UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

BRENO CASTRO CARDOSO

DESENVOLVIMENTO DE UMA TOMADA CONECTADA COM AUTODIAGNÓSTICO PARA A INTERNET DAS COISAS

Joinville
2018

BRENO CASTRO CARDOSO

DESENVOLVIMENTO DE UMA TOMADA CONECTADA COM AUTODIAGNÓSTICO PARA A INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Prof. Dr. Gian Ricardo Berkenbrock

Joinville
2018

RESUMO

Feedback em tempo real do consumo em residências mostra-se como uma forma eficiente de reduzir o consumo nelas, tendo isso em mente esse trabalho tem como objetivo criar um dispositivo de medição de consumo de energia aplicado a internet das coisas que sejam rápidos, seguros e possuam uma interface amigável. Para isso foi criado uma tomada conectada com base na placa de hardware EmoteIII com o objetivo de adquirir os dados de consumo e um dispositivo gateway composto de um servidor e um EmoteIII com o software OpenHAB. Essa arquitetura tem o objetivo de prover uma interface com o usuário, esses dispositivos serão então conectados através de uma rede sem fio através dos transceptores ZigBee disponíveis nos EmoteIII e o gateway terá interface com dispositivos moveis e smartphones através do openHab. Com o objetivo de aumentar a confiabilidade desses dispositivos foi implementado ainda um sistema de autodiagnóstico capaz de identificar falhas de hardware possibilitando ao dispositivo assumir um estado seguro.

Palavras-chave: internet das coisas, openHab, EmoteIII, autodiagnóstico, IoT.

ABSTRACT

Real-time feedback from residential eletrical consumption is shown as an efficiente way to reduce consumption in homes. This work aims to create a device for measuring energy consumption applied to the Internef of Things that are fast, safe and have an friendly interface. For this purpose, an outlet connected to the EmoteIII hardware board was created in order to acquire the consumption data and a device composed of a server and an EmoteIII with the OpenHAB software was created to work as a gateway with the OpenHAB software. This architecture is intended to provide a user interface. These devices will then be connected through a wireless network through the ZigBee transceivers available in EmoteIII and the gateway will interface with mobile devices and smartphones through openHab. In order to increase the reliability of these devices, a self-diagnostic system capable of identifying hardware failures was implemented, enabling the device to assume a safe state.

Keywords: internet of things, openHab, EmoteIII, self-diagnostic, IoT.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACEEE	American Council for an Energy-Efficient Economy
ADESD	Application-Driven Embedded System Design
API	Application Programing Interface
BOM	Bill of Materials
EPOS	Embedded Parallel Operating System
IoT	Internet of Things
ITU	International Telecommunication Union
LISHA	Software/Hardware Integration Lab
OpenHAB	Open Home Automation Bus
RFID	Radio-Frequency Identification

SUMÁRIO

1 INTRODUÇÃO

O primeiro uso do termo Internet of Things (IoT) foi atribuído ao laboratório Auto-ID, um laboratório dedicado a pesquisas com Radio-Frequency Identification (RFID) e sensores, no entanto, o termo sofreu modificações e atualmente existem muitas definições (ATZORI; IERA; MORABITO, 2010). A International Telecommunication Union (ITU) define a IoT como uma infraestrutura global que permite serviços avançados através da conexão das coisas (ITU, 2012).

Conectar os ambientes físicos e virtuais representa um grande avanço, pois sem isso todas as informações do mundo virtual são dependentes de humanos, mas esses têm tempo, precisão e atenção limitadas. A IoT permite aos computadores terem acesso aos dados do ambiente sem ajuda dos seres humanos, o que permite enorme quantidade de dados ser adquirida e analisada fornecendo percepções mais rápidas e precisas do ambiente (ASHTON, 2011).

Uma das aplicações no domínio da IoT é a domótica, que aplica às residências as tecnologias de IoT, proporcionando a gestão dos recursos habitacionais. A domótica tem como objetivo simplificar a vida das pessoas automatizando tarefas rotineiras, aumentando o conforto, segurança e a eficiência de residência levando a uma melhoria na qualidade de vida das pessoas (COSTA, 2012).

A miniaturização e redução de custos dos componentes eletrônicos, assim como o estado avançado das tecnologias de comunicação, o campo da domótica tem sido cada vez mais atrativo. Assim permitindo que as pessoas interajam com os dispositivos conectados de suas casas (COSTA, 2012).

A domótica não se restringe apenas a edifícios residenciais, mas também a ambientes industriais e corporativos, onde é cada vez mais aplicada como forma de melhorar a eficiência energética (COSTA, 2012). O gerenciamento de consumo de energia e água tornam o ambiente de trabalho mais produtivo, saudável e eficiente, contribuindo para o aumento da produção e diminuição dos custos operacionais (DIAS, 2004).

Com a aplicação dessas tecnologias, é possível construir dispositivos para permitir o controle e o acesso de informações sobre os recursos. De acordo com a American Council for an Energy-Efficient Economy (ACEEE), em uma pesquisa realizada entre 1995 e 2010 o acesso às informações de consumo de eletricidade causa uma redução do consumo. A pesquisa encontrou uma redução média de 12% no consumo quando há o acesso a informações em tempo real discriminadas por aparelho (EHRHARDT-MARTINEZ; DONNELLY; LAITNER, 2010). Esses resultados demostram como o desenvolvimento de dispositivos de medição podem impactar no consumo de recursos em uma habitação.

Uma importante razão para a inclusão de sistema de autodiagnóstico nos dispositivos eletrônicos é o aumento da confiabilidade deles. Esses sistemas podem ser usados para detectar previamente falhas ou para a identificação das falhas ocorridas no sistema permitindo que o mesmo possa assumir um estado seguro (OLBRICH; RICHARDSON; BRADLEY, 1996). Esses sistemas podem ser aplicados a dispositivos de domótica para o gerenciamento das falhas. Segundo Utton e Scharf (2004) as casas conectadas serão ambientes inerentemente complexos compostos de dispositivos de diferentes fabricantes e falhas irão ocorrer, nesses ambientes o gerenciamento de falhas será critico e a identificação da falha a componentes individuais do sistema será um requisito chave.

Nessa perspectiva, esse trabalho visa desenvolver uma tomada conectada para medir e proporcionar acesso às informações de consumo. Para isso, será utilizado o hardware EPOSmote III e o sistema operacional EPOS como plataforma de desenvolvimento.

1.1 OBJETIVOS

Para resolver a problemática da aplicação da domótica como forma de aumentar a eficiência energética, propõe-se neste trabalho os seguintes objetivos.

1.1.1 Objetivo Geral

Propiciar a ambientes a capacidade de medir e informar o consumo de energia elétrica de cargas a ele conectado.

1.1.2 Objetivos Específicos

- Desenvolver uma tomada conectada
- Projetar e construir um protótipo do dispositivo
- Analisar erros e parâmetros de desempenho do dispositivo
- Examinar a viabilidade técnica do dispositivo

2 REVISÃO TEÓRICA

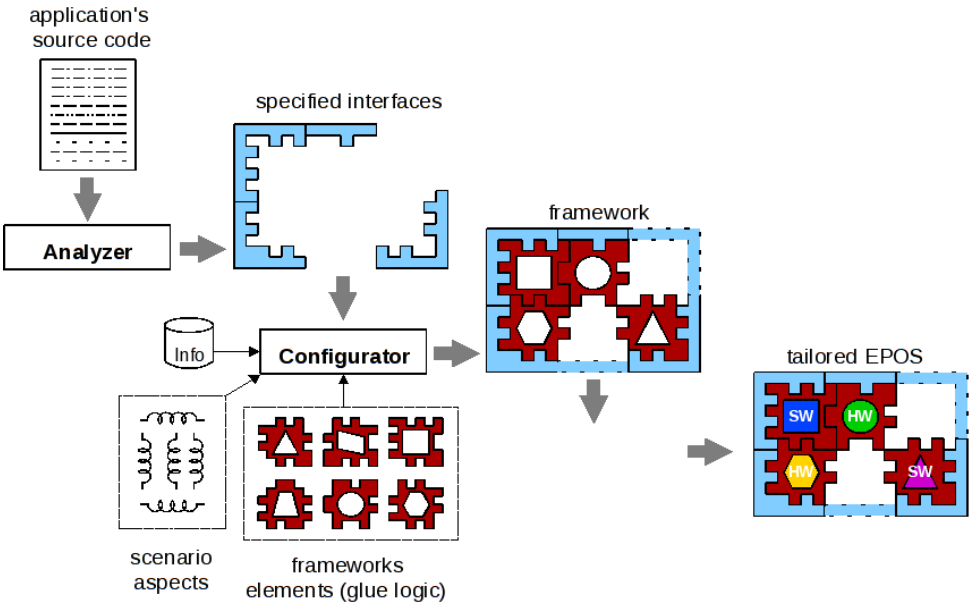
Neste capítulo serão apresentados os principais conceitos utilizados nesse trabalho. Inicialmente, o capítulo trata sobre o sistema operacional EPOS. Em seguida ele mostra o hardware EmoteIII, apresenta ainda o software de interface para IoT, OpenHab. Descreve também, o controle de potência em circuitos alternados utilizando Triacs, a teoria de autodiagnóstico em sistemas eletrônicos, das fontes de alimentação para circuitos de corrente continua e, por fim, apresenta os principais conceitos envolvidos com a medição de potência elétrica.

2.1 EMBEDDED PARALLEL OPERATING SYSTEM

O Embedded Parallel Operating System (EPOS) é um sistema operacional desenvolvido para sistemas embarcados mantido pelo Software/Hardware Integration Lab (LISHA). Ele implementa uma série de recursos que tem como objetivo auxiliar no desenvolvimento da aplicação.

O EPOS é programado na linguagem de programação C++ e utiliza de programação orientada a objeto e metaprogramação. Ele também implementa o método Application-Driven Embedded System Design (ADESD) onde o sistema operacional e construído apenas com os recursos necessários para cada aplicação dessa forma o sistema pode utilizar de metaprogramação para otimizar o máximo possível o código final diminuindo o impacto do sistema operacional no desempenho da aplicação (LISHA, 2017b). A Figura 1 ilustra o processo de criação da aplicação.

Figura 1 – Processo de criação da aplicação no EPOS.

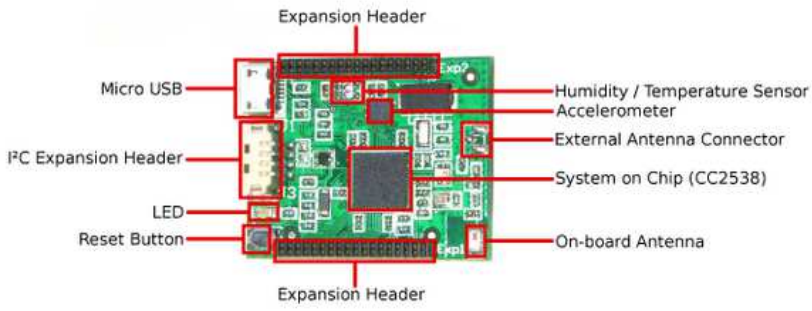


Fonte: LISHA (2017c)

2.2 EMOTE III

O Emote III é uma plataforma de hardware que segue o princípio de ADESD que o EPOS implementa, essa plataforma também é mantida pelo LISHA. Uma ilustração da plataforma e seus principais componentes podem ser visto na Figura 2.

Figura 2 – Emote III.



Fonte: LISHA (2017a)

O Emote III utiliza um microcontrolador da Texas Instruments CC2538, o qual contém um processador ARM Cortex-M3. Esse microcontrolador possui um transceptor de rádio frequência integrado de 2.4GHz e em conformidade com a norma IEEE 802.15.4 e um conversor analógico digital de 12-bit (LISHA, 2017a).

A plataforma ainda implementa outros periféricos externos ao microcontrolador como sensores, botões e conectores para expansão conforme pode ser visto na Figura 2, além disso ele ainda possui um regulador de tensão 3V3 que facilita a conexão com o USB (LISHA, 2017a).

2.3 OPENHAB

Há disponível comercialmente muitos dispositivos para domótica, no entanto não há uma padronização de Application Programing Interfaces (API's), protocolos ou interface tornando inviável a um usuário utilizar dispositivos de diferentes fabricantes em uma mesma interface e residência.

