenvolvimento, como a biblioteca Ratchet.

Visando tornar a inserção da via adicional de comunicação compatível com o Protegemed, a etapa de implementação da comunicação bidirecional buscou seguir o padrão de tecnologias usadas pelo projeto. A próxima seção tem por finalidade apresentar uma breve descrição de cada uma. Nas seções seguintes serão demonstradas alterações realizadas no banco de dados do projeto, versões do *firmware* com telnet e WebSocket, servidor de WebSocket e por fim as alterações conduzidas no software de apoio.

6.1. TECNOLOGIAS UTILIZADAS

O projeto Protegemed une diversas tecnologias, bibliotecas e linguagens de programação. Dentre elas estão C/C++, HTML, CSS, PHP, CodeIgniter, JavaScript, jQuery, Ajax, MySQL, Ratchet e BootStrap.

A linguagem C tornou-se amplamente conhecida como a linguagem do sistema operacional UNIX, proveniente das linguagens BCPL e B, Dennis Ritchie, seu fundador, a desenvolveu nos laboratórios da Bell. Em pouco tempo se tornou padrão no mundo através da cooperação entre a *American National Standarts Institute* (ANSI) e a *International Organization for Standardization* (ISO), organizações que definem padrões mundiais. O C++ é a extensão do C, dentre vários recursos que a aprimoram, o principal é a capacidade de programação orientada a objetos [41].

O *Hyper Text Markup Language* (HTML) é uma linguagem para marcação de hipertexto utilizada com o propósito de produzir documentos e páginas Web. O seu desenvolvimento iniciou em 1990 e evolui até que em 2008 a W3C (organização responsável pela padronização da *World Wide Web*) publicou a especificação para a sua versão 5, última versão lançada [16]. A linguagem é focada em quatro princípios: compatibilidade, utilidade, interoperabilidade e acesso universal. Nesta versão, cinco novas etiquetas foram criadas e algumas alteradas a fim de definir um padrão e melhorar tanto o desenvolvimento quanto o resultado final para o usuário [42].

O *Cascading Style Sheets* (CSS) é a linguagem responsável por controlar o visual da informação exibida pelo código HTML. O conteúdo é formatado pelo CSS para que a sua exibição seja visualmente agradável, independente do meio de acesso utilizado, seja em tablets, smartphones, computadores ou qualquer outro equipamento que consuma a informação, o visual dela será formatado pela linguagem CSS [43].

O PHP é uma linguagem de código aberto capaz de gerar conteúdo dinâmico na *World Wide Web*, possuí o seu código interpretado no lado servidor pelo módulo PHP, não ficando visível ao usuário. O surgimento foi em meados de 1994 e após dez anos, em 2004, foi lançada a sua versão 5. Esta versão trouxe várias inovações, uma delas é a orientação a objetos que foi totalmente reescrita, outra funcionalidade é o suporte para arquivos XML melhorado, bem como o suporte ao MySQL via extensão MySQLi, ainda nesse lançamento o gerenciamento de memória foi aprimorado [44].

O *framework* de desenvolvimento para linguagem PHP CodeIgniter provê um conjunto de bibliotecas para necessidades comuns, assim como uma interface simples e estrutura lógica para acesso as bibliotecas [19].

A linguagem de programação JavaScript (JS) é de propósito geral, dinâmica e possui características do paradigma de orientação a objetos. Foi lançada em 1995 e seu nome se deve à similaridade com a sintaxe da linguagem Java, embora as linguagens não possuam outra relação além desta. Ela está presente, junto com HTML e CSS, na maior parte dos sites da internet, enquanto a linguagem HTML é responsável pelo conteúdo e a linguagem CSS pela formatação visual, a linguagem JavaScript tem por finalidade tornar o conteúdo mais dinâmico. Diferente do PHP, o JS é executado no navegador do cliente, o que otimiza a interação com os usuários através de animações mais complexas, de validação de formulários, entre outras funcionalidades que a linguagem pode desempenhar [45].

Com a advento da utilização da linguagem JS e da sua importância no meio Web, foram sendo desenvolvidas diversas bibliotecas, uma das principais é a *jQuery*. O desenvolvimento da biblioteca iniciou em 2005 e continua ainda sendo aperfeiçoada, sobre o lema “escreva menos, faça mais”, o ponto forte da biblioteca está na simplificação de códigos JavaScript [46].

O Ajax (acrônimo de *Asynchronous JavaScript and XML*) é uma técnica de desenvolvimento web que combina tecnologias como JavaScript e XML, visando tornar páginas mais interativas e dinâmicas. Para isso, utiliza solicitações assíncronas ao servidor web, possibilitando a atualização de conteúdo de determinada página sem a recarregar [47].

O MySQL é um Sistema de Gerenciamento de Banco de Dados (SGBD), sendo servidor e gerenciador de banco de dados. Inicialmente foi projetado para trabalhar com aplicações de pequeno e médio porte, mas atualmente atende aplicações de grande porte e com vantagens sobre os seus concorrentes. A origem do MySQL foi na década de 90, através dos desenvolvedores David Axmark, Allan Larsson e Michael Widenius, utilizando a linguagem de programação C e C++ [48].

A biblioteca Ratchet fornece um conjunto de ferramentas, que fazem uso da linguagem PHP para implementação do servidor de WebSocket. Com o auxílio da biblioteca Ratchet é possível realizar a troca de informações entre clientes e servidor de forma bidirecional e em tempo real [49].

O conjunto de softwares MAMP, acrônimo para **M**ac OS X (sistema operacional), **A**pache (servidor web), **M**ySQL (banco de dados) e **P** de **P**HP, **P**erl ou **P**ython (linguagens de programação), adiciona ao sistema um servidor Web através do Apache, um banco de dados MySQL e ferramentas de gerenciamento. Nas ferramentas estão três gerenciadores de banco de dados (phpMyAdmin, Sequel e MySQLWorkbench), além de suporte a diferentes versões do PHP, módulo de cache para acelerar a execução de PHP, entre outras funcionalidades [50].

O *framework Bootstrap* possui o tema “projetado para todos e em qualquer lugar”, vem sendo uma das principais escolhas para o desenvolvimento web responsivo. A ideia é que se desenvolva apenas uma página e esta se adapte a heterogeneidade de dispositivos e sistemas, como diferentes navegadores, diferentes resoluções, diferentes tamanhos de tela, diferentes dispositivos e a exibição deve se adaptar a todas essas desigualdades. O *Bootstrap* foi inicialmente desenvolvido por Mark Otto e Jacob Thorton, engenheiros do *Twitter,* criado como uma definição de estrutura de código. A biblioteca foi lançada em 2011 como um projeto de software livre e é uma coleção de vários elementos e funções para projetos web [26].

Além das tecnologias acima, a etapa de implementação contou com alguns softwares para auxiliar o desenvolvimento. As linguagens HTML, PHP e JavaScript foram codificadas com auxilio dos ambientes de desenvolvimento integrado NetBeans e Atom.

Para o desenvolvimento do *firmware* a ferramenta utilizada foi o compilador online da plataforma MBED, que permite a codificação em diversas linguagens, dentre elas a linguagem C/C++ (definida para o *firmware*). Além de fornecer bibliotecas para auxílio no desenvolvimento e controle de versões. Por ser o editor online da plataforma, é otimizado para o ecossistema MBED [51].

6.2. BANCO DE DADOS

Para possibilitar o funcionamento adequado da comunicação bidirecional, foi necessário incrementar o sistema de armazenamento de dados utilizado pelo projeto. As alterações no banco de dados foram: adição de tabela para identificação dos módulos; alteração na tabela dos equipamentos para suportar a definição dos limites; criação de novos eventos para identificar uma captura externa e o início e fim do modo de *standby*.

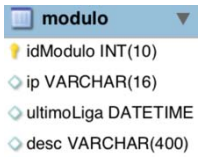


Figura 15 – Tabela modulo adicionada ao banco de dados.

Para possibilitar o gerenciamento e identificação dos módulos, foi inserido no banco de dados uma tabela dedicada as informações destes, como pode ser visualizado na Figura 15, contando com os seguintes campos:

- idModulo – identificador único do banco de dados para cada módulo;
- ip – endereço IP do módulo;
- ultimoLiga – armazena a data e hora da última conexão do módulo;
- desc – destinado para descrição do módulo.

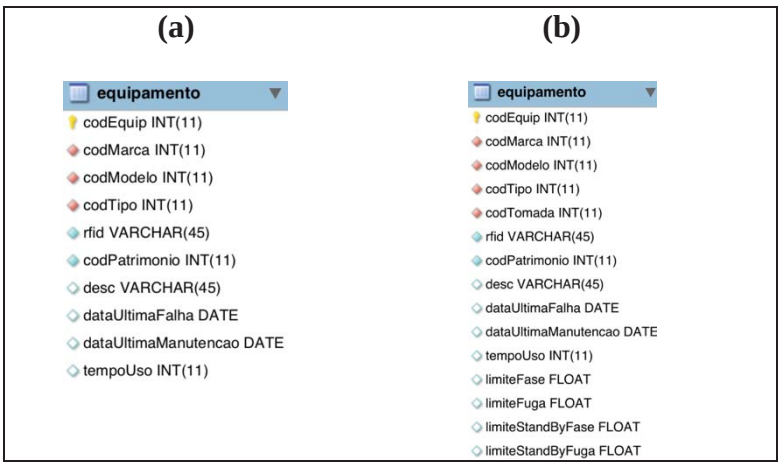


Figura 16 – Alterações na tabela equipamento.

Visando permitir o envio de limites de fase e de fuga por equipamento, tanto em operação normal quanto em *standby*, alterações na tabela de equipamentos foram realizadas e podem ser comparadas entre as Figura 16 (a) e (b). Além da adição dos limites, a tabela já contava com dados referente ao código único do equipamento e códigos para ligação entre

tabelas (marca, modelo, tipo e tomada). Ainda possuí um espaço para armazenar o código RFID, o código do patrimônio institucional, descrição do equipamento, data da última falha e da última manutenção e o tempo de uso total do equipamento.

Na tabela que registra as modalidades de eventos gerados nos módulos embarcados, além das informações de evento de fase ou de fuga, de liga do equipamento ou desliga, foram inseridos campos referentes as capturas externas e início/término de *standby* para os canais de fase e de fuga, como pode ser verificado na Figura 17.

```
INSERT INTO `protegemed`.`eventos` (`codEvento`, `desc`, `formaOnda`) VALUES
(9, 'Captura Externa Fase', 1),
(10, 'Captura Externa Fuga', 1),
(11, 'Inicio StandBy Fase', 1),
(12, 'Inicio StandBy Fuga', 1),
(13, 'Término StandBy Fase', 0),
(14, 'Término StandBy Fuga', 0);
```

Figura 17 – Adição de novos eventos na tabela eventos.

Os eventos da figura acima são essenciais para identificar capturas externas e eventos de *standby* provenientes dos módulos, permitindo que os dados sejam exibidos no software de apoio por categorias, como, por exemplo, apenas capturas externas de fase.

6.3. FIRMWARE MBED

O *firmware* caracteriza uma etapa importante dentre as implementações. Nele é executada toda análise da corrente e realizado as capturas dos dados para o Protegemed, utilizando como componente embarcado para execução do *firmware* o MBED.

Para realizar a implementação o MBED fornece um compilador online, que prove uma interface de desenvolvimento leve, possibilitando a fácil escrita de programas, compilação e download do binário para ser executado no módulo. Isto é possível pelo fato da interface ser um aplicativo web, permitindo o desenvolvimento através dos principais navegadores, como Safari, Internet Explorer, Chrome, Firefox e Opera, rodando sobre os sistemas operacionais Windows, Linux, macOS, iOS e Android. Por ser online, dispensa instalação de software, basta se identificar na plataforma para iniciar a implementação [51].

Diante do exposto, os dois modelos de solução propostos foram implementados através da ferramenta online disponibilizada, codificados na linguagem C/C++. Uma descrição do funcionamento de cada um é o tema das próximas seções.