O Open Home Automation Bus (OpenHAB) é um software de código aberto, mantido pela openHab Foundation, criado como uma solução para esse problema, ele implementa uma interface unificada e modular onde com poucas modificações é possível implementar novos protocolos, dispositivos e API's na mesma interface independente do fabricante. Ele é capaz de rodar em dispositivos com arquitetura ARM desde que esses suportem Linux, Windows ou Mac OS X e possuam instalado o Java (openHab, 2017).

Além de prover uma interface única o OpenHAB ainda permite a definição de regras onde é possível controlar alguma parte da casa a partir de dados de algum sensor, por exemplo ligar a luz quando a porta for aberta, mesmo que os dispositivos que controlam a luz e verificam o estado da porta sejam de fabricantes diferentes (openHab, 2017).

O OpenHab esta em sua segunda versão no entanto no contexto desse trabalho foi utilizado o openHab 1 e portanto serão os conceitos que serão abordados são os relacionados a versão 1.

2.3.1 Item

Itens são substituição do openHab para dispositivos ou serviços, enquanto esses podem ser bem especifico com diferentes protocolos, medidas e configurações itens são substitutos genéricos para eles. Um Item pode ser do tipo String, Número, Interruptor ou de algum dos outros itens básicos no openHab (openHab, 2017).

Itens são tipo de dados básicos que tem estados que podem ser escrito e lidos. Itens podem não apenas serem alterados por software como podem ser conectados a um canal de um binding dessa forma podendo ser alterados por eventos do ambiente (openHab, 2017).

2.3.2 Sitemaps

Sitemaps são usados para selecionar e preparar elementos como itens para serem mostrados por alguma das interfaces de usuários disponíveis no openHab (openHab, 2017).

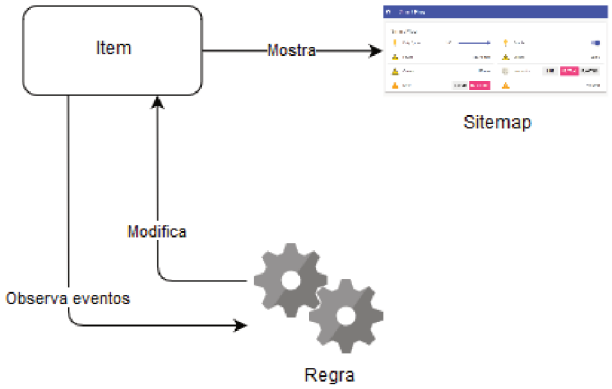
2.3.3 Regras

Regras são usadas para automatizar processos. Uma regra é acionada por algum evento e então ela roda um script que executará as ações definidas. Uma regra pode ser usada, por exemplo, para acender uma luz se a porta for aberta (openHab, 2017).

2.3.4 Relação entre os conceitos

A relação entre os conceitos do openHab está apresentada na Figura 3. Ela ilustra que uma regra pode observar os itens para identificar eventos e pode também modificar esses itens. Um sitemap tem a função de mostrar as informações que estão armazenadas nos itens.

Figura 3 – Relação entre os conceitos do openHab.

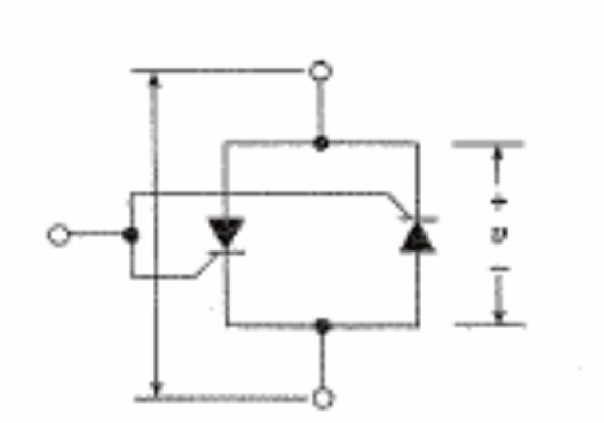


Fonte: Autor (2018)

2.4 CONTROLE DE POTÊNCIA EM CIRCUITOS AC

Um dos componentes que podem ser usados para chavear corrente alternadas são os Triacs, esses podem ser modelado por dois Silicon Controled Rectifier (SCR) em antiparalelo, conforme ilustrado Figura 4. Os quais são um Tiristor que pode chavear correntes de alto valores, no entanto ele só permite a passagem de corrente em um sentido. O Triac sendo modelado por dois SRC em antiparalelo, permite o chaveamento da corrente de forma bidirecional (MALVINO; BATES, 2007).

Figura 4 – Modelo do Triac.

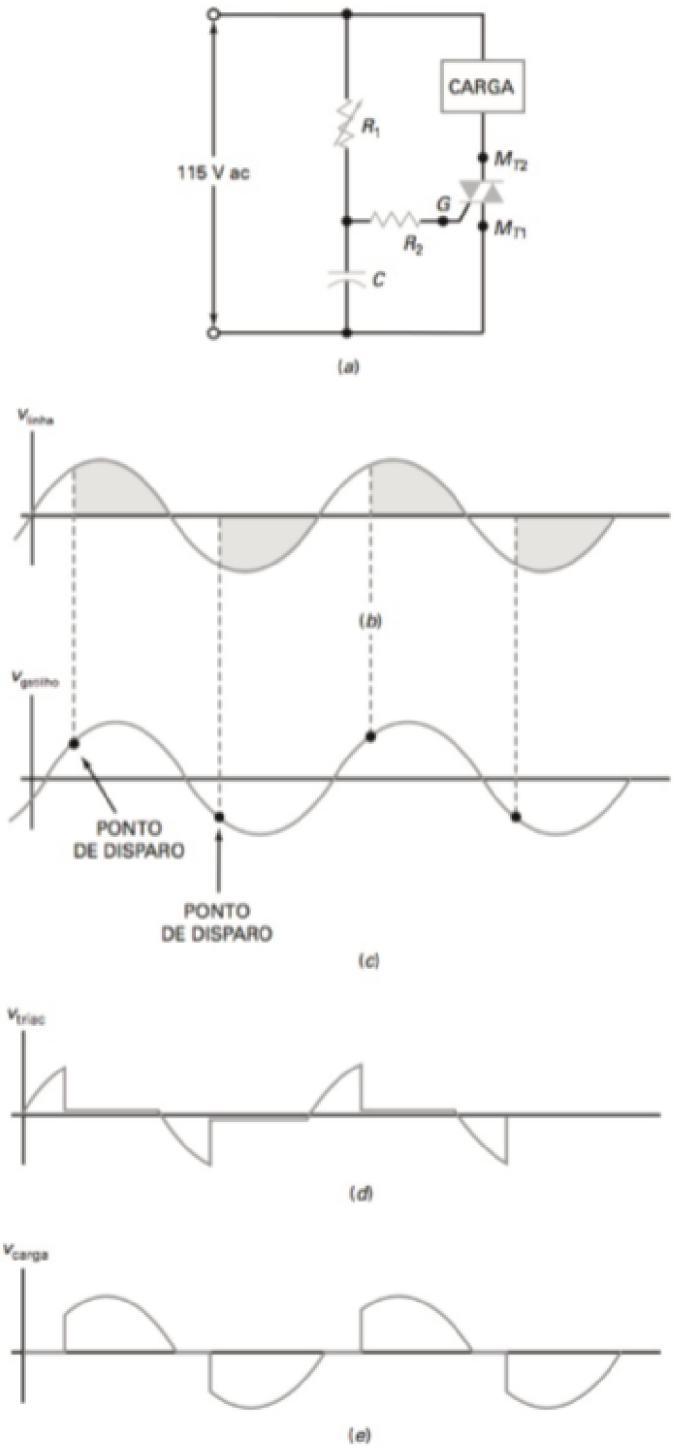


Fonte: MALVINO; BATES (2007)

$$V_{in} = V_{GT} + I_{GT}R_G \tag{2.1}$$

Os SCR's entram em condução quando a tensão no gate é maior que a tensão de gatilho representada na Equação 2.1. E se mantém em condução até que a corrente do anodo para o catodo seja menor que a corrente de manutenção. Dispositivos como esses podem controlar cargas de maior valor de corrente pelo uso do controle de fase, o que permite o controle da corrente média na carga. Na Figura 5, pode-se observar o processo de controle de fase. Como pode ser visto com a variação do ponto de disparo 5(c) é possível modificar a tensão na carga 5(e) (MALVINO; BATES, 2007).

Figura 5 – Controle de fase com Triac.



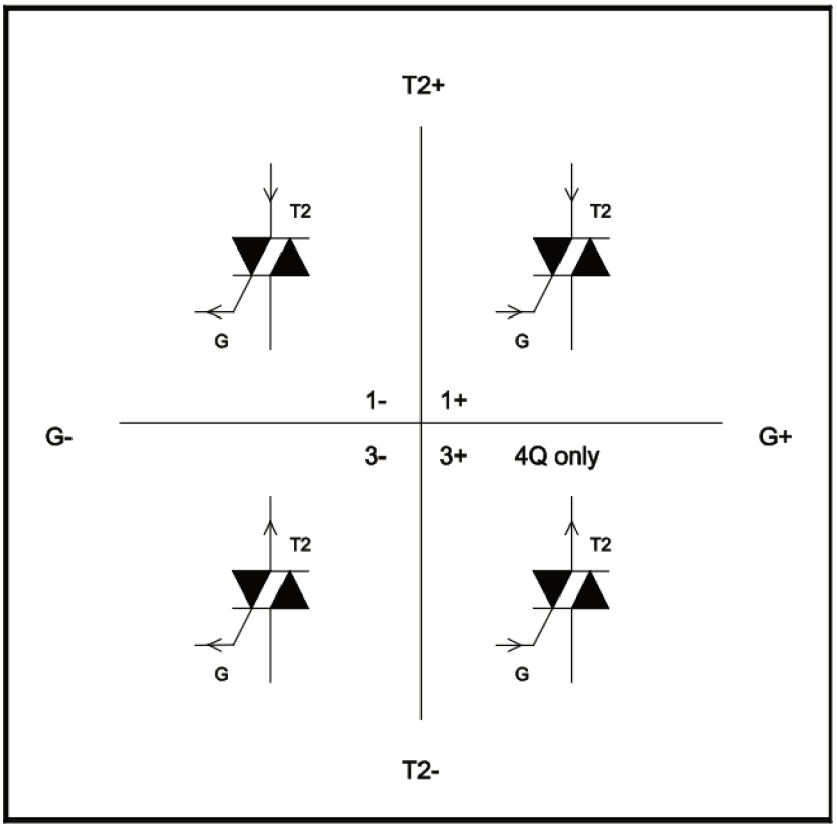
Fonte: MALVINO; BATES (2007)

2.4.1 Quadrantes de operação

A corrente miníma no gatilho dos Triac's varia em qual quadrante esse opera, por exemplo para o Triac BT137S a corrente de gate é de 10 ma para os quadrantes 1,

2 e 3. E de 25 ma para o quadrante 4 (NXP, 2014). Esses quadrantes são definidos pela corrente no gatilho e a tensão entre so terminais T1 e T2 do Triac. A Figura 6 ilustra esses quadrantes.

Figura 6 – Quadrantes de operação dos Triac.



Fonte: NEXPERIA (2013)

A operação do Triac em quadrantes de menor corrente oferece a vantagem de poder aciona-lo com drivers que são menos capazes. Nesse trabalho objetivou-se a utilização do Triac nos quadrantes 2 e 3 para que esta possa ser ativado pelo microcontrolador.

2.5 AUTODIAGNÓSTICO

As principais razões para a implementação de sistemas de autodiagnóstico são o aumento da confiabilidade, da testabilidade e a diminuição do tempo de testes. Diferentes técnicas podem ser usadas para a identificação de falhas, por exemplo, a redundância temporal que pode ser implementada com três etapas em um sistema de medição.

- Medir o valor da entrada

- Trocar o valor da entrada para um valor de referencia, medir e comparar com o resultado esperado
- Medir o valor de entrada novamente e comparar com a primeira medida

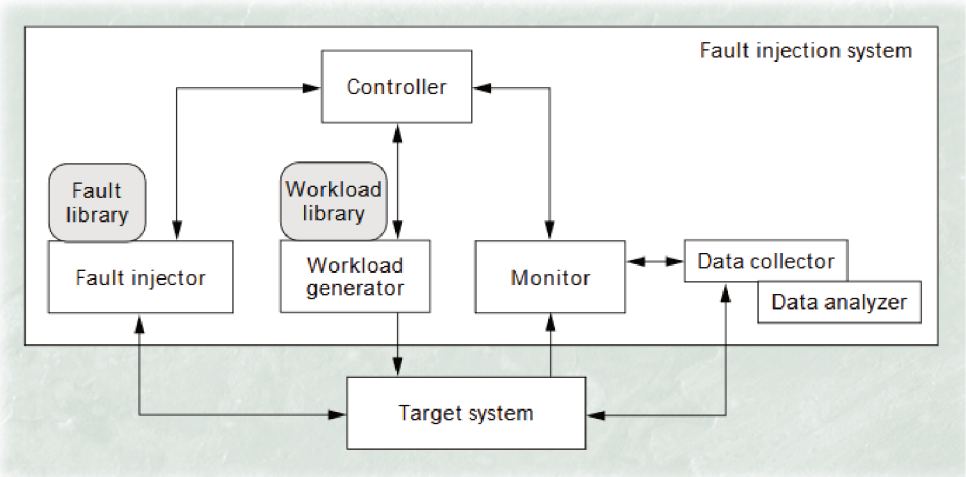
Duas técnicas utilizadas para a identificação de falhas de hardware são a construção de circuitos onde partes desses são utilizadas para esse se autoverificar e a comparação da saída de algum circuito com resposta esperada dele. Essas técnicas permitem que o sistema seja verificado sem a necessidade de parar a operação normal (OLBRICH; RICHARDSON; BRADLEY, 1996).

2.5.1 Injeção de falhas

Em alguns casos apenas a construção de sistemas com identificação de falhas não é o suficiente para garantir que todos os efeitos criticos que podem ser causados por uma falha estão tratados. Além disso, existem um equilíbrio entre o número de falhas identificadas por um sistema e os custos induzidos por esse. Algumas falhas só podem ter sua criticidade verificada quando essas acontecem na aplicação. (ZIADE; AYOUBI; VELAZCO, 2003)

Para executar essa verificação é utilizada a técnica de injeção de falha, onde através de mecanismos de hardware ou software uma falha é inserida no sistema para analisar a resposta dele. Para isso é construído um ambiente de injeção de falha conforme na Figura 7, nesse sistema o Fault injector injeta a falha no alvo enquanto esse executa comandos do Workload generator e o Monitor coleta os dados quando necessário (HSUEH; TSAI; IYER, 1997).

Figura 7 – Ambiente de injeção de falha.

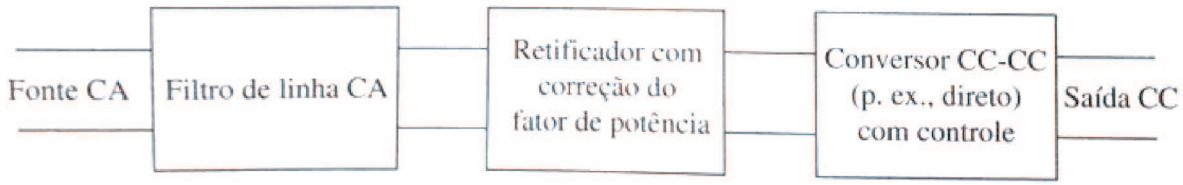


Fonte: HSUEH; TSAI; IYER (1997)

2.6 FONTE DE ALIMENTAÇÃO CC

Uma fonte de alimentação completa geralmente consiste de uma entrada CA, um filtro de linha CA, um retificador com correção do fator de potência e um conversor CC-CC. A Figura 8 mostra o modelo de uma fonte de alimentação completa (HART, 2012).

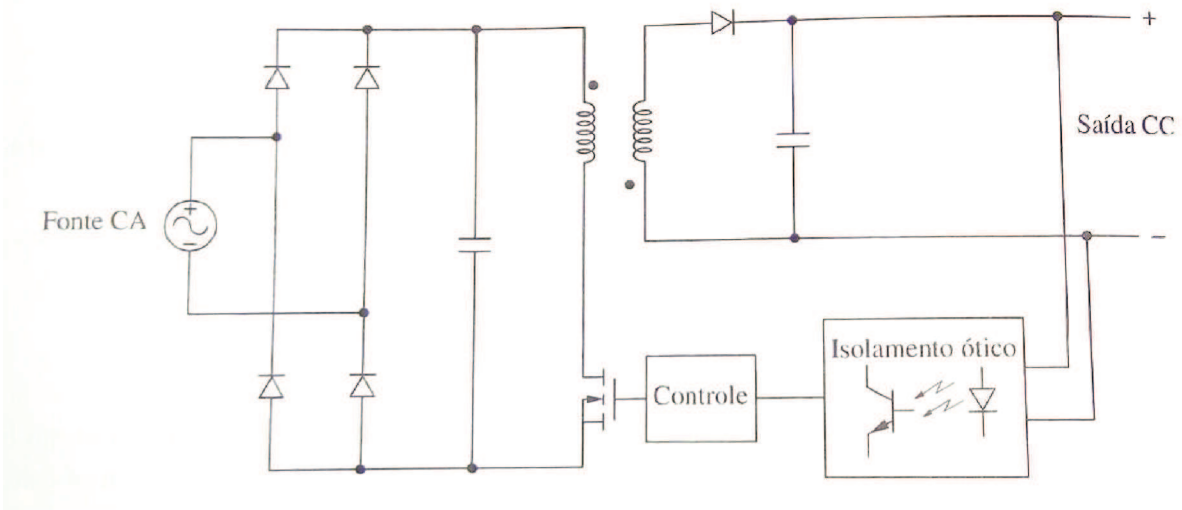
Figura 8 – Fonte de alimentação completa.



Fonte: HART (2012)

Para aplicações de baixa potência podem ser implementadas topologias simplificadas chamadas de conversor fora de linha. A Figura 9 ilustra uma fonte de alimentação fora de linha que utiliza uma ponte completa de retificação com um filtro capacitivo para produzir uma tensão CC a partir da tensão CA e um conversor CC-CC flyback que reduz a tensão de CC. Esses circuitos normalmente utilizam um circuito integrado para o controle da fonte utilizando uma malha de feedback da tensão de saída (HART, 2012).

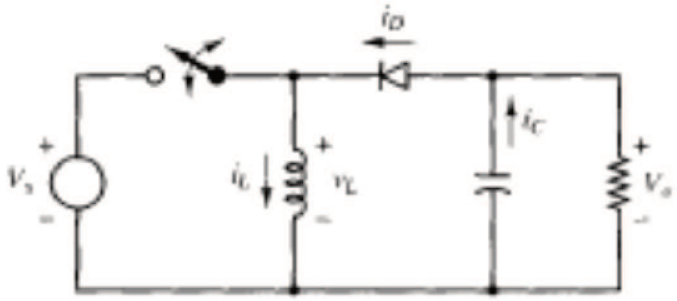
Figura 9 – Fonte Fora de linha.



Fonte: HART (2012)

Esse trabalho utilizou dessa mesma topologia, no entanto, foi a topologia do conversor CC-CC utilizada foi a Buck-Boost. Sua topologia é mostrada na Figura 10, esse conversor permite que a tensão de saída tenha uma magnitude maior ou menor que a tensão de entrada . No entanto, ele produz uma tensão de saída com polaridade invertida o que pode ser uma desvantagem para algumas aplicações.(HART, 2012)

Figura 10 – Conversor CC-CC Buck-Boost.



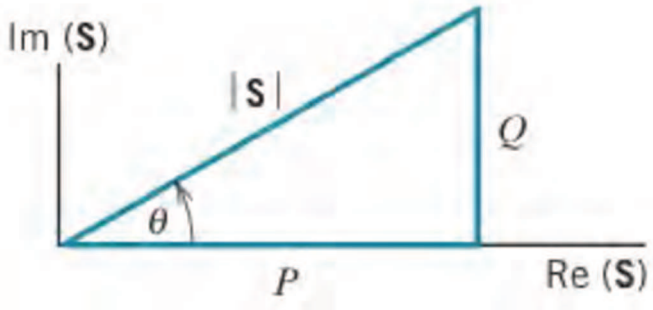
Fonte: HART (2012)

2.7 MEDIÇÃO DE POTÊNCIA ELÉTRICA

A potência elétrica é uma grandeza de interesse no ambiente da domótica. Com ela é possível monitorar a quantidade de energia gasta por um equipamento, sendo um dos dados de maior interesse para a aquisição com dispositivos de domótica.

A potência em sistemas que trabalham com tensão alternada é complexa sendo essa composta de uma componente real chamada de potência ativa e uma componente alternada chamada de potência reativa. A Figura 11 ilustra esse conceito, sendo P a potência média, S a potência aparente e Q a potência reativa (DORF, 2003).

Figura 11 – Triângulo de potências.



Fonte: DORF (2003)

A equação 2.2 pode ser usado para o cálculo da potência média e o módulo da potência aparente, para ambos os casos é necessário os valores de corrente e tensão.

$$P = \frac{1}{T}\int_{t_0+T}^{t_0} p(t)dt, p(t) = v(t)i(t) \tag{2.2}$$

Destaca-se que no contexto desse trabalho considerou-se a potência média mais significativa ao usuário e no resto do trabalho a potência média foi tratada apenas como potência.

3 PROJETO E ANÁLISE DE REQUISITOS

O trabalho objetiva propiciar a ambientes a capacidade de medir e informar o consumo elétrico, a fim de atingir esse objetivo é proposto a criação de uma tomada conectada capaz de medir o consumo de aparelhos a ela conectadas. Para isso foi realizado um levantamento de requisitos. Dessa forma, essa seção analisa os requisitos necessários e a adequação da solução implementada.

3.1 LEVANTAMENTO DE REQUISITOS

Para poder cumprir o objetivo apresentado foram definidos os requisitos do projeto que estão listados a seguir.

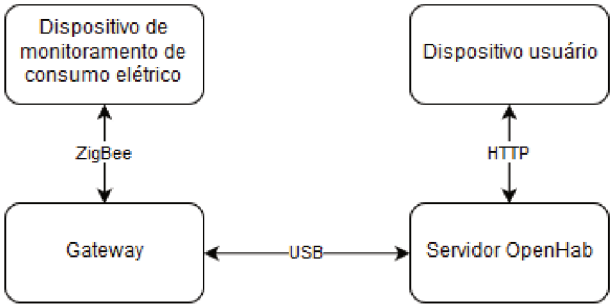
1. Interface gráfica: o sistema deve prover uma interface em que o usuários possa acessar seus dados de consumo a partir de um computador ou de um smartphone.
2. Valor: o sistema deve ter um baixo custo quando comparado com as opções comerciais disponibilizadas.
3. Temporal: os dados de consumo devem ser mostrados com um atraso limite de 2 segundos.
4. Coleta: o sistema deve ser capaz de medir a potência elétrica com um erro máximo de 5%, a aquisição da potência elétrica deve ocorrer em cada tomada.
5. Atuação: o usuário deve ser capaz de atuar sobre o sistema elétrico desligando os aparelhos conectados e controlando a potência das cargas por dimerização. Além disso o sistema deve responder em no máximo 0,5 segundos a comandos.
6. Confiabilidade: o sistema deve ser capaz de identificar e tratar as principais falhas a qual está sujeito.

3.2 ARQUITETURA PROPOSTA

A fim de cumprir os requisitos estabelecidos foi criado dois dispositivo um com a função de gateway que poderá se conectar a um computador possibilitando a interface do sistema com o openHab e outro com a função de medição da potência elétrica e controle das cargas a ele acopladas. A comunicação entre os dispositivos se da através do ZigBee e usa como camada física os transceptores de rádios disponiveis no Emote III. Há também a comunicação do dispositivo de gateway com um computador servidor

do OpenHab que foi feita através de comunicação Universal Serial Bus (USB). A Figura 12 demostra os protocolos de comunicação utilizados.

Figura 12 – Topologia de Comunicações do sistema.



Fonte: O Autor (2018)

Com o sistema apresentado pretende-se cumprir os requisitos estabelecidos. No entanto, isso depende das propriedades e implementação de cada parte do sistema dessa forma será discutido a implementação de cada uma.

3.2.1 Dispositivo de interface

O dispositivo de interface possui a função de prover uma interface que possa ser acessada pelo usuário por meio de um computador ou smartphone. Com o intuito de cumprir esse requisitos foi utilizado um Emote III com a função de gatilhoway, como pode ser visto na Figura 12 ele faz a tradução bidirecional entre o USB e o ZigBee. O ZigBee será então utilizado na comunicação sem fio dos dispositivos de monitoramentos de consumo elétrico. Compõe ainda o dispositivo de interface um computador que será usado como servidor para o openHab e faz a comunicação com o dispositivo usuário.