6.3.1. Firmware com comunicação bidirecional usando o Telnet

O *firmware* com Telnet inicia uma *Thread* dedicada para comunicação do protocolo. Para prover a conexão externa o módulo atua como servidor Telnet, recebendo dados de clientes conectados a ele.

Para possibilitar a ligação no servidor embarcado, um *socket* TCP é iniciado e permanece aguardando conexões na porta do protocolo telnet em um laço de repetição. O cliente que desejar encaminhar dados para o servidor deve ter conhecimento do endereço IP do componente e iniciar uma conexão através do protocolo Telnet neste IP.

A forma de comunicação do cliente com o servidor embarcado é através de comandos, para isso o cliente deve ter conhecimento prévio dos comandos suportados ou pode encaminhar o comando “help” (ajuda) para obter a relação de comandos possíveis.

Na atual versão do *firmware* alguns comandos de controle já estão presentes, como a alteração do endereço IP do servidor para envio das capturas e do endereço do módulo embarcado, assim como consulta de parâmetros de configuração, entre outros. No entanto, não há comando para capturas instantâneas, por exemplo, para possibilitar a conexão entre o servidor de Telnet com a área de capturas do módulo embarcado, sendo desenvolvido este no presente trabalho.

O servidor ao receber um comando do cliente o armazena em um *buffer*, analisa se está dentre os comandos permitidos e o processa. Cada comando está inserido em uma função no *firmware*. Ao receber o comando a função correspondente é iniciada e no final encaminha para o cliente o resultado da execução, informando se o comando realizou a operação com sucesso ou retornando dados, se for o caso.

Se o comando encaminhado for o de captura, por exemplo, o servidor Telnet embarcado o identifica e aciona a função correspondente dentro da *Thread* Telnet. A função informa para *Thread* de Captura os parâmetros para realizar a obtenção dos dados, esta os adquire e os encaminha para *Thread* de POST, que envia para o banco de dados.

Como o Telnet trabalha em *half-duplex*, enquanto um dos lados encaminha informação, o outro lado, caso deseje trafegar dados em simultâneo, não conseguirá. Fornecendo, assim, um canal bidirecional não simultâneo.

6.3.2. Firmware com comunicação bidirecional usando o WebSocket

A implementação do modelo WebSocket adiciona uma nova *Thread* no *firmware* para se comunicar através do protocolo WebSocket com o servidor de WebSocket. Além disso, ela se comunica internamente com outras *Threads*, como a que processa os eventos de captura, por exemplo, utilizando, para isso, variáveis estáticas globais.

A *Thread* WebSocket começa com a busca do endereço IP do servidor, definido nas configurações do *firmware*, direcionando após uma requisição de conexão para este endereço. Caso a conexão ocorra, o módulo encaminha uma mensagem inicial para o servidor informando que está ativo e buscando parâmetros de configuração. O lado servidor verifica em um banco de dados as informações referentes ao módulo (MBED) através do seu endereço IP, como limites para cada tomada, encaminhando estes dados após receber a mensagem inicial.

A *Thread* no módulo, então, inicia um laço de repetição que verifica o estado da conexão. Se ocorrer queda, realiza uma nova tentativa de conexão ao servidor em um intervalo de tempo personalizável (para este estudo foi definido dez segundos). Cabe ressaltar que em caso de indisponibilidade do servidor de WebSocket, a implementação do *firmware* conta com uma forma alternativa para envio dos dados, apenas envio, não há forma alternativa para receber comandos no módulo. Ao detectar novas informações a serem enviadas, a *Thread* WebSocket analisa a conexão com o servidor de WebSocket, caso não haja contato, encaminha os dados através de uma implementação similar ao método sem comunicação bidirecional, enviando por POST. No entanto, sem a necessidade de utilizar uma nova *Thread* para isso, iniciando e encerrando o envio por POST dentro da própria *Thread* WebSocket.

A modalidade de envio alternativa visa aprimorar a entrega dos dados coletados, sendo acionada somente em quedas do servidor de WebSocket. Caso a conexão apresente sucesso, os dados são enviados pelo protocolo WebSocket. A *Thread* WebSocket, além de enviar dados, permanece monitorando mensagens com comandos provenientes do servidor. Quando detecta uma nova mensagem, verifica qual comando foi encaminhado e o executa.

Três comandos foram implementados para serem utilizados com a comunicação bidirecional, estes podem ser visualizados na Tabela 1. Adicional a cada comando, na mesma mensagem, é possível encaminhar parâmetros complementares, que definem a execução do comando. A forma de envio dos parâmetros é com o uso do símbolo de dois pontos “:” separando os parâmetros do comando e de outros parâmetros (se necessário).

Tabela 1 – Comandos WebSocket.

Comando	Parâmetro 1	Parâmetro 2	Parâmetro 3	Exemplo
Capturar	Tomada	Canal (Fase ou Fuga)	-	capture:1:p
Limite	Tomada	Canal (Fase ou Fuga)	Valor	setLimit:1:p:2
Reiniciar	-	-	-	reset

A *Thread* WebSocket ao receber o comando de captura, por exemplo, identifica os parâmetros e informa para a *Thread* de captura a tomada e o canal que os dados devem ser adquiridos. Esta obtém os dados e os encaminha para a *Thread* de WebSocket, onde são direcionados ao servidor WebSocket, este os recebendo, armazena em um banco de dados.

6.4. SERVIDOR WEBSOCKET EM PHP

O modelo de comunicação bidirecional com uso do protocolo WebSocket prevê um servidor para gerenciar as conexões de clientes e a troca de mensagens entre eles. No entanto, não especifica a linguagem de programação que deve ser adotada para implementar esse servidor.

A escolha desse trabalho para o servidor foi pela linguagem PHP, mas poderia ser implementado em Python, NodeJS, Java, entre outras. A definição do PHP se baseou na evolução do software de apoio do Protegemed, que ao ser implementado para versão web fez uso, dentre outras linguagens, principalmente de PHP.

Como base para implementar o servidor WebSocket em PHP foi utilizada a biblioteca Ratchet [38], que fornece um conjunto de componentes para realizar a interface entre cliente e servidor. Na Figura 18 pode ser visualizado um trecho do código com alguns destes componentes.

```
require_once "ws-comunicacao.php";
$server = IoServer::factory(
        new HttpServer(
            new WsServer(
                new Comunicacao()
            )
        ),
        8080
    );
$server->run();
```

Figura 18 – Código para iniciar o servidor de WebSocket.

No código acima é possível visualizar a inclusão da página ws-comunicação.php, que faz o gerenciamento dos clientes. Para possibilitar o recebimento de mensagens, os clientes

devem estar conectados ao servidor. Para que isso ocorra, a variável $server recebe alguns parâmetros:

- IoServer – é a base para o servidor WebSocket, sendo responsável por receber novas conexões, enviar e receber informações destas conexões, terminar a conexão e tratar erros;
- HttpServer – este componente é responsável por receber as requisições HTTP, dentre elas as de upgrade de protocolo;
- WsServer – utilizando em conjunto com o IoServer, permite a conexão com os navegadores;
- Comunicação – inicia a classe que gerencia os clientes;
- 8080 – porta definida para receber comunicações.

Após a variável receber os parâmetros, o servidor é iniciado através do comando $server-run(), permitindo a partir deste momento a conexão de clientes na porta especificada. O gerenciamento dos clientes é realizado através da classe Comunicação, que fornece quatro tipos de eventos para as mensagens dos clientes. A Figura 19 ilustra estes eventos através das funções:

- onOpen – trata as conexões de novos clientes. Cada um é adicionado a uma lista de clientes conectados e recebe um identificador único, além do servidor armazenar o seu endereço IP;
- onMessage – evento para tratar as mensagens dos clientes, toda troca de mensagem é gerenciada por esta função;
- onClose – evento que finaliza a conexão de um cliente, o removendo da lista de clientes conectados e liberando o seu ID;
- onError – evento para tratar erros.

```
class Comunicacao implements MessageComponentInterface {
    public function onOpen(ConnectionInterface $conn) {
    }

    public function onMessage(ConnectionInterface $from, $msgRecebida) {
    }

    public function onClose(ConnectionInterface $conn) {
    }

    public function onError(ConnectionInterface $conn, \Exception $e) {
    }
}
```

Figura 19 – Eventos da classe comunicação.

Os eventos definem a forma como um cliente pode interagir com o servidor, sendo possível a este encaminhar mensagens e conectar ou desconectar do servidor. As mensagens

podem ser direcionadas a outros clientes ou executarem funções implementadas no servidor de WebSocket, como, por exemplo, o acesso a um banco de dados.

O objetivo para o projeto é que o servidor trate mensagens entre navegadores web e componentes embarcados. Possibilitando, por exemplo, no caso do módulo, passar parâmetros iniciais de configuração e encaminhar limites para capturas, entre outras opções viáveis. Com auxílio do software web é permitido realizar capturas em tempo real no módulo e reiniciá-lo. Como essas etapas são realizadas dentro da função onMessage, foi necessário desenvolver um método para identificar o tipo de comando e a qual módulo deveria ser direcionado. Para isto, foi definido um padrão de envio de mensagens, que pode ser visualizado na Figura 20.

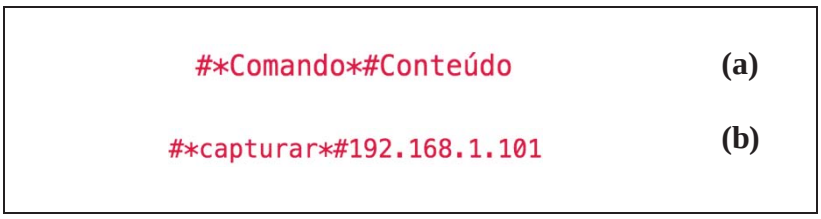


Figura 20 – Padrão de mensagens.

O padrão de mensagens é separado em duas partes, na primeira, entre os símbolos "#*" e "*#", é o espaço destinado ao comando que se deseja executar, a segunda é o conteúdo que será enviado por parâmetro ao comando, sendo este opcional.

A Figura 20(a) apresenta o padrão de mensagem de forma genérica. Na Figura 20(b) pode ser conferido um exemplo de mensagem, com o comando "capturar" que irá executar a função de captura no servidor, passando por parâmetro o endereço IP do módulo que a captura deve ser realizada.

6.5. SOFTWARE DE APOIO EM PHP

Como parte integrante do Projeto, o software de apoio é onde ocorre a interação do usuário com o projeto. Pois o módulo, o *firmware*, os servidores e o banco de dados são elementos de suporte, analisando e processando as informações dos equipamentos. No software de apoio o usuário pode visualizar os dados capturados, receber alertas de periculosidade, realizar capturas, reinicializar módulos, entre outras funcionalidades.

Como descrito na seção 2.4, o software de apoio evolui da versão em Java para versão Web, utilizando como base a linguagem PHP. Dessa forma, a implementação da comunicação seguiu esse princípio, complementando a atual versão do software com a adição do suporte a comunicação com módulos embarcados.

Outra etapa com relação ao software de apoio foi a reformulação no visual, comparado com a primeira versão web, o visual ficou mais leve e intuitivo. Um exemplo da interface reformulada e com comunicação pode ser visualizada na Figura 21.