3.2.2 Dispositivo monitor de consumo elétrico

O monitor de consumo elétrico possui a função de medir o consumo de energia e prover atuação do usuário sobre o sistema elétrico. A fim de cumprir essa função, o sistema proposto possui as seguintes funcionalidades.

- Medição de corrente e tensão para cálculo da potência.
- Atuação no circuito principal.
- Alimentação do sistema.
- Comunicação com o gateway.

Tendo em vista que esse dispositivo pode ser instalado em todas as tomadas de uma casa, um dos requisitos que foi dado a maior importância foi o custo. Com o objetivo de cumprir esse requisito, o sistema usa soluções de baixo custo. A Tabela 1 mostra a solução adotada para cada funcionalidade.

Tabela 1 – Soluções propostas

Funcionalidade	**Solução**
Medição de corrente	Resistor Shunt e INA219
Medição de tensão	Conversor analógico digital disponível no Emote III
Atuação	Triac
Alimentação do sistema	Conversor Buck-Boost não isolado
Comunicação	Zigbee Emote III

Com o sistema proposto para esse nó é possível garantir os requisitos de baixo custo e medição da potência elétrica consumida, assim como de atuação nas cargas acopladas ao dispositivo.

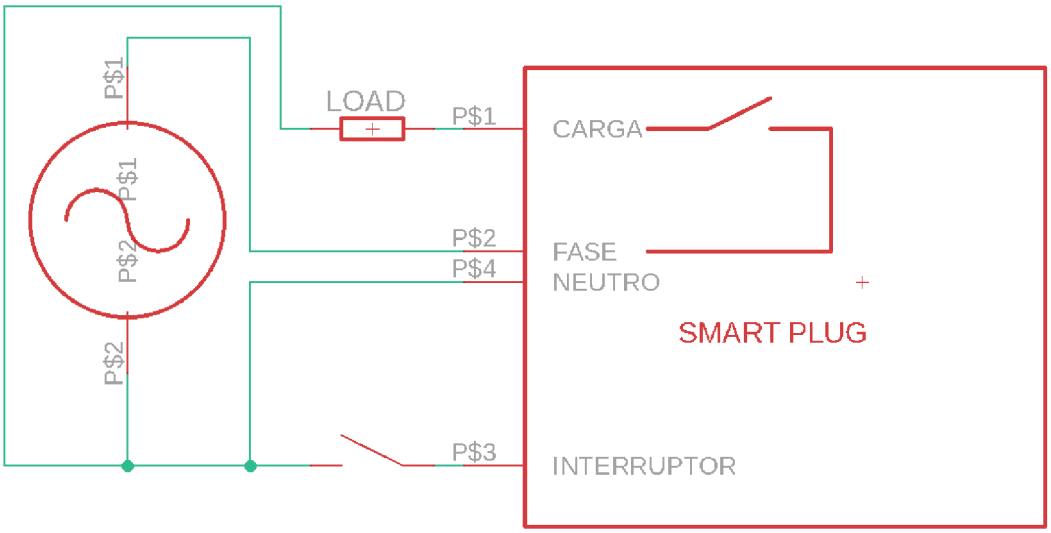
3.2.3 Projeto do dispositivo monitor do consumo elétrico

Para cumprir os requisitos e soluções estabelecidas foi desenvolvido um dispositivo que pode ser incluído dentro de uma caixa de tomada visto que esse é o lugar esperado para a instalação do dispositivo. Foi necessário o desenvolvimento de hardware e software para esse sistema, os quais serão detalhados.

3.2.3.1 Hardware

A interface do hardware desenvolvido é feita através de 4 conexões elétricas, duas para a alimentação do dispositivo, uma para a carga e uma para uso do interruptor. O sistema funciona como um interruptor que chaveará um dos pinos de alimentação para a carga, a Figura 13 ilustra esse conceito e demostra como a carga e o interruptor podem ser conectados ao sistema. É importante destacar que nesse diagrama assim como em todos os diagramas subsequentes os termos fase e neutro são usados apenas com o objetivo de diferenciar as duas linhas de entrada da rede podendo eles serem invertidos sem alteração no funcionamento do produto.

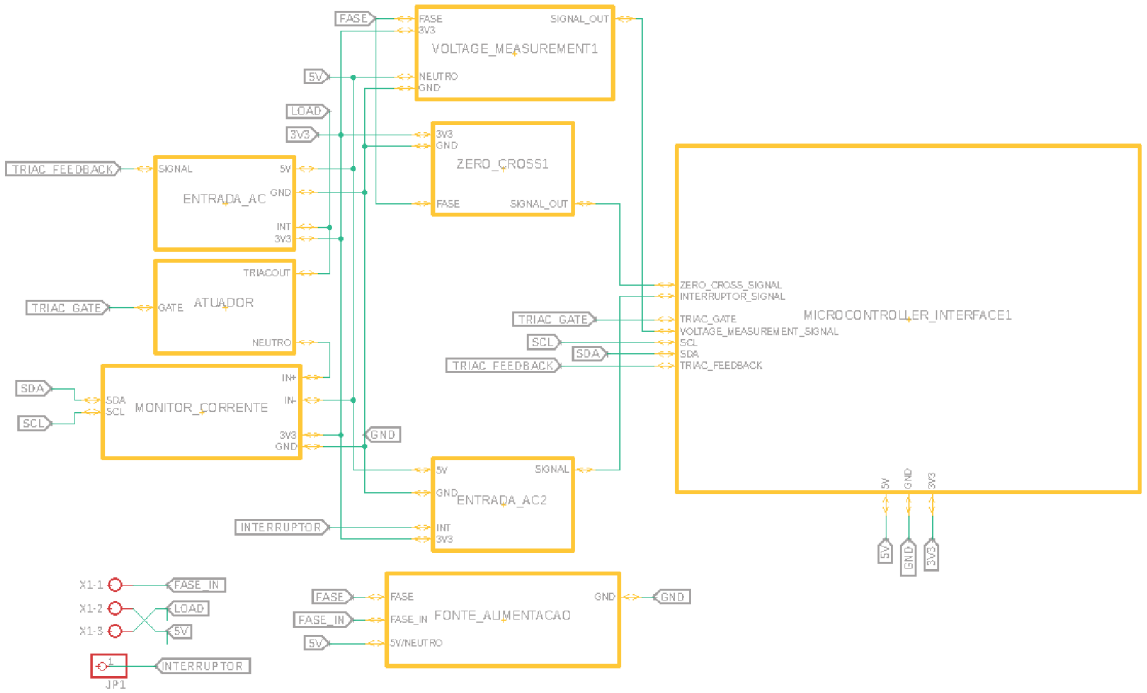
Figura 13 – Diagrama de alto nivel da instalação.



Fonte: O Autor (2018)

Para o funcionamento do sistema com todas as suas funcionalidades foi necessário o desenvolvimento de vários circuitos com diferentes funções. O diagrama da Figura 14 mostra a ligação entre os diferentes circuitos do sistema.

Figura 14 – Diagrama de conexões entre os diferentes circuitos do dispositivo.

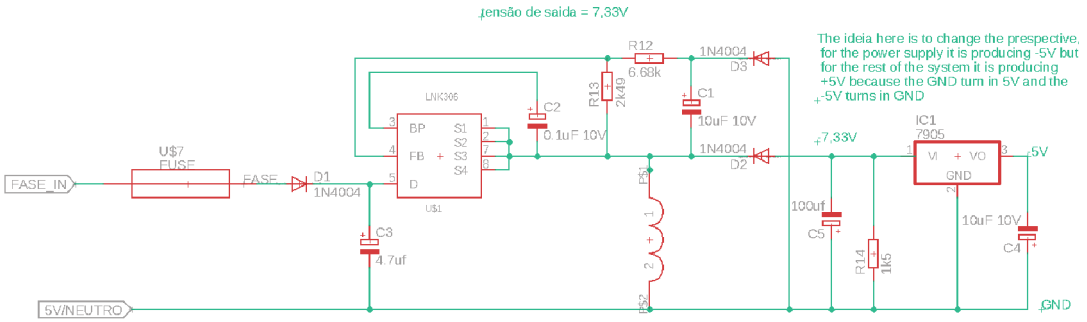


Fonte: O Autor (2018)

3.2.3.2 *Fonte de alimentação*

O sistema possui uma fonte de alimentação que fornece 5V e até 360mA. Essa é não isolada com uma arquitetura buck-boost e produz -6,4V. Após isso, é utilizado um regulador de tensão negativo para obter uma tensão de -5V com baixo ruído e que tem o terra na mesma referência que uma das tensões de alimentações, fato que é importante para o acionamento do Triac e a leitura de corrente. A Figura 15 demostra o circuito completo da fonte.

Figura 15 – Fonte de alimentação do dispositivo.



Fonte: O Autor (2018)

A solução escolhida foi uma fonte não isolada com o objetivo de diminuir a área de placa e o custo do projeto. Como o dispositivo foi pensado para ser instalado dentro de caixas de tomadas, é necessário que ele seja pequeno e o custo é um requisito do projeto como já destacado. Destaca-se também que o local de instalação é de difícil acesso o que diminui os riscos associados a fontes não isoladas.

O controlador escolhido para a fonte foi o LNK306 da Power Integrations esse possui um MOSFET integrado diminuindo o número de componentes necessários e possibilitando um projeto compacto. O controlador também permite a escolha da tensão de saida através do circuito de retorno para o circuito de retorno escolhido os resistores podem ser calculados através da formula 3.1, onde $\mathrm{V_{FB}} = 1,65\mathrm{V}$ e $\mathrm{I_{FB}} = 49\mu\mathrm{A}$ (POWER INTEGRATIONS, 2014).

$$R_{FB} = \frac{V_o - V_{FB}}{\frac{V_{FB}}{R_{BIAS}} + I_{FB}} \tag{3.1}$$

Os resistores de retorno e bias utilizados no projeto foram de 6,68k e 2,49k respectivamente, dessa forma a tensão de saida é de 6,4V.

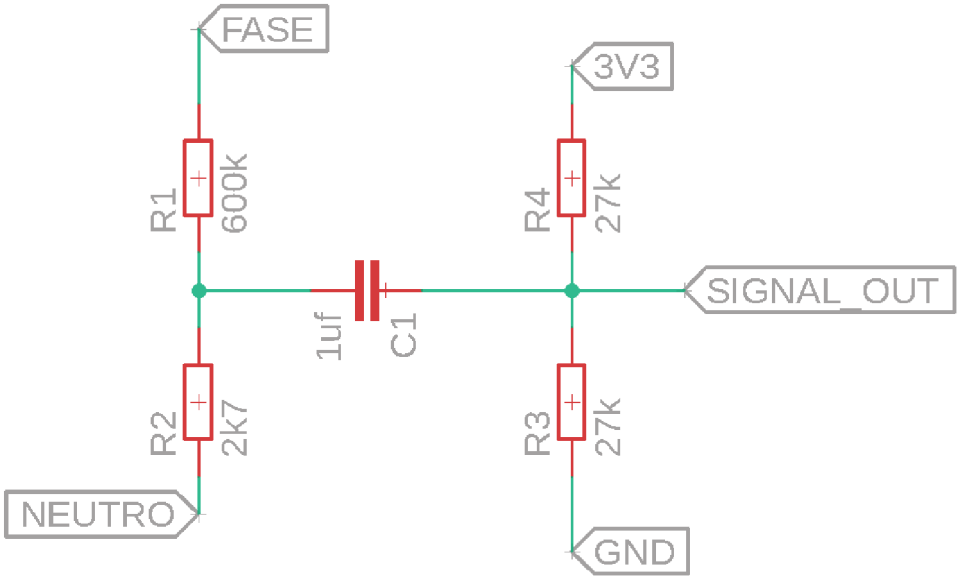
3.2.3.3 Monitoramento da corrente elétrica

O monitoramento da corrente elétrica no dispositivo ocorre através do monitoramento da tensão em um resistor Shunt, como esses sinais são pequenos foi utilizado o circuito integrado INA219 para amplificar e ler o sinal e melhorar a resolução de leitura do sistema. Esse dispositivo se comunica através de I2C com o microcontrolador enviando o valor de tensão lido nos terminais do resistor, o qual é então convertido em corrente pelo microcontrolador.

3.2.3.4 Monitoramento de tensão

O monitoramento da tensão da rede foi realizado com o circuito representado na Figura 16. Esse circuito possui um divisor de tensão que diminui as tensões de 622$V_{peak-peak}$ para um sinal de 3,3$V_{peak-peak}$ e após isso passa por um filtro passa alta que coloca um deslocamento DC de 1,6V permitindo assim que o microcontrolador possa ler tanto o subciclo positivo como o subciclo negativo da tensão.

Figura 16 – Circuito meditor de tensão.

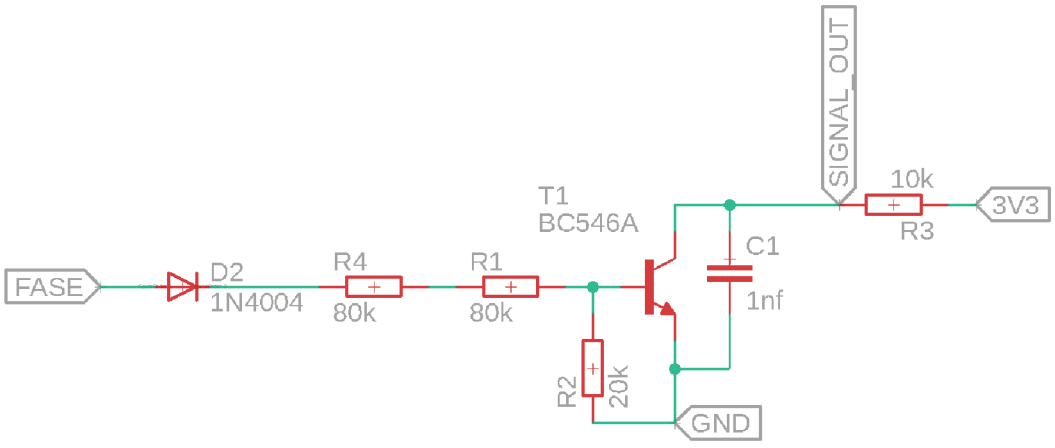


Fonte: O Autor (2018)

3.2.3.5 Circuito de Zero Cross

O sistema possui ainda um circuito de zero cross, seu objetivo é auxiliar no acionamento do Triac, o circuito pode ser visto na Figura 17. Além de permitir a dimerização através do Triac, o zero cross é utilizado para o software identificar a frequência da rede e ter a informação dos inícios e fim dos ciclos permitindo cálculos do valor RMS.

Figura 17 – Circuito de Zero Cross.

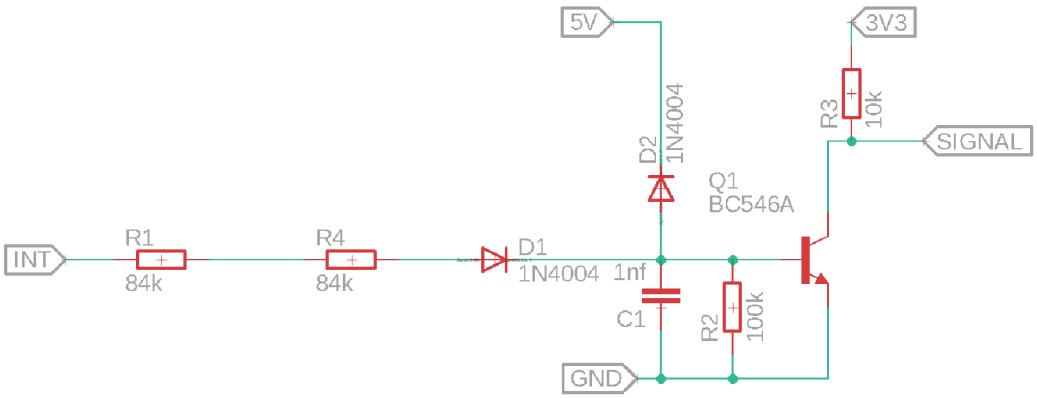


Fonte: O Autor (2018)

3.2.3.6 Circuito de entrada AC

O dispositivo possui também um circuito de entrada AC esse circuito possui a função de identificar se há tensão AC na entrada dele. Isso permite que o sistema possa se conectar a interruptores com apenas um fio, mantendo a segurança que independentemente se o sinal colocado na entrada for fase ou neutro ele ainda identificará corretamente o estado do interruptor. Além disso o circuito também foi utilizado como feedback para a identificação de falhas no Triac. A Figura 18 apresenta o circuito.

Figura 18 – Circuito de entrada AC.



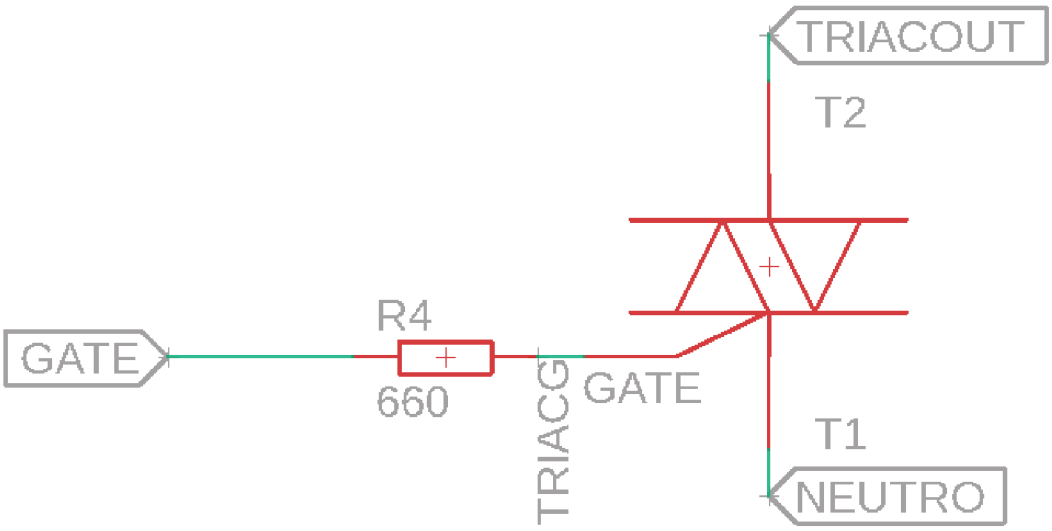
Fonte: O Autor (2018)

Esse circuito produz um sinal constante de 3,3V quando está flutuando, um sinal de 0V quando conectado ao neutro e uma onda quadrada com a frequência da rede e amplitude 3,3V quando conectado ao fase.

3.2.3.7 Circuito Atuador

O dispositivo possui ainda o circuito atuador que efetivamente controlará a carga. Esse circuito é composto de um Triac e um resistor e pode ser visto na Figura 19.

Figura 19 – Circuito Atuador.



Fonte: O Autor (2018)

Destaca-se que nesse circuito o gatilho do Triac é conectado diretamente no microcontrolador para tornar isso possível foi escolhido um “Logic Level Triac” que podem ser controlados diretamente pelo microcontrolador por possuírem gates mais sensíveis que necessitam de corrente menor para serem ativados (STMICROELECTRONICS, 2008). O Triac utilizado foi o BT137s, esse Triac possui uma corrente de gatilho máxima de 10 mA para os quadrantes de 1 a 3 e de 25 mA para o quadrante 4. (NXP, 2014)

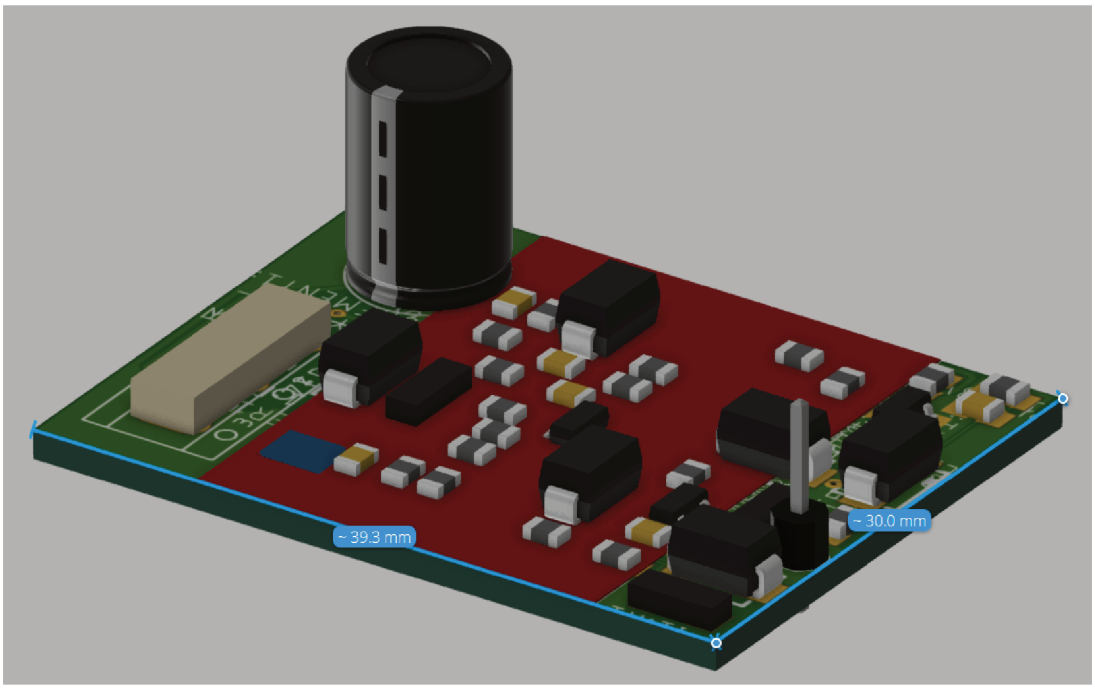
A escolha da arquitetura da fonte influência em quais quadrantes o Triac trabalhará. Como no projeto o Terminal 1 do Triac foi conectado ao neutro, com a arquitetura Buck-Boost o terra do microcontrolador terá uma tensão negativa em relação ao neutro. O que faz que a corrente no gatilho seja sempre negativa e o Triac opere nos quadrantes 2 e 3 os quais possuem menor corrente, portanto a escolha da fonte como Buck-Boost permite a conexão direta desse Triac ao Emote III.

3.2.3.8 Projeto da Placa de circuito impresso

O projeto foi desenvolvido para que seja instalado dentro de uma caixa de tomada, para isso é necessário que o dispositivo seja pequeno. Além disso, o dispositivo precisa se conectar ao Emote III através de seus conectores laterais. Para isso foi desenvolvido uma placa de circuito impresso que utiliza componentes SMD’s para minimizar o tamanho. A Figura 20 apresenta um modelo 3D do resultado esperado. Além do tamanho o desenvolvimento do layout da placa também considerou a dissipação de potência pelo Triac e otimizou a área de dissipação disponível para o mesmo através

da construção de um plano de cobre ao seu redor. Como pode ser visto na Figura 20 a placa final possui um tamanho de 39,3mm por 30mm. Um modelo 2D da placa pode ser encontrado na Anexo A.

Figura 20 – Modelo 3D da placa de circuito impresso desenvolvida.



Fonte: O Autor (2018)

3.2.3.9 Custo

Como um requisito de projeto a analise do custo é fundamental a Tabela 2 mostra o custo dos principais componentes do sistemas, esses custos foram cotados na MOUSER e a cotação utilizada para conversão de dólares para reais foi do dia 29/11/2018. A Bill of Materials (BOM) completa está disponível no Apêndice B.

Tabela 2 – Tabela de custos dos principais componentes do dispositivo.