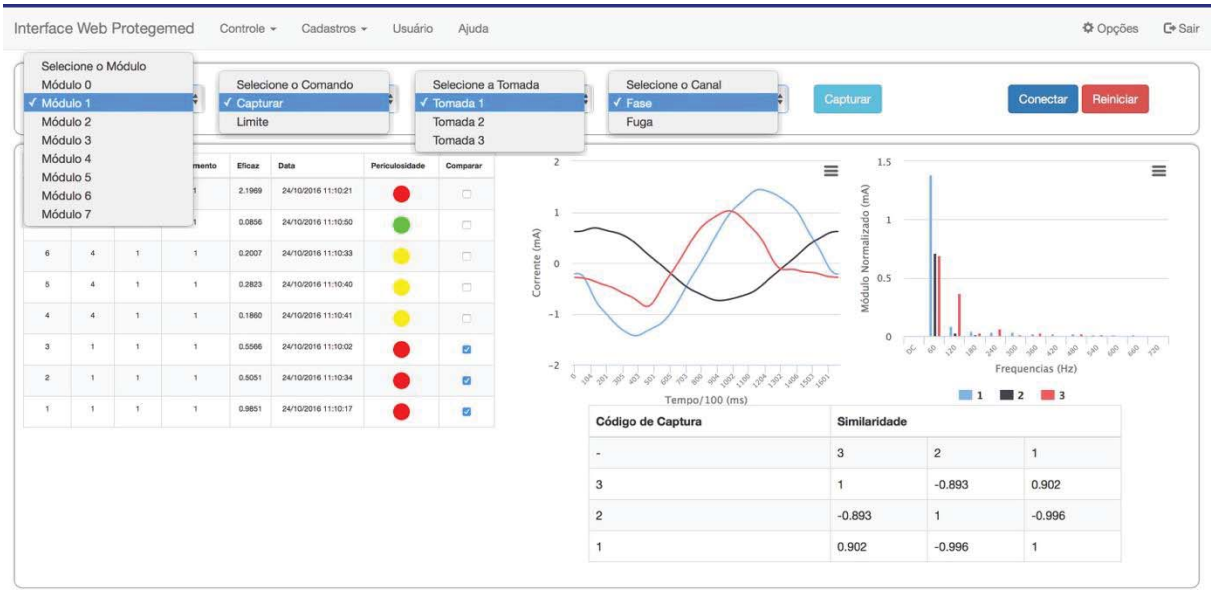


Figura 21 – Versão web reformulada e com comunicação bidirecional.

A ilustração acima apresenta a versão web com integração da comunicação com módulos. Conta com um menu superior para navegação entre páginas, como controle ou cadastros. Abaixo do menu seguem alguns *dropdowns*, que são ativados em sequência. O *dropdown* inicial exibe o módulo que se pretende realizar a comunicação, após pode ser escolhido o comando capturar ou enviar limites, no exemplo o comando definido foi o capturar. Após a seleção do comando, o próximo *dropdown* é para escolha da tomada que se deseja realizar a captura, compreendendo 3 opções por módulo. A última seleção é o canal, permitindo escolher entre fase e fuga. Ao final o botão capturar é exibido, permitindo que sejam obtidos os dados. Além do botão capturar, ainda é possível realizar a conexão com o servidor WebSocket e reiniciar o módulo escolhido.

No exemplo o comando para capturar dados em tempo real foi ilustrado, através deste é possível obter dados de determinado equipamento pelo software de apoio, utilizando para isso a comunicação bidirecional. Além de capturar dados, há possibilidade de encaminhar os limites por equipamento. A opção de limite comporta dados para o equipamento em operação e quando não está sendo utilizado, mas permanece consumindo corrente, período compreendido como *standby*. Os limites para equipamento em operação são a base para detecção do Protegemed, já presentes desde a primeira versão. Os limites de *standby* foram implementados por este trabalho, necessitando, nesse caso, alterações no banco de dados. Em conjunto, os

dados para as duas modalidades de limite somam quatro valores, dois para cada modo de operação (normal e de *standby*), um correspondente ao limite para o canal de fase e o outro para o canal de fuga de cada modo.

Além dos comandos implementados, a exibição dos dados é comportada pelo software de apoio. Abaixo dos *dropdowns* se encontra a tabela que é alimentada com a busca no banco de dados, apresentando os dados referentes a cada captura, como valor eficaz e a data da captura. Além de permitir que sejam comparadas entre si através do *checkbox* a direita da tabela. O qual ao ser selecionado apresenta visualmente a informação através de formas de onda e espectro de frequência, além de realizar o cálculo de periculosidade na tabela abaixo dos gráficos.

```
function enviarReset() {
    moduleIP = module.options[module.selectedIndex].getAttribute("ip");
    if (!conexaoAtiva) {
        alert("Conexão não estava ativa, favor enviar novamente a mensagem!!!");
    } else
    {
        websocket.send("#*reset*#" + moduleIP);
    }
}
```

Figura 22 – Função JavaScript para reiniciar um módulo.

Para que fosse possível manter a persistência da conexão WebSocket, foi necessário adicionar ao software web em PHP o controle da página através da linguagem JavaScript. Pois o PHP é executado no lado servidor, necessitando atualização da página a cada comando, encerrando, assim, a conexão persistente com o servidor WebSocket. Como o JavaScript é executado no navegador, permite manter a conexão aberta e receber atualização na página sem necessidade de a recarregar. Possibilitando que, após a primeira conexão, a página esteja apta a encaminhar mensagens sem necessitar nova conexão a cada envio, realizando, ainda, a atualização dos dados provenientes do banco de dados em tempo real.

Para possibilitar a atualização da página e o envio de mensagens foram utilizadas funções JavaScript para cada necessidade, um exemplo de função pode ser conferido na Figura 22. A função encaminha uma mensagem de reinicialização para o módulo selecionado, buscando o endereço IP do módulo por atributo e testando se a conexão com o servidor WebSocket está ativa. Caso o teste retorne negativo, realiza a conexão e informa que o servidor não estava conectado, caso contrário encaminha a mensagem para o módulo.

7. TESTES

A implementação dos modelos demonstrou que ambos foram capazes de encaminhar dados para os módulos embarcados. A etapa seguinte, para definir o modelo que atenda as necessidades do Protegemed, foi conduzir testes de estabilidade, de intervalo de tempo na troca de mensagens e de confiabilidade.

Para realizar a etapa de testes foi utilizado um computador equipado com um processador Intel Core i5 operando a 1.8 GHz, com 6 GB de memória DDR3 1600 MHz e 240 GB de armazenamento em flash, executando o sistema operacional Windows 10 Pro de 64 bits.

O primeiro teste para definir a forma de comunicação foi o envio de capturas para um módulo a cada cinco segundos durante oito horas. A expectativa era ao final do teste ambas as tecnologias realizarem 5.760 capturas (12 capturas/minuto x 60 minutos x 8 horas). No entanto, o teste com Telnet não atingiu o número esperado, pois reinicializava o módulo embarcado após uma sequência de, em média, 87 capturas. A suspeita, neste caso, é o consumo excessivo de memória do servidor Telnet embarcado, visto a limitação deste recurso no módulo.

Diante disso, foi constatada que a comunicação utilizando Telnet não foi capaz de concluir os testes iniciais de capturas, além de apresentar pontos negativos frente a comunicação com WebSocket, como a falta de suporte ao tráfego de dados *full duplex* e comunicação sem persistência. Dessa forma, foram descontinuados os testes com essa possibilidade de comunicação e o estudo foi direcionado para testes apenas com a comunicação utilizando o protocolo WebSocket.

Os testes com o protocolo WebSocket estão separados da seguinte forma: a seção seguinte apresenta a definição dos testes; as três seções na sequencia detalham cada tipo de teste e ilustram os resultados obtidos; por fim, a última seção elenca considerações sobre a etapa de testes.

7.1. DEFINIÇÃO

A busca com os testes foi por simular situações de uso cotidiano da comunicação, com diversos módulos operando e diferentes clientes interagindo com eles. A ideia principal é compreender o uso de módulos embarcados e navegadores web, trocando informações entre si com o auxílio da comunicação bidirecional usando o protocolo WebSocket.

Os testes utilizam componentes para apoio e se dividem em três grupos: o primeiro visa buscar a estabilidade da troca de informações entre módulos e clientes web; o segundo teste pretende definir o intervalo de tempo que uma mensagem leva para tramitar entre diferentes clientes e o servidor de WebSocket; o último teste, denominado de confiabilidade, pretende verificar se os dados obtidos pela comunicação bidirecional são válidos.

7.2. APOIO

Para condução dos testes serão utilizados componentes para apoiar a obtenção e comparação dos dados. Dentre eles, estão os métodos para repetição de funções e para comparação da similaridade, além da bancada de testes para obtenção dos dados.

Com o intuito de simular a ação repetitiva de usuários, a etapa de testes vai utilizar scripts desenvolvidos utilizando a linguagem de programação JavaScript, principalmente atrelados ao uso do método “setInterval” [52]. Com ele é possível executar funções em intervalos de tempo especificados, sendo expresso o valor da repetição em milissegundos.

Outro método para apoio na etapa de testes é o utilizado na comparação da similaridade entre formas de ondas. Desenvolvido em um trabalho final de graduação [11] e contribuindo para o cálculo de escalas de periculosidade em uma tese de doutorado [9], o método de Spearman Deslocado procura identificar numericamente a associação entre dois conjuntos de dados.

O método de Spearman retorna um coeficiente entre 1 e -1, sendo que valores próximos a 1 (negativo ou positivo) demonstram maior associação e valores próximos a zero menor associação. O sinal indica o sentido da associação, é positivo quando os conjuntos de dados se associam em uma mesma direção, ou seja, quando um aumentar o outro também irá aumentar, sendo negativo quando o oposto ocorrer, um aumentar e outro diminuir. O coeficiente retorna 1 (um) quando há associação perfeita entre os conjuntos e 0 (zero) na ausência de associação.

No trabalho de graduação foram propostos intervalos de valores para o retorno do método, visando definir escalas de similaridade para o coeficiente. São cinco escalas (mínima, baixa, média, alta e máxima). Na Tabela 2 é possível consultar as escalas.

Os valores propostos na tabela consideram o coeficiente em módulo, não diferenciando valores negativos de positivos. Sendo necessário, caso o método retorne valor negativo, considerar o valor absoluto do coeficiente.

Além dos métodos, a etapa de teste utilizou uma bancada de testes composta por um painel similar ao exposto na Figura 3. Neste painel estão os componentes físicos e lógicos para obtenção de dados do projeto Protegemed.

Tabela 2 – Escalas de similaridade.

Similaridade	De	Até
Máxima	1	1
Alta	0,9500	0,9999
Média	0,8500	0,9499
Baixa	0,5000	0,8499
Mínima	0,0001	0,4999

A bancada de testes deste trabalho conta com os seguintes componentes: um painel, que possuí dois módulos embarcados monitorando três tomadas cada; um módulo adicional sem ligação com tomadas; um computador equipado com o software de apoio e servidores. A Figura 23(a) mostra a visão frontal do painel que compõe o ambiente de testes e a Figura 23(b) o ambiente em operação, com visão da lâmpada led inserida no soquete a esquerda e do computador que monitora os módulos embarcados no painel ao centro.

(a) **(b)**

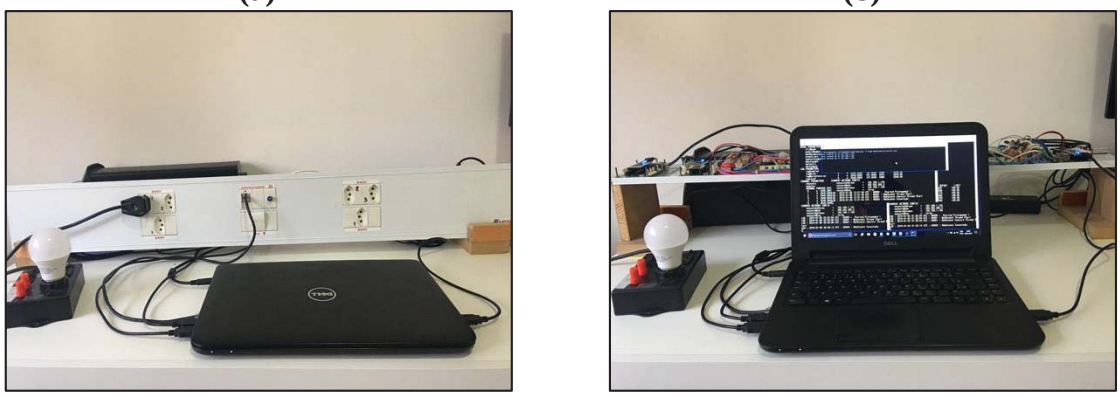


Figura 23 – Bancada de testes.

O computador atua como servidor e cliente WebSocket, como servidor recebe os dados da comunicação pela interface de rede cabeada, como cliente interage com o servidor local através de um navegador web conectado ao servidor. Além da comunicação, o computador possuí um banco de dados para armazenar as informações capturadas e o resultado dos testes conduzidos. O computador ainda contém o software de apoio através de um servidor web e monitora os módulos embarcadas através das portas USB.

Na bancada foram conduzidos os testes da comunicação bidirecional, o módulo adicional inserido nesta bancada visa incrementar a quantidade de clientes do servidor de WebSocket. O fato de não estar conectado a tomadas não ocasiona problemas, pois o processamento no módulo e o tamanho das mensagens é igual quando há equipamento nas tomadas ou na ausência destes, diferenciando apenas os valores obtidos. Para os testes de estabilidade e do intervalo de tempo este módulo foi utilizado, no teste de confiabilidade, que necessita informação de equipamento conectado as tomadas, o módulo adicional não foi usado.

7.3. TESTE DE ESTABILIDADE

O teste de estabilidade pretende definir se a implementação do servidor de WebSocket consegue transmitir mensagens entre clientes sem que ocorram perdas. O objetivo, neste caso, é que o servidor seja capaz de processar as mensagens de solicitação de captura provenientes do cliente web, as direcionando para os módulos embarcados e recebendo o retorno deste. Para o teste serão utilizados três módulos, cada módulo monitora três tomadas e cada tomada possui dois canais (fase e fuga).

7.3.1. Metodologia

O método utilizado será o envio de solicitações de captura para os módulos embarcados, por um cliente web, a cada cinco segundos durante oito horas. Para realizar o teste foi desenvolvido um script utilizando a linguagem JavaScript, que executa a função de captura em um intervalo de tempo personalizado, através do método “setInterval”, simulando a ação de um usuário. A função por sua vez realiza o envio de pedidos de capturas para os módulos, gerando aleatoriamente o módulo, o canal e a tomada a cada chamada da função de captura. Para isso, foram definidas variáveis contendo os endereços IP dos módulos e os canais de fase (p, do inglês *phase*) e de fuga (d, do inglês *differential*), como pode ser consultado na Figura 24.

```
var moduleRand = ["192.168.1.101", "192.168.1.102",
                  "192.168.1.103"];
var channelRand = ["p", "d"];
```

Figura 24 – Exemplo de código JavaScript.

A aleatoriedade foi obtida através da função “Math.random” (gera números pseudoaleatórios em ponto flutuante) e “Math.floor” (converte para inteiro), que em conjunto

geram números inteiros aleatórios para definir qual o índice das variáveis será utilizado e qual tomada a captura deve ser realizada.

Após a definição dos dados pela função de captura, estes são encaminhados através de uma mensagem para o servidor de WebSocket, que interpreta os comandos e os direciona com uma nova mensagem para o módulo informado. O módulo recebe a informação, realiza a captura na tomada e no canal definido aleatoriamente, encaminhando estes dados para o servidor de WebSocket, que os insere em um banco de dados.

O desejo é comparar a quantidade de mensagens que partiram do cliente, com a quantia inserida no banco de dados, após passarem pelo servidor de WebSocket. Para isso, o teste utiliza variáveis de controle em todos os componentes dos testes, visando armazenar o total de mensagens encaminhadas por cada elemento. Na Figura 25 é mostrada a imagem da tela onde os dados dos testes serão coletados.

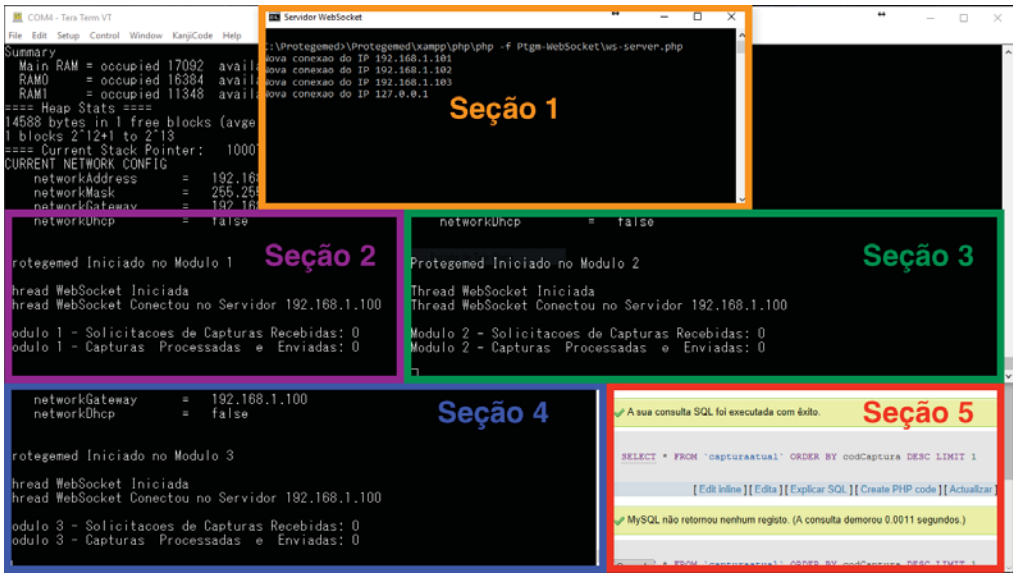


Figura 25 – Tela de coleta de dados inicial.

A figura está separada em cinco seções. Na Seção 1, centro superior da imagem, é exibido o servidor de WebSocket, ele receberá as solicitações de capturas e encaminhará aos módulos, armazenando o quantitativo de solicitações de capturas recebidas, direcionas a cada módulo e enviadas ao banco de dados. Nas seções 2, 3 e 4, centro e porção inferior esquerda da imagem, estão identificadas as três instâncias do aplicativo TeraTerm, que monitora individualmente os três módulos conectados via USB, nele é possível identificar as solicitações de capturas recebidas pelo módulo e as enviadas ao servidor de WebSocket após processamento. Na Seção 5, parte inferior direita da imagem, é ilustrada a consulta ao banco de dados, para exibir a quantidade de capturas inseridas, na imagem é possível consultar que o banco de dados está sem registros.

Tabela 3 – Programação do teste de estabilidade.

	Módulos Utilizados		
	1ª Etapa	**2ª Etapa**	**3ª Etapa**
1ª Execução	1	1 – 2	1 – 2 – 3
2ª Execução	2	2 – 3	1 – 2 – 3
3ª Execução	3	1 – 3	1 – 2 – 3

Na primeira etapa as mensagens serão direcionadas para apenas um dos módulos. Na segunda etapa dois módulos receberão as solicitações aleatoriamente. Concluindo a última etapa com três módulos operando e tramitando dados. Os módulos utilizados em cada etapa e em cada execução podem ser conferidos na Tabela 3. Nas três etapas as mensagens serão direcionadas para apenas um módulo a cada iteração, o objetivo é definir se o incremento de módulos conectados no servidor influenciará o tráfego de mensagens.

7.3.2. Resultado

Após a condução das três etapas, os dados foram armazenados em um banco de dados e coletadas imagens de tela de monitoramento contendo as variáveis em cada etapa de cada execução. As capturas de tela podem ser consultadas no Apêndice A e a imagem final dos testes pode ser conferida na Figura 26. Nas capturas de tela é possível identificar o quantitativo de capturas de cada variável em cada etapa.

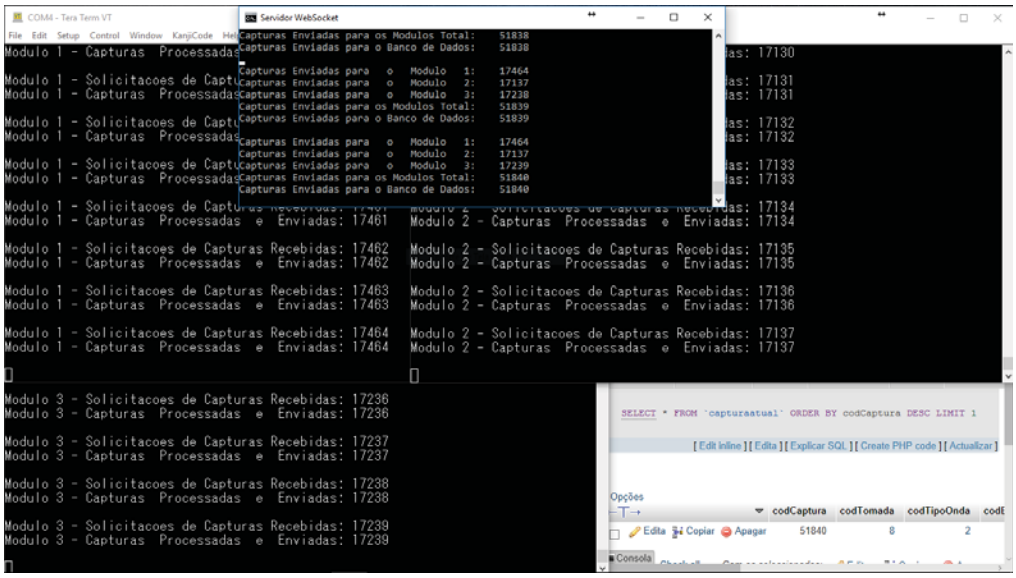


Figura 26 – Tela de coleta de dados final.

Os dados finais dos testes, após obtenção de média para cada etapa, podem ser consultados na Tabela 4, que está separada em quatro linhas, a primeira contendo a identificação de cada coluna e as demais compostas pelo resultado das etapas.

Tabela 4 – Teste de estabilidade.

Quantidade Módulos	Cliente Enviou	Módulo Recebeu	Módulo Enviou	BD Recebeu	Capturas Esperadas	Taxa de Sucesso
1	5.760	5.760	5.760	5.760	5.760	100%
2	5.760	5.760	5.760	5.760	5.760	100%
3	5.760	5.760	5.760	5.760	5.760	100%

Como a função captura executa a cada cinco segundos, a tabela acima possuí um número de 5.760 capturas esperadas (8 horas x 60 minutos x 60/5 segundos). Para mensurar os dados cada etapa executada no teste recebeu uma variável para contagem das mensagens trocadas, são elas:

- Cliente enviou – quantidade de solicitações de capturas encaminhadas pelo script para o servidor WebSocket;
- Módulo Recebeu – quantidade de mensagens que o módulo recebeu;
- Módulo Enviou – quantidade de dados que, após processar, o módulo encaminhou ao servidor de WebSocket;
- BD Recebeu – quantidade de capturas inseridas no banco de dados;
- Capturas esperadas – valor esperado de capturas;
- Taxa de Sucesso – porcentagem de sucesso contabilizando todas as etapas.

O resultado do teste, expresso na Tabela 4, mostra que o objetivo foi alcançado. Todas as mensagens trafegaram entre cliente web, servidor de WebSocket e módulos embarcados sem que houvessem perdas. Ainda foi possível identificar que o acréscimo de módulos conectados ao servidor, bem como a aleatoriedade entre módulos, tomadas e canal, não influenciaram a realização das capturas. Pois todas as solicitações de capturas conduzidas pelo cliente web foram recebidas no banco de dados.

A obtenção de cem por cento de sucesso nos testes com diferentes módulos, e nas três repetições destes para obter a média, demonstra estabilidade do servidor de WebSocket compatível com o esperado na condução de mensagens entre os clientes, assim como na interação com o banco de dados onde as capturas são armazenadas. Proporcionando comunicação bidirecional sem que hajam perdas na troca de mensagens.

7.4. TESTE DO INTERVALO DE TEMPO

O espaço temporal entre uma mensagem disparada por um cliente contendo um comando e o retorno, após execução, é importante para determinar usos para a comunicação bidirecional. No caso do Protegemed, este tempo pode definir, por exemplo, em uma captura instantânea, quantos ciclos completos serão processados desde a solicitação de captura pelo cliente web, até que o módulo a realize.

7.4.1. Metodologia

O teste será realizado através da medida do tempo em que uma mensagem tramita entre cliente e servidor. Para isso, foram utilizadas duas modalidades de clientes, módulos embarcados e navegadores web. A ideia é mensurar a quantia de milissegundos que uma determinada informação levará para se deslocar entre o servidor de WebSocket e os diferentes clientes conectados a este. Buscando identificar três intervalos de tempo: o maior, o menor e a média.