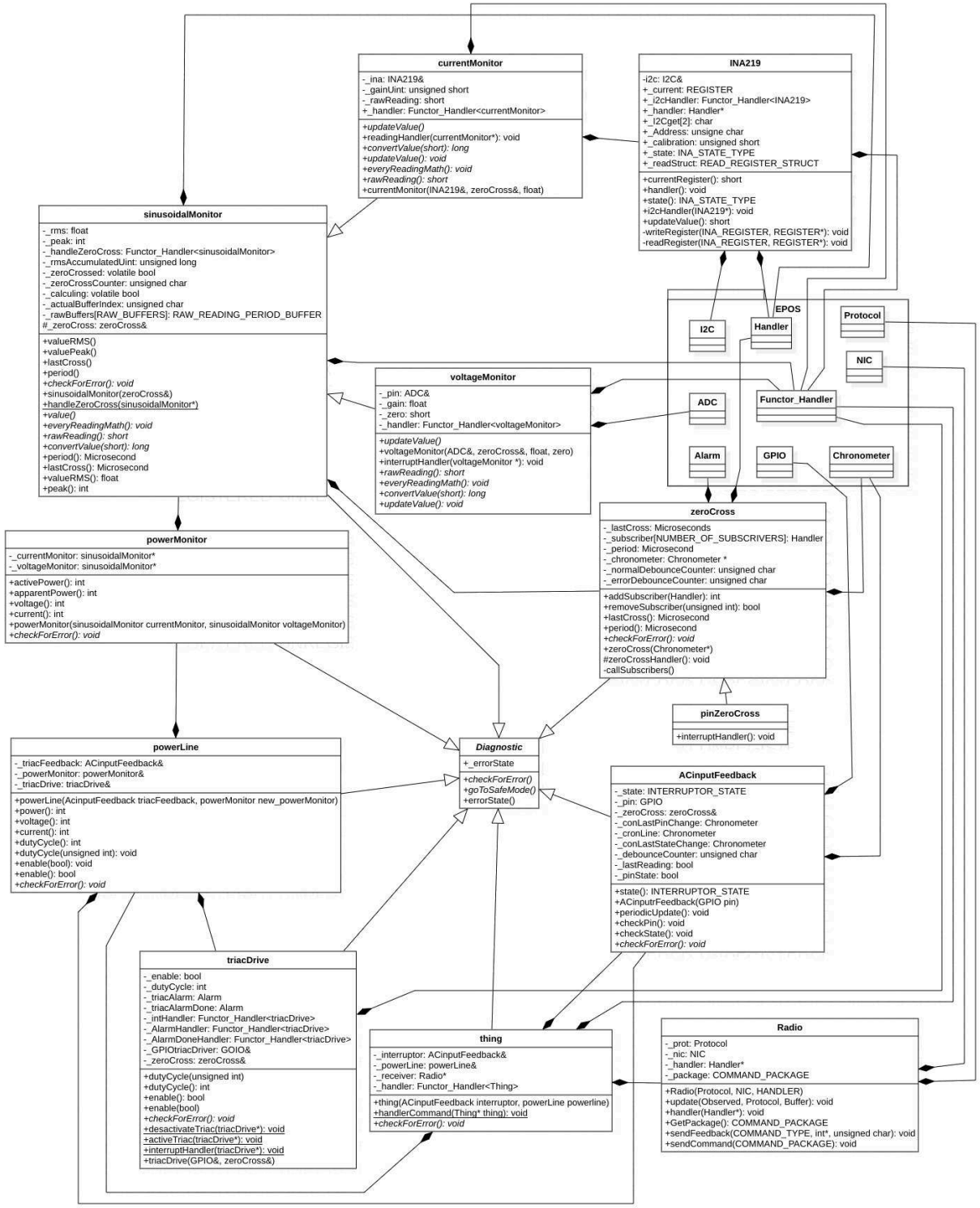
Componente	Valor (Dólares)
INA219	2,07
Capacitor de saída da fonte	1,24
Lnk306	1,77
Indutor	0,97
Triac	0,68
Conector	0,74
PCB	0,5
Total (com os demais componentes)	13,45 (51,78 reais)

Destaca-se que o sistema apresenta um valor competitivo em relação as opções de mercado, por exemplo, o Wi-Fi Smart Plug da Insignia que é vendido por $24,99 dólares na Best Buy ou o Wi-Fi Smart Plug da Amazon que é vendido por $24,99 dólares nela.

3.2.4 Software

O software foi desenvolvido segundo a orientação a objeto, para isso as funcionalidade foram separadas em diversas funções e módulos de software afim de prover reutilização de código e deixar o software modular e com fácil manutenção. O diagrama de classe implementado pode ser visto na Figura 21.

Figura 21 – Diagrama de classes do software implementado no dispositivo monitor de consumo elétrico.



Fonte: Autor (2018)

Nele podemos ver que foram criadas classes com o objetivo de abstrair as

funcionalidades do sistema.

3.2.4.1 Drivers de Hardware

Cada circuito de hardware teve seu funcionamento implementado por uma classe. A Tabela 3 relaciona as classes implementadas com os circuitos de hardware.

Tabela 3 – Abstrações implementadas pelas classes

Circuito	Classe
Circuito monitor de corrente elétrica	Classe INA219
Circuito medidor de tensão	Classe ADC provida pelo EPOS
Circuito de Zero Cross	Classe pinZeroCross
Circuito de entrada AC	Classe ACInputFeedback
Circuito Atuador	Classe triacDrive

Dessa forma, uma mudança no hardware que mantenha o mecanismo de funcionamento, como por exemplo, uma mudança do ganho da leitura de tensão implicaria na mudança apenas dessas classes.

3.2.4.2 Camada de abstração de Hardware

Algumas classes foram implementadas com o objetivo de prover uma camada de abstração de hardware, essas classes tem o objetivo de abstrair as diferentes implementações das classes drivers de hardware. Esse é o caso das classes, currentMonitor e voltageMonitor. A troca de um mecanismo de hardware por outro, como por exemplo a utilização de um sensor externo para a leitura de tensão causaria uma mudança nelas.

3.2.4.3 Classes de reuso

As classes sinusoidalMonitor, powerMonitor e zeroCross foram implementadas como forma de aumentar o reuso de código e prover algumas funcionalidades que são comuns a diversas partes do sistema. Como é o caso da sinusoidalMonitor que implementa os cálculos de valores RMS que são utilizados tanto para o currentMonitor como para o voltageMonitor. Essas classes só precisarão ser mudadas em caso de mudanças fundamentais do sistema como a retirada do circuito de zero cross ou a adaptação para o funcionamento em corrente continua.

3.2.4.4 Classe powerLine

Essa classe tem a função de agrupar e utilizar as abstrações providas pelas demais classes. Essa classe representa uma funcionalidade geral do sistema de controlar e medir a potência de uma carga, dessa forma se torna modular. Como no

caso de construir um hardware que possa controlar mais de uma carga, essa classe seria instanciada mais de uma vez.

3.2.4.5 Abstrações do protocolo

As abstrações implementadas pelo protocolo de comunicação são implementadas pelas classes thing e Radio sendo que a classe thing é a que implementa a API do dispositivo permitindo que o dispositivo receba comandos do gateway, execute e envie uma reposta adequada. A classe Radio é a que implementa a abstração do protocolo e a utilização das classes NIC e Protocol fornecidas pelo EPOS que abstraem de fato a a comunicação por ZigBee entre os Emote III.

3.2.4.6 Autodiagnóstico

A funcionalidade de autodiagnóstico foi implementada com cada classe sendo responsável por identificar os erros relevantes a ela, por exemplo, a classe zeroCross é capaz de identificar que nenhum sinal esta sendo encontrado, identificando portanto um erro nesse circuito. Para criar uma interface genérica e modular foi implementada a classes Diagnostic que tem a função de oferecer uma API para as funções de diagnostico no sistema.

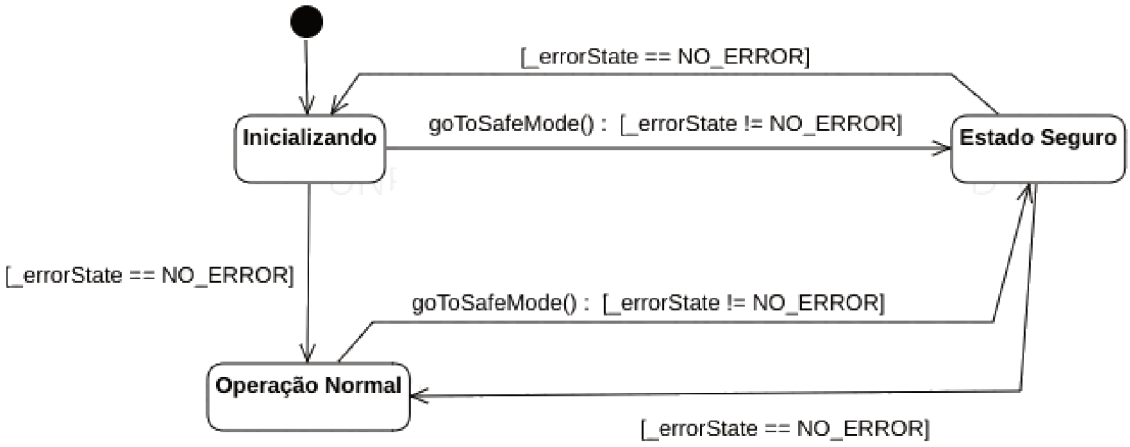
Uma das regras de negócios aplicado nessa construção determina que um objeto que é composto por outros deve herdar o estado de erro se uma falha foi identificada, dessa forma todo erro no dispositivo deve ser refletido no objeto thing.

Foram desenvolvidos quatro algoritmos de detecções de erros, todos tendo o objetivo de detectar erros de hardware. Quando qualquer erro é detectado o sistema vai para um estado seguro, a Tabela 4 relaciona a ação do sistema com os error identificados. A implementação do estado seguro responsabilidade individual de cada classe, no entanto, a classe Diagnostic prevê em sua interface essa transição. A Figura 22 representa o diagrama de estado da detecção de um erro.

Tabela 4 – Argumentos possíveis do comando SET

Erro	**Ação**
Zero Cross sem sinal	Medidas de tensão, corrente e potência respondem zero. Triac só ativa com 100% do duty cycle
Triac em curto	O microcontrolador para de acionar o Triac
Triac aberto	O microcontrolador para de acionar o Triac
Erro na leitura de corrente	Medidas de tensão, corrente e potência respondem zero.

Figura 22 – Diagrama de estado dos estados de erro do sistema.

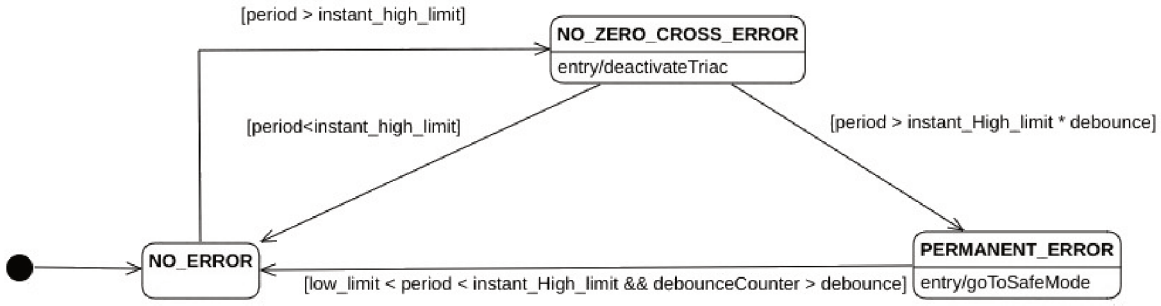


Fonte: Autor (2018)

3.2.4.6.1 Falha de Zero Cross

A falha de zero cross observa o sinal de saída do circuito para verificar quando não ha sinal de zero cross. Na Figura 23, é apresentado a dinâmica da detecção de falha no zero cross.

Figura 23 – Diagrama de estado do algoritmo de identificação de erro do zero cross.



Fonte: Autor (2018)

Como pode ser visto no diagrama de estado 23 o mecanismo de detecção de erro estabelece que, quando o período desde a ultíma borda do zero cross passar do 8,85 ms ele entrará em estado de erro temporário e o acionamento do Triac será desativado nesse estado. Caso a condição permaneça por um período de 26,55 ms o zero cross entrará em estado de erro permanente e o sistema entrará em um estado

seguro e só retornará depois de identificado 3 zero cross com períodos normais.

A escolha do limite de 8,85 ms é dado considerando metade do período para a frequência de 56,5Hz definida pela ANEEL (Agência Nacional de Energia Elétrica) como a frequência limite em situações extremas (AGêNCIA NACIONAL DE ENERGIA ELéTRICA, 2018). A escolha de 3 bordas de zero cross foi definida como forma para prevenir a entrada em erro quando há apenas um ciclo de energia com frequência não adequada e pode ser futuramente estudado para a escolha de um valor que apresenta melhor resultado na aplicação, vale destacar que o valor de 26,55 ms é a multiplicação do período de $8,85 \cdot 3$ sendo este o tempo máximo para conter 3 bordas de zero cross dentro da frequência esperada.