A metodologia utilizada será a mensagem partir do servidor de WebSocket com direção a um cliente contendo um comando de teste, iniciando o contador do tempo no momento em que for disparada no servidor. O cliente recebe a mensagem com o comando de teste e encaminha uma nova mensagem ao servidor informando sucesso no recebimento. O servidor ao receber o retorno do cliente interrompe o contador de tempo e realiza a métrica, o valor obtido é composto do envio ao cliente e do retorno da mensagem deste.

Um script, utilizando a linguagem JavaScript, foi desenvolvido para realizar o teste, este executa a função para iniciar o cálculo do tempo a cada 5 (cinco) segundos através da função “setInterval”. O script informa ao servidor, a cada execução da função, para encaminhar uma mensagem de teste ao cliente. Cada cliente foi testado durante oito horas e o teste foi repetido três vezes, realizando a média dos valores ao final.

Para obter o espaço de tempo entre a saída de uma mensagem do servidor e o retorno do cliente, após execução, foi utilizada a função PHP “microtime” [53], que retorna à quantidade de microssegundos desde 01 de janeiro de 1970. A função é executada duas vezes, quando a mensagem parte do servidor e quando retorna do cliente, o espaço de tempo é obtido com a diferença de microssegundos entre as duas chamadas da função. Os seguintes passos são executados no teste:

- Cliente Web – função em JavaScript dispara mensagens para o servidor de WebSocket a cada cinco segundos, informando o cliente destino para o teste;
- Servidor de WebSocket – recebe e interpreta a mensagem, direcionando um comando teste para o cliente informado, executando a função “microtime” e mantendo o valor em uma variável;
- Cliente do Teste – recebe e interpreta e mensagem, retornando com uma nova mensagem para o servidor de WebSocket informando sucesso no recebimento;
- Servidor de WebSocket – recebe a confirmação do cliente e executa novamente a função “microtime”, procedendo a comparação com execução anterior, encaminhando o intervalo de tempo obtido para ser armazenado em um banco de dados.

No teste com navegadores serão utilizados três tipos de clientes, os navegadores Google Chrome, Internet Explorer e Firefox, visando definir, além do intervalo de tempo na troca de mensagens, se o uso de diferentes navegadores influencia a métrica do espaço de tempo. Neste caso, ao invés de repetir o teste três vezes para cada navegador e contabilizar a média ao final, este estudo fez uso de três navegadores diferentes, executando um teste em cada navegador durante oito horas.

Para os módulos embarcados, objetivo principal deste estudo, além da métrica entre a saída da mensagem do servidor com um comando teste e o retorno do módulo, um teste adicional será conduzido. Este visa encontrar o espaço temporal para realizar uma captura completa, desde o momento em que se direciona o comando de captura para o módulo, através de um cliente web, até o servidor de WebSocket receber o retorno do módulo contendo os dados da captura. O teste é similar ao anterior dos navegadores, os passos executados são:

- Cliente Web – função em JavaScript dispara uma solicitação de captura a cada cinco segundos para o servidor de WebSocket, informando o módulo, a tomada e o canal;
- Servidor de WebSocket – recebe e interpreta a mensagem, direciona a solicitação para o módulo informado, executando a função “microtime” e armazenando o resultado em uma variável;
- Módulo Embarcado – recebe e interpreta a mensagem, conduz a captura dos dados e retorna para o servidor de WebSocket com uma nova mensagem contendo os valores obtidos;
- Servidor de WebSocket – recebe os dados do módulo e encaminha para o banco de dados, executando a função “microtime” e comparando com o valor armazenado anteriormente.

Após os passos acima é possível encontrar o intervalo de tempo entre a solicitação de captura no cliente web e o recebimento dos dados capturados no servidor de WebSocket, realizando, para isso, a comparação entre as execuções da função “microtime”.

7.4.2. Resultado

A métrica do intervalo de tempo entre solicitações de diferentes tipos de clientes, para o servidor de WebSocket, pretende definir a capacidade deste em diversos cenários, como na interação com módulos embarcados e navegadores web. Os resultados obtidos nos testes podem ser consultados na Tabela 5, onde são expostos os valores mínimo, médio e máximo para cada teste. Os dez menores, os dez médios e os dez maiores valores de cada teste podem ser consultados no Apêndice B, onde o tempo é expresso em milissegundos.

Tabela 5 – Espaço de tempo (ms).

Cliente	Mínima	Média	Máxima
Chrome	1.0350	3.0074	11.2641
Firefox	1.2531	2.5271	15.0042
Internet Explorer	5.7821	6.4300	11.2870
Módulo - Teste	31.6162	42.9227	60.5922
Módulo - Captura	208.5400	229.8362	247.7620

Os dados ilustrados na tabela demonstram que, utilizando o protocolo WebSocket, o intervalo de tempo sofreu pouca variação entre o valor mínimo e máximo em todos os testes. Constatando regularidade na troca de mensagens, pois em nenhum teste foi observado pico de valor (mínimo ou máximo) distante dos demais.

O teste com navegadores mostrou que não há alteração significativa na escolha do cliente web para uso com a comunicação bidirecional, pois os três navegadores indicaram variação entre 1 (um) e quinze milissegundos na troca de mensagens. Definindo o Internet Explorer como o mais regular entre valor mínimo e máximo, o Chrome com o menor valor mínimo único e o Firefox com o maior valor único, porém com a menor média entre os três. O teste apresentou, ainda, intervalos de tempo, em média, aproximadamente onze vezes menores comparados aos obtidos nos testes com módulos. Esta diferença de valor pode estar ligada com o hardware utilizado pelos clientes, no caso dos módulos embarcados a unidade processadora, por exemplo, opera a 96 MHz, frente aos 1800 MHz do computador no qual o teste dos navegadores foi conduzido.

Para os módulos embarcados, os valores obtidos evidenciam que apenas o processo de troca de mensagens consome em torno de 43 ms, já a condução de uma captura completa leva aproximadamente 214 ms. Neste caso, se for subtraído da captura o espaço de tempo da troca de mensagens, pode ser encontrado o intervalo que o módulo utiliza para capturar os dados internamente, sendo o valor médio de 171 ms para proceder a captura, acrescido de, em média, 43 ms para receber o comando e encaminhar os dados. O valor menor obtido no processo de troca de mensagens frente a captura completa pode estar ligado com o uso de *Threads*, enquanto no comando teste apenas a *Thread* de WebSocket está envolvida, para conduzir uma captura instantânea a *Thread* de WebSocket recebe e direciona para a *Thread* de Captura o comando, que, além de realizar a captura instantânea solicitada, permanece averiguando a rede elétrica a cada ciclo, consumindo recursos do módulo embarcado.

Para o Protegemed é importante a comparação dos valores encontrados com os ciclos da rede elétrica brasileira, que ocorrem a cada 16,67 ms. Neste sentido, para tramitar uma mensagem a comunicação com uso do protocolo WebSocket consome de dois a quatro ciclos. Caso a mensagem contenha o comando de captura, o módulo embarcado levará, aproximadamente, dez ciclos adicionais para proceder com a captura. Cabe ressaltar que, apesar de uma solicitação de captura instantânea levar alguns ciclos para apresentar os dados, a corrente elétrica não sofre alterações a todo instante. Sendo os valores para as solicitações de capturas instantâneas obtidos em no máximo 15 ciclos, frente aos sessenta que ocorrem a cada segundo na rede elétrica.

Como os dados obtidos com os testes em navegadores apresentaram intervalos de tempo menores do que 1 (um) ciclo no máximo, pode-se observar que a comunicação utilizando o protocolo WebSocket permite o tráfego de dados em espaço temporal menor do que os obtidos nos testes com a plataforma embarcada MBED. Caracterizando esta como fator de limite na obtenção da métrica temporal da comunicação com os módulos. No entanto, apesar do limite da plataforma, os dados ainda são obtidos em uma fração de segundo. Demonstrando que o uso deste modelo de comunicação pode ser aplicado ao projeto.

7.5. TESTE DE CONFIABILIDADE

Com a obtenção da estabilidade da conexão e do intervalo de tempo no tráfego das mensagens, o último teste objetiva identificar se as informações capturadas dos equipamentos, que estão ligados aos módulos embarcados, são confiáveis.

Para determinar se a informação obtida possui confiabilidade, o teste irá capturar dados de fontes com pouca variação elétrica entre ciclos, possibilitando a comparação entre os valores capturados. Neste caso, a escolha foi por utilizar lâmpadas como origem dos dados, definição motivada pelo fato das lâmpadas possuírem um padrão de comportamento elétrico regular e conhecido pelo projeto.

O teste de confiabilidade será separado em três etapas, na primeira as capturas serão obtidas de uma lâmpada incandescente de 60 Watts, na segunda de uma lâmpada led de 11 Watts e na terceira nada estará conectado ao soquete de lâmpadas. Ao final cada captura será comparada com um padrão de comportamento elétrico para cada teste. O objetivo é que as capturas provenientes das lâmpadas apresentem comportamento compatível com o padrão, já o teste sem lâmpada conectada ao soquete é esperado que este padrão não seja obtido, pois não haverá um padrão de consumo no canal de fase, apenas capturas de amostras de sinais de valor muito pequenos, próximos de zero, resultado de ruídos produzidos e amplificados pelo próprio circuito eletrônico do Protegemed. Este ruído não aparece nos demais casos, pois seu valor é muito pequeno comparado com os valores de corrente das duas lâmpadas ou de outros equipamentos.

7.5.1. Metodologia

Para realizar o teste uma função JavaScript irá disparar solicitação de captura instantânea para determinado módulo, com auxílio do método “setInterval” para repetir as solicitações em intervalo de tempo personalizado.

A lâmpada estará conectada em uma das três tomadas do módulo, com o auxílio de uma caixa contendo um soquete para lâmpadas em uma das extremidades e na outra uma tomada compatível com a do painel da bancada de testes. Para obter os dados das lâmpadas estas permanecerão ligadas, gerando consumo elétrico no canal de fase.

Com o conhecimento da tomada em que as lâmpadas estarão conectadas, a função para capturas irá encaminhar solicitações direcionadas a esta tomada e para o canal de fase. Armazenando os dados obtidos em um banco de dados para comparação posterior. As solicitações ocorrem com periodicidade de cinco segundos durante oito horas, sendo repetido três vezes o teste para cada etapa e gerado a média ao final.

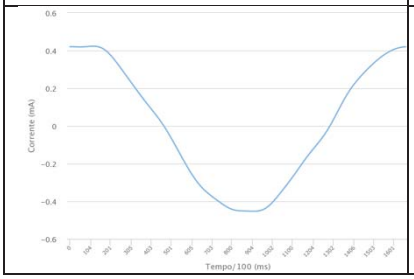
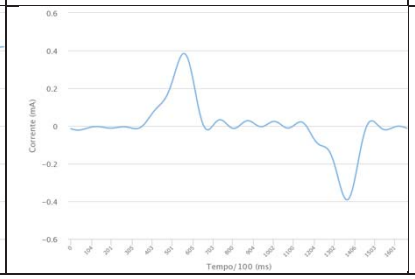
Após a coleta dos dados das lâmpadas incandescente e led e sem lâmpada conectada ao soquete, o próximo passo foi definir a confiabilidade destas informações. Para isso, este

estudo fará uso do método de Spearman Deslocado, que possibilita a comparação da similaridade entre formas de ondas elétricas.

Como o método realiza a comparação entre dois conjuntos de dados, pode ser definida uma captura como confiável realizando a comparação dos dados dela com um conjunto de dados tido como padrão para aquele equipamento. A ideia é realizar a comparação individual das capturas e obter ao final uma escala de similaridade Alta ou Máxima para as lâmpadas, ou seja, o método deve retornar um coeficiente maior ou igual a 0,95. Para o teste sem conexão a similaridade esperada é média ou menor, resultado do método deve ser inferior a 0,95.

O conjunto de dados padrão é obtido com a realização de 100 capturas, com intervalo de cinco segundos entre cada uma, para as três etapas do teste. As capturas são comparadas individualmente com as demais (100 comparações no total), daquela etapa, através do método de Spearman Deslocado, o coeficiente em cada iteração é armazenado e incrementado. Ao final é comparado o somatório de cada captura com as demais e definida como padrão a que obtiver o maior valor.

Tabela 6 – Formas de onda padrão.

Incandescente	Led	Sem Conexão
		

As formas de onda padrão, resultantes do conjunto de dados das três etapas, podem ser conferidas na Tabela 6. Ainda é possível identificar a diferença visual entre as FO das lâmpadas, bem como a corrente em relação ao tempo. Na FO sem conexão a corrente é próxima a zero, pois não há consumo neste caso. Os trinta valores mais significativos, para cada etapa da definição do conjunto de dados padrão, podem ser consultados no Apêndice C, separados entre os dez maiores, os dez menores e os dez somatórios médios.

7.5.2. Resultado

O quantitativo de capturas, baseado nas escalas de similaridade, após a execução do método de Spearman Deslocado entre cada captura e o conjunto de dados padrão para aquela

etapa, pode ser conferido na Tabela 7. A primeira linha expõe o conteúdo de cada coluna, as três linhas seguintes separam os componentes do teste. A coluna inicial demonstra a origem dos dados, a segunda coluna apresenta a quantidade de capturas realizadas, as próximas três colunas ilustram a quantidade de capturas dentro de cada escala de similaridade.

Tabela 7 – Análise da similaridade.

Origem	**Quantidade de Capturas**	**Similaridade**		
		Média ou Menor	**Alta**	**Máxima**
Incandescente	5.760	0	5.760	0
Led	5.760	0	5.760	0
Sem Conexão	5.760	5.760	0	0

Nos testes com as lâmpadas incandescente e led, onde a similaridade esperada era alta ou maior, o resultado apresentou todas as capturas, de ambos os testes, dentro desta escala. Já no teste sem conexão, onde havia expectativa de similaridade média ou menor, os dados confirmaram que a inexistência de um padrão de consumo gera escala de similaridade menor, pois todas as capturas apresentam escala média ou menor.

Nas três etapas não foi constatada nenhuma captura com similaridade máxima, fato que pode estar ligado com a oscilação da rede elétrica, pois para obter similaridade perfeita o conjunto de dados comparado necessita ser idêntico ao padrão. Os principais resultados obtidos em cada etapa podem ser consultados no Apêndice D, onde estão separados pelo coeficiente de similaridade e separados em três partes: os dez com maior, os dez centrais e os dez menores coeficientes.

Os dados obtidos demonstram que a implementação usando o protocolo WebSocket para comunicação bidirecional fornece a confiabilidade esperada, pois os valores desejados nos três testes foram encontrados.

7.6. CONSIDERAÇÕES SOBRE OS TESTES

A etapa de testes foi um dos principais pontos dessa pesquisa, após definir a possibilidade de comunicação bidirecional e realizar o levantamento teórico, definir modelos de solução e os implementar. O foco desse estudo foi por testar as formas de comunicação.

Os módulos embarcados permaneceram operando initerruptamente por aproximadamente 30 dias. Nesse tempo, apenas um contratempo foi encontrado, na primeira semana de testes um dos três módulos parou de responder. Consultando o seu sistema de

arquivos, onde fica localizado o seu *firmware*, não foi encontrado nenhum arquivo. A suspeita, nesse caso, foi que o sistema de arquivos do módulo tenha corrompido. Pois após adicionar um novo arquivo de *firmware* no módulo, esse não apresentou mais problemas no decorrer dos testes.

O indício que o sistema de arquivos tenha corrompido se fortalece pois em etapas anteriores do estudo, principalmente na etapa de implementação, em alguns momentos fatos similares ocorreram, onde aleatoriamente o *firmware* desaparecia do sistema de arquivos ou o arquivo de configuração do módulo, que fica no sistema de arquivos, não era encontrado.

O uso da plataforma MBED, além de apresentar problemas ao longo deste estudo, pode ser um fator de limite para a performance do projeto Protegemed. Este fato é identificado, principalmente, com a diferença significativa do intervalo de tempo na troca de mensagens comparada com os navegadores, bem como no espaço de tempo para realizar uma captura completa. Ainda, outro fator limitante do MBED é a quantidade de memória RAM disponível, dificultando a ampliação de funcionalidade para o Protegemed. Diante disso, este estudo sugere a análise da escolha da plataforma MBED como módulo do Protegemed, visando buscar alternativas que forneçam maior confiança, com incremento no processamento e na quantidade de memória disponível.

8. CONCLUSÃO

O presente trabalho buscou o desenvolvimento de uma forma de comunicação bidirecional para ser utilizada na plataforma embarcada usada pelo projeto Protegemed. A comunicação entre a plataforma MBED, que captura os dados, e o servidor, que contém um banco de dados, era conduzida por apenas uma via de tráfego das informações, do MBED para o servidor.

Diante disso, esta pesquisa buscou na área de redes, mais especificamente nos protocolos de comunicação, possíveis soluções para a comunicação bidirecional. Foram encontrados três protocolos que poderiam suprir a demanda: o primeiro foi o protocolo HTTP, atuando como servidor embarcado e fornecendo uma via adicional oposta a atual, podendo ser utilizado para receber os dados no MBED e mantendo a atual forma de comunicação para envio dos dados; o segundo protocolo, Telnet, através de um servidor embarcado, oferece duas vias de comunicação, nesse caso o MBED atua como servidor e recebe os dados a partir da conexão externa de clientes nesse servidor, sendo capaz de trafegar dados em ambas as direções e suprimir o envio de dados unidirecional; o terceiro permite comunicação bidirecional full-duplex e persistente, o protocolo WebSocket se comunica com o MBED através de um servidor externo dedicado ao protocolo, o módulo embarcado se conecta nesse servidor como cliente, passando a trocar informações nos dois sentidos, substituindo a via em apenas um sentido, até então utilizada.

Após abordagem teórica das possíveis soluções, constatando que dentre as soluções pesquisadas duas fornecem tráfego de dados em dois sentidos sobre o mesmo protocolo, foi definido que a via atual de comunicação seria substituída ao invés de incrementada, excluindo, assim, o protocolo HTTP como possível solução.

Com dois protocolos restantes, a próxima etapa foi de implementação das possíveis soluções. Para isso, dois modelos de solução foram propostos e implementados, um usando o protocolo Telnet e outro o protocolo WebSocket. Após implementação foi constatado que ambas as soluções permitiam contato com o MBED, fornecendo comunicação bidirecional para o Protegemed. Para definir qual modelo seria utilizado, o próximo passo foi conduzir testes simulando situações de uso cotidiano da comunicação. No entanto, no primeiro teste em que solicitações de capturas foram direcionadas aos componentes embarcados, o protocolo Telnet não demonstrou estabilidade para concluir o teste, reiniciando o MBED após uma sequência baixa de capturas, não sendo encontrada a causa para esta reinicialização. O que culminou com

a exclusão preliminar desse protocolo e os testes foram direcionados apenas para o protocolo WebSocket.

A etapa de testes contou com a simulação de ações rotineiras para a comunicação, visando compreender três pontos cruciais para escolha do protocolo WebSocket como substituto para o Protegemed: a estabilidade da conexão, o intervalo de tempo na troca de mensagens e a confiabilidade das informações capturadas.

No teste de estabilidade, onde a cada cinco segundos uma solicitação de captura foi direcionada ao MBED durante oito horas, o modelo de comunicação com uso do protocolo WebSocket conseguiu atingir 100% de sucesso. Contabilizando as mensagens provenientes do cliente web com solicitações de capturas, as mensagens que o módulo embarcado recebeu após passar pelo servidor de WebSocket, as mensagens que o módulo enviou depois de processar a captura e a quantidade de capturas inseridas no banco de dados.

Na definição do espaço de tempo entre a troca de mensagens do servidor de WebSocket com diferentes clientes, no qual uma mensagem partia do servidor de WebSocket com direção ao cliente e esse retornava com uma nova mensagem informando o recebimento, a comunicação com uso do protocolo WebSocket apresentou resultados que a credenciam para ser utilizada no projeto Protegemed. Fornecendo intervalos de tempo, na troca de mensagens entre cliente e servidor, entre um e quinze milissegundos para os testes com navegadores, entre 31 e 61 ms para mensagem de testes com módulos embarcados e entre 209 e 248 ms para condução de captura instantânea.

No teste de confiabilidade, com a obtenção de dados de lâmpada incandescente e de led, bem como sem lâmpada conectada ao soquete, o método de Spearman Deslocado, utilizado para definir a confiança dos dados através da comparação da similaridade de cada conjunto de dados com um padrão, apresentou todos os valores dentro do esperado. Comprovando que os dados transmitidos via comunicação bidirecional com uso do protocolo WebSocket são válidos.

Os resultados demonstram que o uso do modelo de comunicação bidirecional, baseado no protocolo WebSocket, é capaz de substituir a atual forma de envio dos dados em apenas uma direção. Ele fornece estabilidade, intervalo de tempo no tráfego de mensagens compatível com o projeto e confiabilidade na obtenção das informações.

Portanto, a conclusão deste trabalho é que o protocolo WebSocket como forma de comunicação principal para o projeto Protegemed, fornecendo, sobre o mesmo protocolo, comunicação bidirecional full-duplex e persistente é capaz de resolver o problema da atualização dos limites de corrente contidos nos arquivos de configuração do MBED.

Por fim, o objetivo inicial foi alcançado, o projeto Protegemed foi capacitado com comunicação em duas vias. Os pesquisadores e utilizadores do Protegemed podem, agrora, interagir com os módulos embarcados através de clientes web, podendo realizar capturas instantâneas, reinicializar os módulos, encaminhar parâmetros de configurações, dentre outras possibilidades viáveis com o envio de dados para os módulos.

8.1. CONTRIBUIÇÃO PARA O PROJETO PROTEGEMED

Durante o desenvolvimento do estudo, algumas contribuições secundárias a este trabalho foram inseridas no projeto Protegemed. Uma das melhorias foi o uso de *templates* para as páginas web do projeto. Estas contavam com a inserção descentralizada de cabeçalhos, rodapés e informações comuns a todas as páginas, como *tags* dentro de cada "view". Sendo inseridas em todos os arquivos o mesmo conteúdo, dificultando, por exemplo, a atualização de versão de bibliotecas, já que todo arquivo realizava a chamada individual de cada biblioteca. Além de poluir as páginas "views" com excesso de código. Estas informações foram movidas para uma área de *template* contendo uma página de cabeçalho, uma de rodapé e outra de menu, sendo inseridas na "view" pelo controller do modelo MVC (*Model*, *View* e *Controller*). Um exemplo da diferença de uma "view" com uso de *template* e sem pode ser consultado na Figura 27.

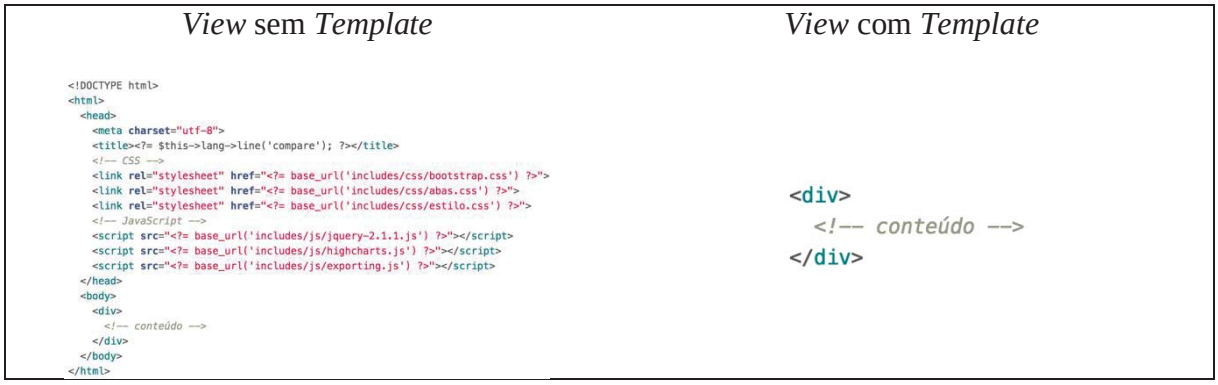


Figura 27 – Página exemplo do uso de *Template*.