Analisando o circuito de zero cross que está representado na Figura 17, pode-se ver que essa falha pode ser causada pela falha do transistor T1 tanto aberta, como fechada, assim como pela falha dos resistores R1 e R4 e do diodo D2 em modo de falha aberta ou a falha dos componentes R2 e C1 com modo de falha em curto. Como o zero cross é um circuito importante que é utilizado tanto para o acionamento do Triac como nos cálculos de RMS a identificação dessa falha é de extrema importância para o funcionamento do dispositivo.

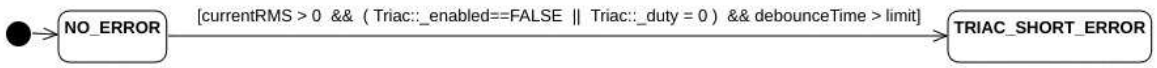
No caso de uma falha ser detectada tanto a dimerização do Triac como os cálculos de valores RMS e potência não serão mais confiáveis e portanto o sistema responderá com zero para a quantidade de corrente, a medição de tensão e a potência. O Triac também entra em um modo seguro, no entanto como a única funcionalidade que de fato foi prejudicada no funcionamento do Triac é a dimerização. Quando o erro do zero cross estiver ativo a dimerização será desativada sendo que o sistema só ligará a carga quando o duty cycle requisitado for de 100%;

3.2.4.6.2 Falha de Triac em curto circuito

Um dos modos de falha de um Triac é em curto, embora esse modo de falha só represente 10% dos casos (MEELDIJK, 1995). A identificação dessa falha é importante pois o dispositivo perde a funcionalidade de controle da carga. Além de ser um estado perigoso, principalmente se o funcionamento adequando do sistema sendo controlado depende do correto controle de potência pelo dispositivo. Por isso, foi desenvolvido um algoritmo de identificação de falha para quando o Triac se encontra ativado mesmo quando o microcontrolador não o ativou.

Para a detecção dessa falha o sistema usa do monitoramento de corrente caso a corrente na carga seja maior do que zero e o sinal no gatilho do Triac esteja desligado um estado de erro é ativado. Conforme pode ser visto na Figura 24 há um tempo de debounce que é necessário para evitar que o sistema identifique um erro quando o sinal do Triac foi desativado porém a medição de corrente ainda esta considerando valores lidos antes disso.

Figura 24 – Diagrama de estado do algoritmo de identificação de erro de Triac em curto.



Fonte: Autor (2018)

Quando uma falha de Triac em curto é detectada o sistema entrará em um estado seguro onde ele parará de tentar acionar o Triac e enviará um sinal de erro ao Gateway. Como a falha de Triac em curto é uma falha irreversível quando essa for identificada o sistema para de verificar a condição de erro até que o dispositivo seja reiniciado.

3.2.4.6.3 Falha de Triac Aberto

O modo de falha mais comum de um Triac é com ele aberto representando 90% dos casos (MEELDIJK, 1995). Apesar de não tão perigoso como o caso do Triac em curto esse modo de falha retira a capacidade do dispositivo de controlar a carga e, por ser, o mais comum a sua identificação é importante.

A abordagem para a identificação dessa falha difere da falha de Triac em curto pois a condição em que se analisa a corrente e o sinal do gatilho no Triac. A qual levaria a um estado de falha quando a corrente é igual a zero e o sinal do gatilho está ligado é a mesma condição de quando não há carga conectada ao dispositivo. Para ser possível separar essas duas condições foi adicionado o circuito de entrada AC conectado na saída do Triac conforme ilustrado na Figura 14. Dessa forma, o algoritmo de detecção de falha pode separar as duas situações, pois na condição em que a carga não esta conectada ao dispositivo o circuito de entrada AC identificará o sinal como neutro enquanto se a carga estiver conectada e mesmo assim não possui corrente o circuito identificará o sinal como fase. A Figura 25 ilustra o diagrama de estado do algoritmo.

Figura 25 – Diagrama de estado do algoritmo de identificação de erro de Triac aberto.

Fonte: Autor (2018)

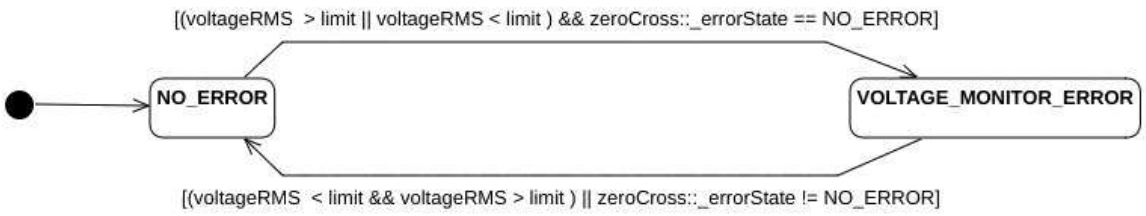
Quando detectado um erro de Triac aberto o sistema entra em um estado seguro onde ele para de acionar o Triac e envia um sinal de erro ao Gateway. Como essa falha é irreversível o sistema não tenta se recuperar do erro até que ele seja reiniciado.

3.2.4.6.4 Falha no circuito monitor de tensão

A identificação de falhas no circuito monitor de tensão pode ocorrer pela comparação entre o valor da tensão lida e a saída do circuito de zero cross. Quando há sinal no circuito de zero cross indica que ainda a tensão na rede e portanto se a leitura de tensão não estiver dentro dos limites de 191 Vrms e 233 Vrms, sendo esse os limites das tensões consideradas criticas para locais com tensão nominal de 220 Vrms, o algoritmo identifica a falha. (AGêNCIA NACIONAL DE ENERGIA ELéTRICA, 2018)

O algoritmo de detecção de falha para o circuito monitor de tensão verifica se as tensões RMS estão fora dessa faixa. No entanto, ele só é executado quando uma requisição do valor da tensão RMS é feita. Esse erro só é estabelecido se o zero cross não estiver em erro como forma de evitar que a identificação de um erro do circuito se confunda com uma queda de energia. A Figura 26 ilustra o diagrama de estado desse algoritmo.

Figura 26 – Diagrama de estado do algoritmo de identificação de erro no circuito monitor de tensão.



Fonte: Autor (2018)

Quando uma falha é identificada no circuito monitor de tensão, a medição de potência e de tensão não são mais confiáveis e o dispositivo retorna zero quando solicitado essas grandezas. Como a falha pode ter causas temporárias como a flutuação da tensão ou uma queda de energia, o sistema é capaz de se recuperar desse erro.

3.2.4.7 Threads

Foram criados três Thread no sistema sendo elas analogThread, diagnosticThread e a thread principal. A analogThread tem como objetivo adquirir as informações dos sensores de corrente e tensão e das entradas AC, essa thread tem um período de 1 ms. A diagnosticThread tem o objetivo a identificação de falhas no sistema e possui um período de 5 ms. A thread principal tem a função de imprimir pelo USB informações de status do sistema e possui um período de 1 segundo.

3.2.5 Projeto do gateway e da interface com o openHab

Para o cumprimentos de todos os requisitos, o sistema, além de adquirir os dados e ser capaz de atuar na carga, precisa de uma interface onde os usuários sejam capazes de acessar os dados e enviar comando ao dispositivo. Para isso, foi desenvolvido a solução de um gateway formado por um Emote III e um servidor com o openHab instalado.

3.2.5.1 Comunicação dos dispositivos

A comunicação do openHab com os dispositivos usuários é totalmente abstraída e não necessita de modificações pelo projeto.

A comunicação do openHab com o gateway se dá através do USB para isso um Emote III foi conectado ao computador servidor e o binding Serial do openHab foi utilizado para o envio e recebimento de dados. Para essa comunicação ser possível, foram criados dois itens do tipo String, Epos_TX e Epos_RX e esses foram ligados ao binding Serial. Dessa forma, quando uma mensagem é enviada pelo Emote III o item Epos_RX tem seu estado alterado para a string enviada pelo gateway e quando o openHab precisa enviar um comando o estado desses itens são alterados, o que executa uma regra que envia o comando pelo USB.

3.2.5.2 Protocolo de envio dos comandos

Para o envio de um comando foi desenvolvido um protocolo com o objetivo de transmitir comandos e informações entre o openHab e o dispostivo de monitoramento de consumo elétrico. O protocolo desenvolvido envia um pacote com o formato {Endereço, Comando, Argumentos, ... , Argumentos}, sendo que o mesmo pode ter quantos argumentos forem necessários separados por virgulas e cada um dos campos é composto de um numero inteiro de até 7 dígitos. Esse protocolo garante a possibilidade de vários dispositivos na mesma rede de sensores e a fácil expansão dos comandos.

3.2.5.3 Comandos implementados no dispositivo

Foram definidos os comandos GET e SET para a implementação no dispositivo permitindo assim a recuperação e a configuração de dados no sistema. Esse comandos recebem como primeiro argumento qual das grandezas do sistema deve ser alterada sendo seguido pelo numero de argumentos que for necessário para a alteração da propriedade do dispositivo.

Os argumentos implementados para o comando SET estão relacionados na Tabela 5.

Tabela 5 – Argumentos possíveis do comando SET

Argumento	**Número**
SET_TRIAC_DRIVE_DUTY_CYCLE	0
SET_TRIAC_DRIVE_ENABLED	1

Os argumentos implementados para o comando GET estão relacionados na Tabela 6.

Tabela 6 – Argumentos possíveis do comando GET

Argumento	**Número**	**Argumentos da resposta**
GET_TRIAC_DRIVE_DUTY_CYCLE	0	Duty cycle
GET_TRIAC_DRIVE_ENABLED	1	Triac enabled
GET_POWER_MONITOR_POWER	2	Potência
GET_INTERUPTOR_STATE	3	Estado interruptor
GET_VOLTAGE_MONITOR_VALUE	4	Tensão
GET_CURRENT_MONITOR_VALUE	5	Corrente
GET_ALL	6	Duty cycle, Triac enabled, Potência, Estado interruptor, Tensão, Corrente, Estado de erro
GET_ERROR	7	Estado de erro

3.2.5.4 Criação do dispositivo no openHab

Para a integração do dispositivo no openHab foi necessário a criação de itens, regras e sitemaps.

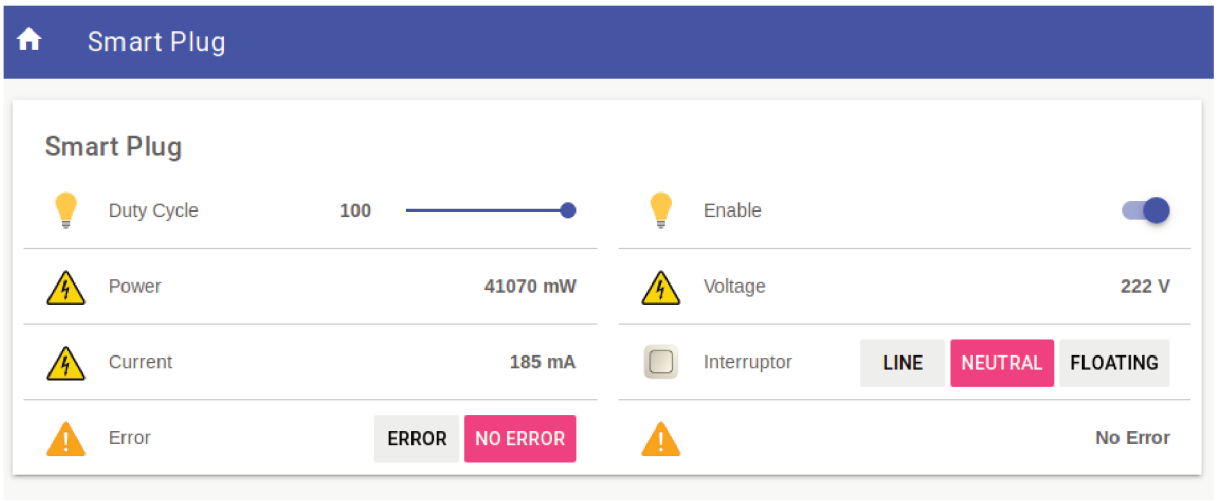
Os itens criados tem o objetivo de armazenar temporariamente as informações recebidas do dispositivo portanto foi necessário a criação de um item para cada dado que pode ser configurado ou informações que pode ser recebida do dispositivo, o arquivo de itens utilizado esta disponível no Anexo C.