Como pode ser visualizado na figura acima, o modelo com *Template* ilustra a "view" apenas com o conteúdo visual da página, ficando toda codificação e tags html, por exemplo, centralizados nas páginas do *Template*.

Em conjunto com o uso de *Templates*, outra contribuição para o projeto foi a reformulação visual. A diferença pode ser consultada entre a Figura 5 e a Figura 21. As alterações visuais realizadas foram no cabeçalho, no rodapé e no menu. Além de atualização da biblioteca jQuery e dos frameworks CodeIgniter e BootStrap para as últimas versões disponibilizadas, conferindo maior confiabilidade, correções de bugs e visual moderno.

Além de contribuições visuais e aprimoramento na gestão das páginas, foi implementado o método de Spearman para comparar a similaridade entre ondas elétricas, como pode ser consultado na tabela da Figura 21.

8.2. TRABALHOS FUTUROS

Com o desenvolver do trabalho, algumas constatações para futuras colaborações ao projeto foram encontradas. Uma delas, pode ser a implementação de uma página para monitorar o estado dos módulos conectados, realizando a verificação de atividade através de mensagens de controle em um determinado espaço de tempo. Possibilitando, ainda, verificar equipamentos conectados a cada módulo, além de informações gerais do módulo, como logs, limites, capacidade de processamento, quantidade de memória usada, entre outros.

Outro aprimoramento é com relação a captura de dados dos módulos, atualmente atrelada a tomada. Podendo ser desenvolvida a atualização do equipamento ao ser ligado em determinada tomada, encaminhando ao banco de dados esta informação. Facilitando a aquisição de dados, pois não será mais necessário o conhecimento da tomada e do módulo em que o equipamento está ligado, tarefa esta automatizada via codificação. Ainda com relação aos equipamentos, assim como os módulos, pode ser implementado o controle de equipamentos ativos.

A parte de monitoramento pode ser acrescida de interface gráfica para controle dos equipamentos em tempo real, ilustrando dados provenientes deste de forma gráfica, com uso de formas de onda, por exemplo. Esta interface pode conter valores dos canais de fase e fuga e realizando a atualização dos dados através da obtenção contínua de dados via comunicação bidirecional, exibindo de forma gráfica no software de apoio.

O quesito atualização de *firmware* dos módulos embarcados pode ser desenvolvido através do protocolo WebSocket, sendo implementada uma página, por exemplo, em que o arquivo do novo *firmware* seja selecionado e definido a qual módulo será direcionado.

REFERÊNCIAS

[1] L. E. S. Spalding, W. P. Carpes, and N. J. Batistela, “A Method to Detect the Microshock Risk During a Surgical Procedure,” *IEEE Trans. Instrum. Meas.*, vol. 58, no. 7, pp. 2335–2342, 2009.

[2] R. Fowler, *Fundamentos de Eletricidade: Corrente Continua e Magnetismo*, 7ª Edição. 2012.

[3] P. Cutler, *Análise de Circuito CA*. McGraw-Hill do Brasil Ltda, 1976.

[4] M. Gussow, *Eletricidade Básica*, 2ª Edição. Pearson Education do Brasil Ltda, 1985.

[5] L. F. Netto, “Sinais Elétricos e suas Formas de Onda,” 2016. [Online]. Available: http://www.feiradeciencias.com.br/sala15/15_07.asp. [Accessed: 10-Aug-2016].

[6] G. Kindermann, *Choque elétrico*, 3 edição. Florianópolis, 2005.

[7] J. G. Webster, *Medical Instrumentation: Application and Design*. 1998.

[8] L. E. S. Spalding, “Detecção de risco de microchoque através da corrente diferencial em equipamentos eletromédicos”. Tese de doutorado, Grucad, UFSC. 2009.

[9] M. T. Rebonatto, L. E. S. Spalding, F. P. Hessel, and L. A. Amaral, “Protegemed2: an extended platform based on RFID to identify EME and improve the detection of microshocks.,” *Med. Biol. Eng. Comput.*, vol. 51, no. 6, pp. 719–727, Jun. 2013.

[10] M. T. Rebonatto, “Métodos para análise de correntes elétricas de equipamentos eletromédicos em procedimentos cirúrgicos e detecção de periculosidade aos pacientes.” Tese de doutorado, Facin, PUC-RS. 2015.

[11] M. A. Schmitz and M. T. Rebonatto, “Análise de Métodos para Comparação da Similaridade Entre Ondas Elétricas,” 2014.

[12] “MBED LPC 1768,” 2015. [Online]. Available: https://developer.mbed.org/platforms/mbed-LPC1768/. [Accessed: 09-Oct-2016].

[13] D. Ritchie and B. Kernighan, *The C programming language*, vol. 78. 1988.

[14] E. S. Trentin and M. T. Rebonatto, “Interface Web Para Monitoramento de Salas de Cirurgia com uso do Sistema Protegemed,” 2011.

[15] M. Perego and M. T. Rebonatto, “Interface Web Para Monitoramento de Salas de Cirurgia com uso do Sistema Protegemed,” 2014.

[16] “HTML, The Web’s Core Language,” 2016. [Online]. Available: http://www.w3.org/html/. [Accessed: 15-Oct-2016].

[17] “Hypertext Transfer Protocol -- HTTP/1.1,” 1999. [Online]. Available: http://www.rfc-base.org/txt/rfc-2616.txt. [Accessed: 12-Sep-2016].

[18] “PHP,” 2016. [Online]. Available: https://secure.php.net. [Accessed: 07-Dec-2016].

[19] “CodeIgniter,” 2016. [Online]. Available: http://www.codeigniter.com. [Accessed: 18-Oct-2016].

[20] “JavaScript,” 2016. [Online]. Available: http://www.w3schools.com/js/. [Accessed: 07-Dec-2016].

[21] “jQuery,” 2016. [Online]. Available: http://jquery.com. [Accessed: 07-Dec-2016].

[22] “MySQL,” 2016. [Online]. Available: http://www.mysql.com. [Accessed: 07-Dec-2016].

[23] “CSS,” 2016. [Online]. Available: http://www.w3schools.com/css/. [Accessed: 07-Dec-2016].

[24] “Ajax,” 2016. [Online]. Available: http://www.w3schools.com/xml/ajax_intro.asp. [Accessed: 07-Dec-2016].

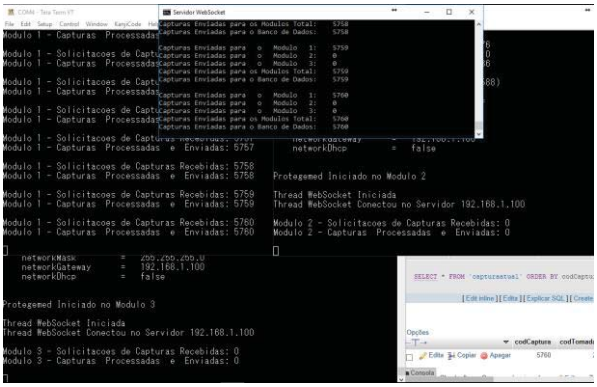
[25] “HighCharts,” 2016. .

[26] “Bootstrap,” 2016. [Online]. Available: http://getbootstrap.com. [Accessed: 17-Oct-

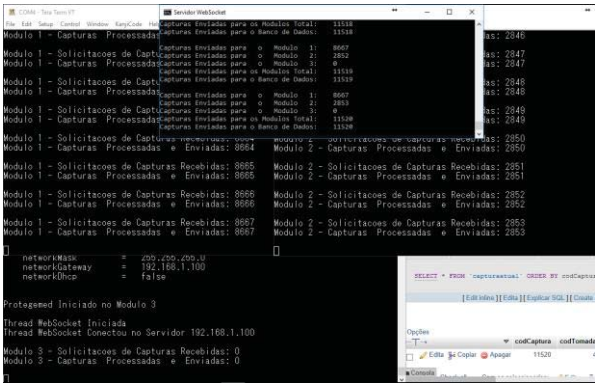
2016].
[27] W. Wolf, *Computer as Components*. 2008.
[28] P. Marwedel, *Embedded System Design*. 2006.
[29] R. Toulson and T. Wilmshurst, *Fast and Effective Embedded Systems Design*. Elsevier, 2012.
[30] D. W. Lewis, *Fundamentals of Embedded Software with the ARM Cortex-M3*, Second Edi. Pearson Education do Brasil Ltda, 2013.
[31] "Plataforma MBED LPC1768," 2016. .
[32] B. A. Forouzan, *Comunicação de Dados e Redes de Computadores*, 4ª Edição. 2010.
[33] A. S. Tanenbaum and D. Wetherall, *Redes de Computadores*, 5ª Edição. 2011.
[34] J. Kurose and K. Ross, *Redes de Computadores e a Internet: uma abordagem top-down*, 6ª Edição. 2013.
[35] T. A. P. Fucilini and M. T. Rebonatto, "Comparação entre modelos de comunicação com o Protegemed," 2010.
[36] "RFC 15," 1969. [Online]. Available: https://tools.ietf.org/html/rfc15. [Accessed: 20-Sep-2016].
[37] "RFC 137," 1971. [Online]. Available: https://tools.ietf.org/html/rfc137. [Accessed: 20-Sep-2016].
[38] "RFC 854," 1983. [Online]. Available: https://tools.ietf.org/html/rfc854. [Accessed: 15-Sep-2016].
[39] T. D. R. Varela and S. Loh, "Implementação e Análise da Utilização de WebSockets em Sistemas Computacionais," 2010.
[40] "RFC 6455," 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455. [Accessed: 20-Sep-2016].
[41] H. M. Deitel and P. J. Deitel, "Como Programar C++," 5ª Edição, Ed. 2006.
[42] M. S. Silva, *HTML 5. A Linguagem De Marcaçao Que Revolucionou A Web*. 2011.
[43] D. Eis and E. Ferreira, *HTML5 e CSS3*. 2011.
[44] W. Soares, *PHP 5 - Conceitos, Programação e Integração com Banco de Dados*, Sétima Edi. 2013.
[45] F. D. N. Grillo and R. P. de M. Fortes, *Aprendendo JavaScript*. 2008.
[46] "jQuery," 2016. [Online]. Available: http://jquery.com. [Accessed: 17-Oct-2016].
[47] J. Niederauer, "Web Interativa com Ajax e PHP," 2ª Edição., 2013.
[48] A. Milani, *MySQL - Guia do Programador*. 2007.
[49] "Ratchet," 2016. [Online]. Available: http://socketo.me. [Accessed: 23-Sep-2016].
[50] "MAMP," 2016. [Online]. Available: https://www.mamp.info/. [Accessed: 17-Oct-2016].
[51] "mbed Compiler," 2016. [Online]. Available: https://developer.mbed.org/handbook/mbed-Compiler. [Accessed: 13-Oct-2016].
[52] "JavaScript Timing Events," 2016. [Online]. Available: http://www.w3schools.com/js/js_timing.asp. [Accessed: 24-Dec-2016].
[53] "Microtime," 2016. [Online]. Available: http://www.w3schools.com/php/func_date_microtime.asp. [Accessed: 21-Dec-2016].

Apêndice A

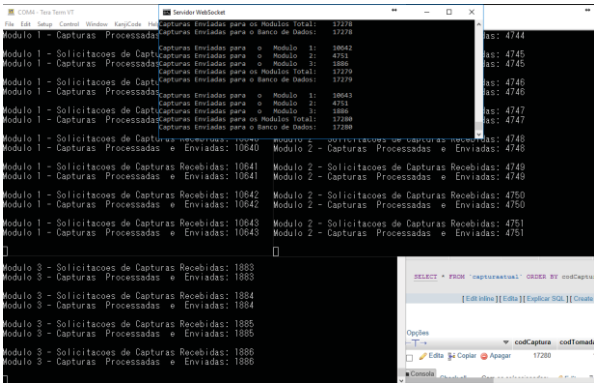
Primeiro Teste - 1 Módulo



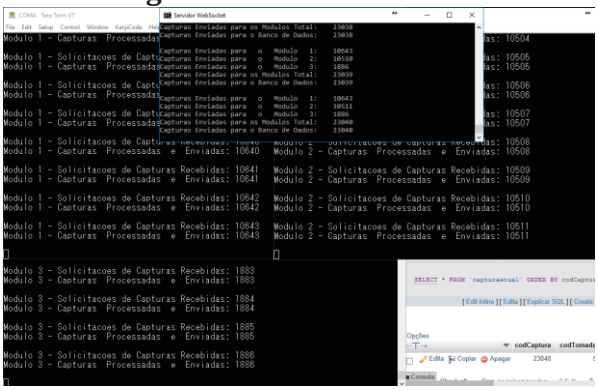
Primeiro Teste - 2 Módulos



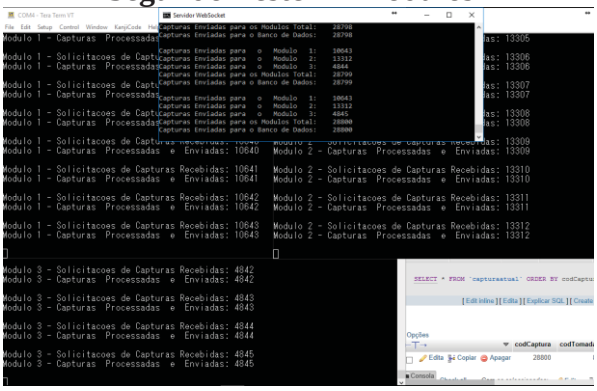
Primeiro Teste - 3 Módulos



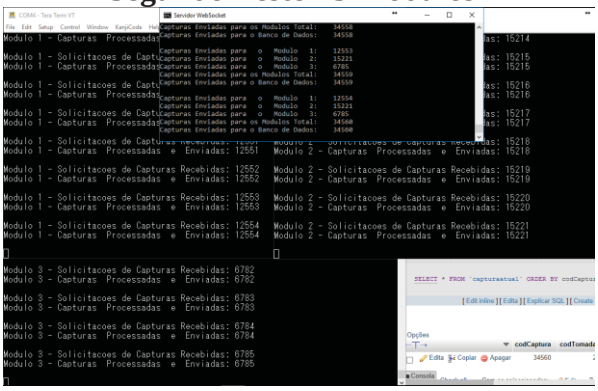
Segundo Teste - 1 Módulo



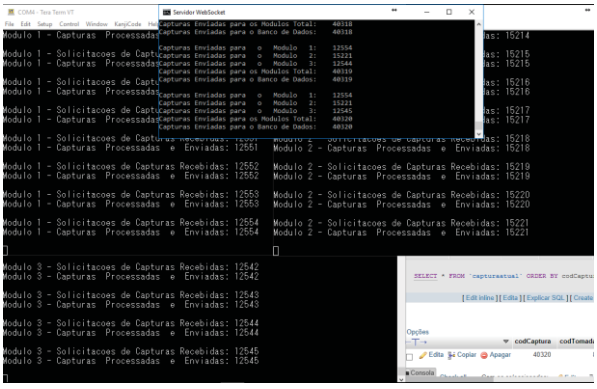
Segundo Teste - 2 Módulos



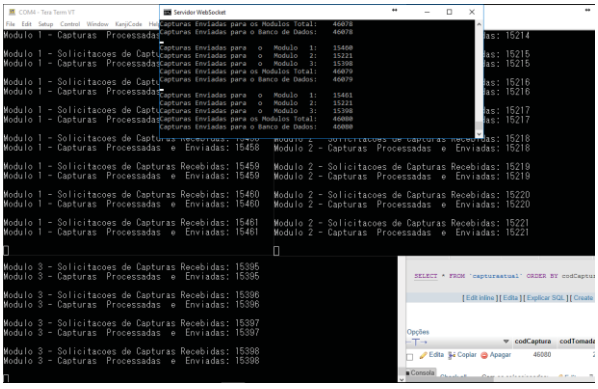
Segundo Teste - 3 Módulos



Terceiro Teste - 1 Módulo



Terceiro Teste - 2 Módulos



Apêndice B

Tempo Navegadores (em milissegundos)								
Chrome			Firefox			Internet Explorer		
Mínimo	Médio	Máximo	Mínimo	Médio	Máximo	Mínimo	Médio	Máximo
1.0350	3.0720	5.4960	1.2531	1.8361	10.3691	5.7821	6.3798	9.4311
1.2400	3.0720	5.6040	1.2839	1.8361	10.4940	5.7940	6.3798	9.4690
1.2500	3.0720	5.8610	1.2960	1.8361	10.6730	5.8072	6.3798	9.5441
1.2739	3.0720	5.8680	1.2982	1.8361	10.8020	5.8100	6.3798	9.5510
1.2770	3.0720	7.5421	1.3030	1.8368	11.1351	5.8150	6.3798	9.5599
1.3011	3.0720	8.6260	1.3058	1.8370	11.7211	5.8169	6.3801	9.6810
1.3101	3.0720	10.9742	1.3061	1.8370	11.7671	5.8169	6.3801	9.6841
1.3189	3.0720	11.0748	1.3080	1.8370	11.9882	5.8188	6.3801	9.7229
1.3299	3.0720	11.2541	1.3080	1.8370	14.9510	5.8200	6.3801	10.0322
1.3309	3.0730	11.2641	1.3092	1.8370	15.0042	5.8239	6.3801	11.2870
Tempo Módulo Teste (em milissegundos)								
Teste 1			Teste 2			Teste 3		
Mínimo	Médio	Máximo	Mínimo	Médio	Máximo	Mínimo	Médio	Máximo
32.5201	43.2410	46.2511	33.0729	43.2129	46.4308	31.6162	43.2382	46.3381
36.0022	43.2410	46.2539	33.4151	43.2131	46.5338	34.7669	43.2389	46.3381
36.1350	43.2429	46.2570	35.9471	43.2138	46.9260	35.9662	43.2410	46.4160
36.1769	43.2429	46.2570	36.0019	43.2138	49.7189	36.0200	43.2410	46.4568
36.1900	43.2439	46.2639	36.0970	43.2141	49.9389	36.0279	43.2422	46.5081
36.2132	43.2460	46.2952	36.1300	43.2150	51.2471	36.0832	43.2432	46.8280
36.2229	43.2479	46.2990	36.1400	43.2181	51.3289	36.1588	43.2451	46.9301
36.2430	43.2489	46.3281	36.1440	43.2210	51.8138	36.1660	43.2470	48.7530
36.2470	43.2489	47.8199	36.1731	43.2220	51.8861	36.1669	43.2470	49.1629
36.2530	43.2491	50.5819	36.1738	43.2239	60.5922	36.1679	43.2479	53.3550
Tempo Módulo Captura (em milissegundos)								
Teste 1			Teste 2			Teste 3		
Mínimo	Médio	Máximo	Mínimo	Médio	Máximo	Mínimo	Médio	Máximo
208.540	230.028	243.442	208.822	229.953	243.491	208.863	229.334	243.328
208.601	230.032	243.461	209.084	229.971	243.508	209.222	229.337	243.465
210.120	230.043	243.478	210.558	229.971	243.521	209.416	229.338	243.516
210.153	230.045	243.478	211.315	229.971	243.565	209.636	229.345	243.525
211.599	230.051	243.528	211.408	229.974	243.593	209.786	229.346	243.572
212.067	230.054	243.551	211.867	229.976	243.763	209.911	229.348	243.717
212.134	230.061	243.629	212.998	229.980	243.887	210.016	229.349	243.761
212.197	230.069	245.344	213.166	229.985	245.983	210.124	229.351	246.279
212.485	230.071	245.513	213.168	229.985	246.129	210.815	229.354	247.532
212.817	230.074	246.702	213.195	229.986	247.762	211.265	229.356	247.732

Apêndice C

Lâmpada Incandescente		Lâmpada Led		Sem Conexão	
Código de Captura	**Similaridade Somada**	**Código de Captura**	**Similaridade Somada**	**Código de Captura**	**Similaridade Somada**
Maiores Valores					
86620	99.9124	92434	99.3921	98355	69.1393
86659	99.9107	92433	99.3908	98345	69.0408
86641	99.9106	92489	99.3893	98377	68.9021
86587	99.9103	92511	99.3852	98369	68.6151
86582	99.9101	92440	99.3837	98330	68.5943
86656	99.9094	92482	99.3770	98337	68.4477
86562	99.9089	92447	99.3750	98319	68.4418
86565	99.9082	92492	99.3671	98375	68.0873
86604	99.9077	92459	99.3623	98365	68.0484
86644	99.9073	92452	99.3500	98352	67.9759
Valores Médios					
86576	99.8846	92497	99.2603	98283	65.5041
86575	99.8845	92484	99.2523	98322	65.3120
86643	99.8844	92518	99.2456	98321	65.2016
86637	99.8834	92439	99.2434	98348	65.1202
86573	99.8820	92460	99.2388	98281	64.9124
86567	99.8806	92495	99.2340	98299	64.7647
86632	99.8802	92453	99.2320	98333	64.7573
86628	99.8796	92507	99.2306	98351	64.6944
86653	99.8781	92426	99.2191	98353	64.6061
86591	99.8779	92455	99.2172	98303	64.5995
Menores Valores					
86616	99.7773	92505	98.3391	98328	60.3202
86618	99.7739	92432	98.2909	98349	60.2198
86568	99.7738	92496	98.2702	98317	59.6553
86615	99.7736	92513	98.1887	98335	59.4807
86588	99.7724	92421	98.1440	98378	59.3454
86626	99.7719	92463	98.0752	98360	58.6938
86647	99.7710	92429	97.6208	98308	56.9455
86595	99.7416	92515	97.6144	98327	56.8741
86617	99.7189	92437	97.3372	98290	56.1486
86561	98.8787	92465	96.1596	98347	56.0215

Apêndice D

Lâmpada Incandescente		Lâmpada Led		Sem Conexão	
Código de Captura	**Similaridade**	**Código de Captura**	**Similaridade**	**Código de Captura**	**Similaridade**
Maiores Valores					
86737	0.9999	94771	0.9997	103263	0.9181
86818	0.9999	95242	0.9997	98432	0.9092
86823	0.9999	96013	0.9997	121147	0.8995
86832	0.9999	94216	0.9996	98882	0.8980
86918	0.9999	93378	0.9995	138554	0.8975
86928	0.9999	94272	0.9995	121321	0.8968
86983	0.9999	94298	0.9995	133600	0.8958
87163	0.9999	94429	0.9995	103612	0.8935
87287	0.9999	94539	0.9995	104109	0.8894
87321	0.9999	94809	0.9995	117009	0.8881
Valores Médios					
125010	0.9993	114050	0.9966	120652	0.6758
125043	0.9993	114274	0.9966	133937	0.6758
125047	0.9993	114287	0.9966	134563	0.6758
125157	0.9993	114316	0.9966	136456	0.6758
125179	0.9993	114406	0.9966	136738	0.6758
125187	0.9993	114443	0.9966	99838	0.6757
125207	0.9993	114465	0.9966	100409	0.6757
125259	0.9993	114469	0.9966	100870	0.6757
125272	0.9993	114541	0.9966	101426	0.6757
125278	0.9993	114542	0.9966	101864	0.6757
Menores Valores					
87158	0.9963	111505	0.9621	117460	0.4349
90020	0.9963	95437	0.9609	100161	0.4338
104141	0.9963	113274	0.9605	104107	0.4312
109345	0.9963	112652	0.9600	104138	0.4309
121543	0.9963	112382	0.9592	118727	0.4298
107049	0.9962	114063	0.9583	137766	0.4290
121477	0.9962	113329	0.9567	98839	0.4228
121597	0.9962	112086	0.9555	98961	0.4223
126128	0.9961	95735	0.9515	135836	0.4146
107638	0.9960	112571	0.9499	118785	0.4081