Além disso foram necessários a criação de regras com o objetivo de abstrair o protocolo de comandos transformando assim um comando na interface do openHab em um comando para o dispositivo. O arquivo de regras utilizado está disponível no

Anexo D.

Também foi criado um sitemap para mostrar as informações ao usuário e receber os comandos dele. O arquivo do sitemap utilizado está disponível no Anexo E. A Figura 27 mostra a interface de usuário resultante dessas configurações.

Figura 27 – Interface de usuário criada no openHab.



Fonte: Autor (2018)

A Figura 27 apresenta os campos de informações disponíveis ao usuário, entre eles está dois campos de erro localizados na ultima linha. O campo esquerdo tem a função de apresentar se o sistema possui algum erro ou não. O campo direito tem a função de apresentar o código de erro identificado no sistema, a Tabela 7 apresenta os códigos implementados no dispositivo.

Tabela 7 – Códigos de erro do sistema.

Código	Erro
F2	Erro no Zero Cross, em debounce
F3	Erro confirmado no Zero Cross
F4	Erro no Triac, modo de falha em curto
F5	Erro no Triac, modo de falha aberto
F6	Erro na leitura de tensão

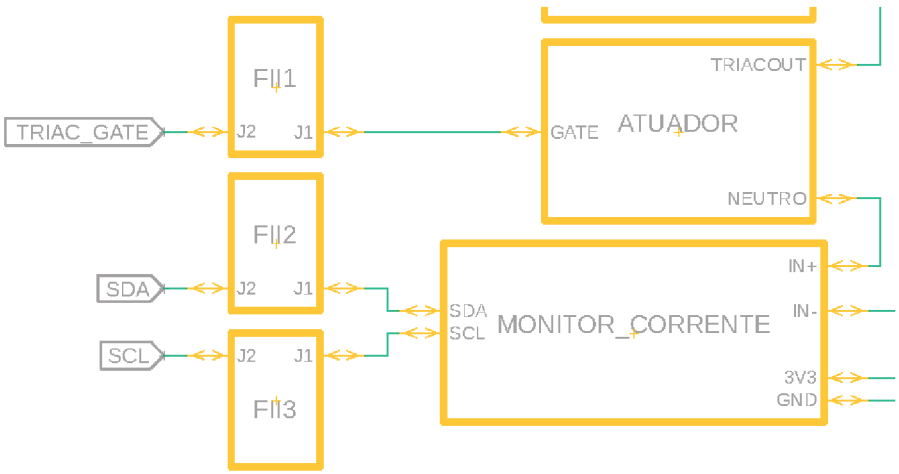
4 EXPERIMENTOS E RESULTADOS

Para a realização de experimentos e validação das soluções foram fabricadas duas placas, uma sendo uma versão modificada do dispositivo de monitoramento do consumo elétrico com pontos de injeção de falhas e uma placa para a injeção automática de falhas no sistema.

4.1 AJUSTES DO DISPOSITIVO DE MONITORAMENTO DO CONSUMO ELÉTRICO PARA INJEÇÃO DE FALHAS

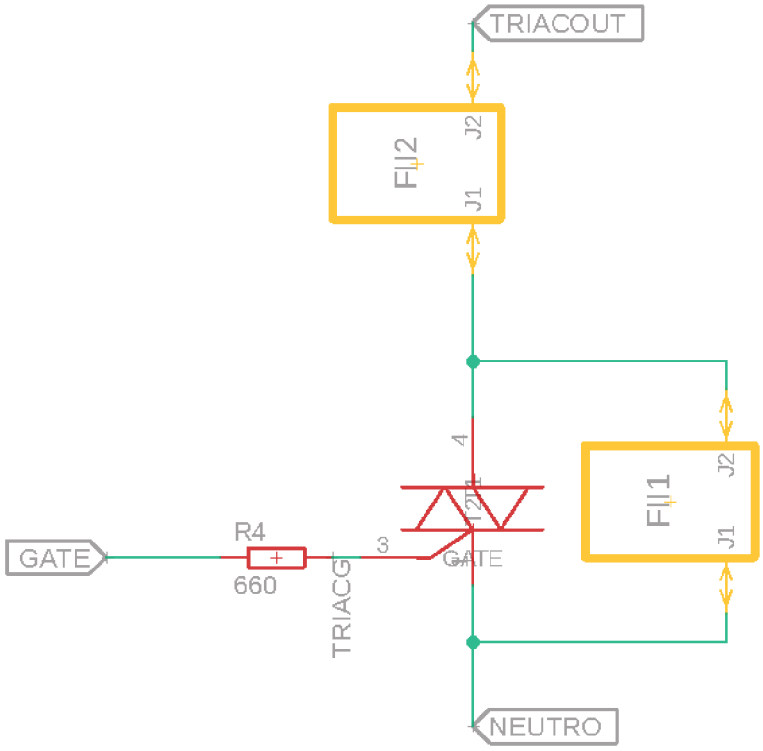
No dispositivo foram escolhidos nove pontos de injeção de falha, apesar de o dispositivo utilizar apenas quatro algoritmos de detecção a placa foi desenvolvida para que possa ser utilizada em estudos futuros. Sendo construída com nove pontos de injeção de falhas os quais podem ser vistos nas Figuras 28, 29, 30, 31, 32. Como pode ser visto nas Figuras os pontos de injeção de falhas foram nomeador como FII (Fault Injection Interface) e foram colocados em nove pontos, sendo esse no SDA e SCL do I2C no gatilho do Triac, no circuito de zero cross, no circuito de entrada AC, no circuito de medição de corrente e dois no circuito de acionamento, vale destacar que apesar de aparecerem apenas oito pontos de injeção de falhas no esquemático o circuito de entrada AC é utilizado duas vezes.

Figura 28 – Pontos de falhas no circuito principal.



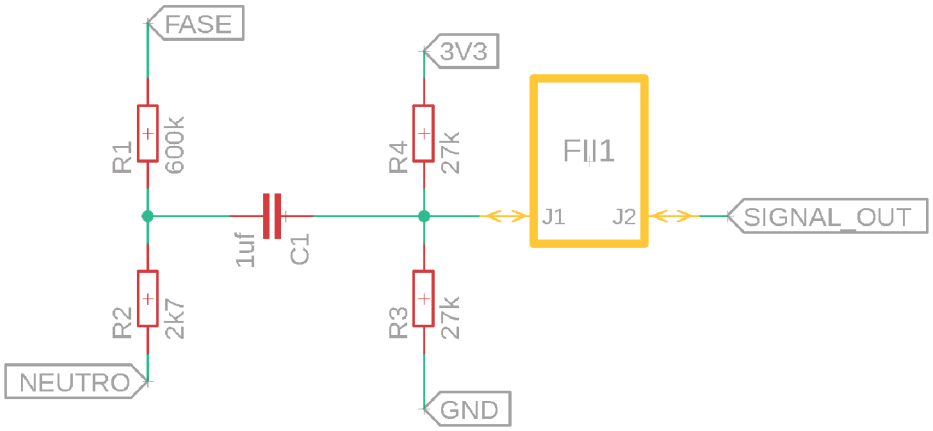
Fonte: Autor (2018)

Figura 29 – Pontos de falha no circuito de acionamento.



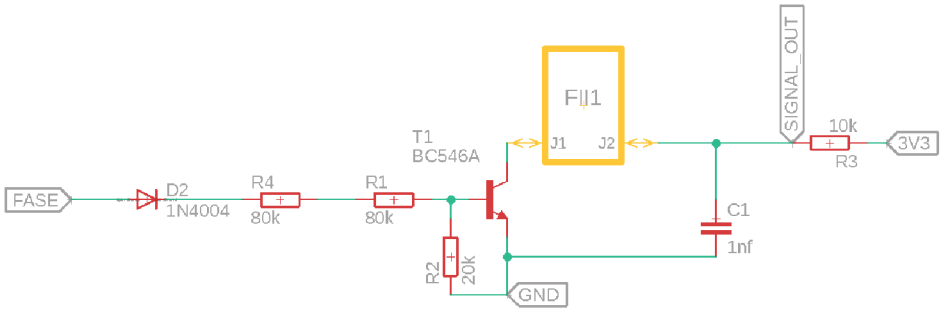
Fonte: Autor (2018)

Figura 30 – Ponto de falha no circuito monitor de tensão.



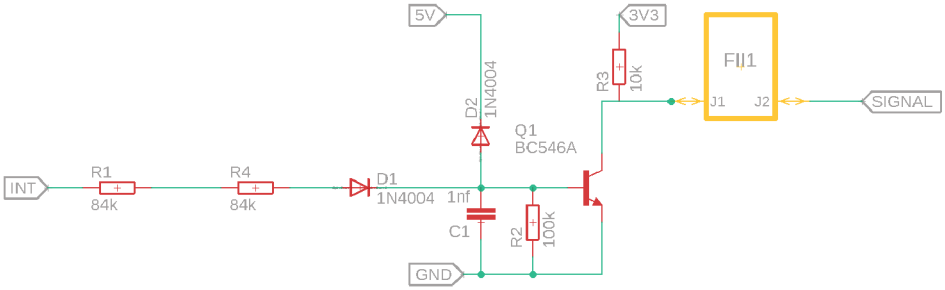
Fonte: Autor (2018)

Figura 31 – Ponto de falha no circuito de Zero Cross.



Fonte: Autor (2018)

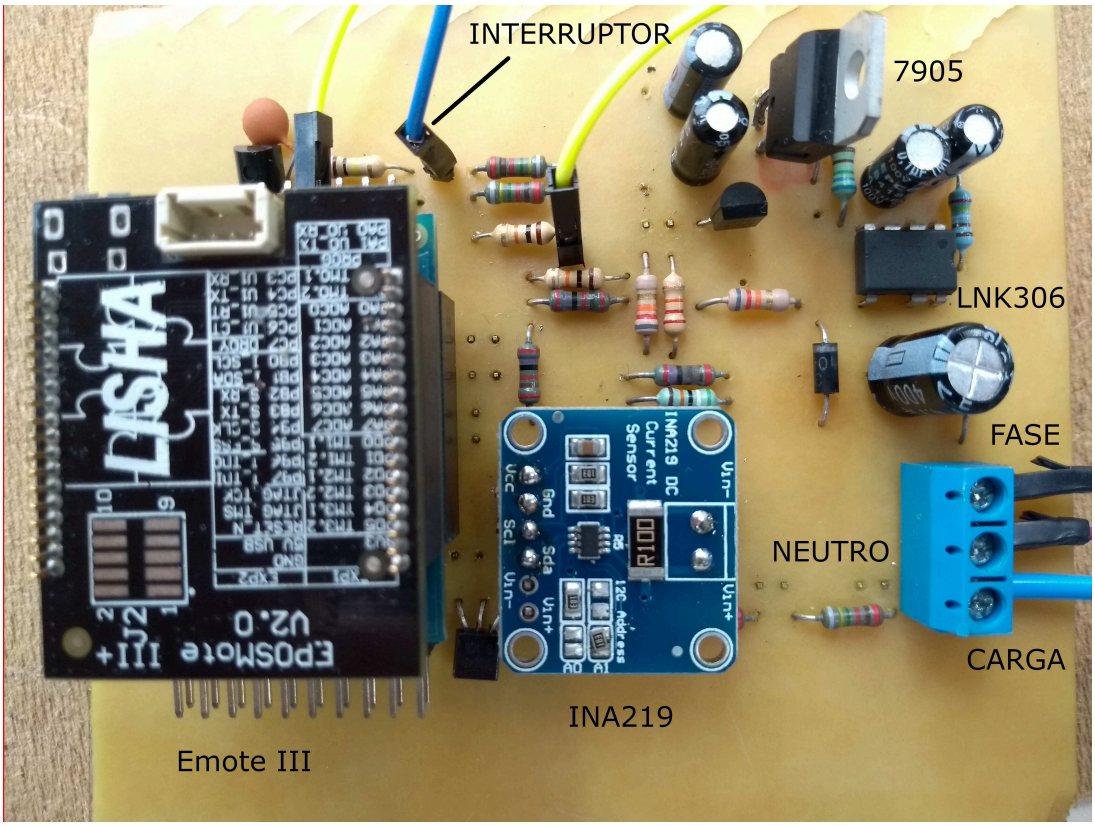
Figura 32 – Ponto de falha no circuito de entrada AC.



Fonte: Autor (2018)

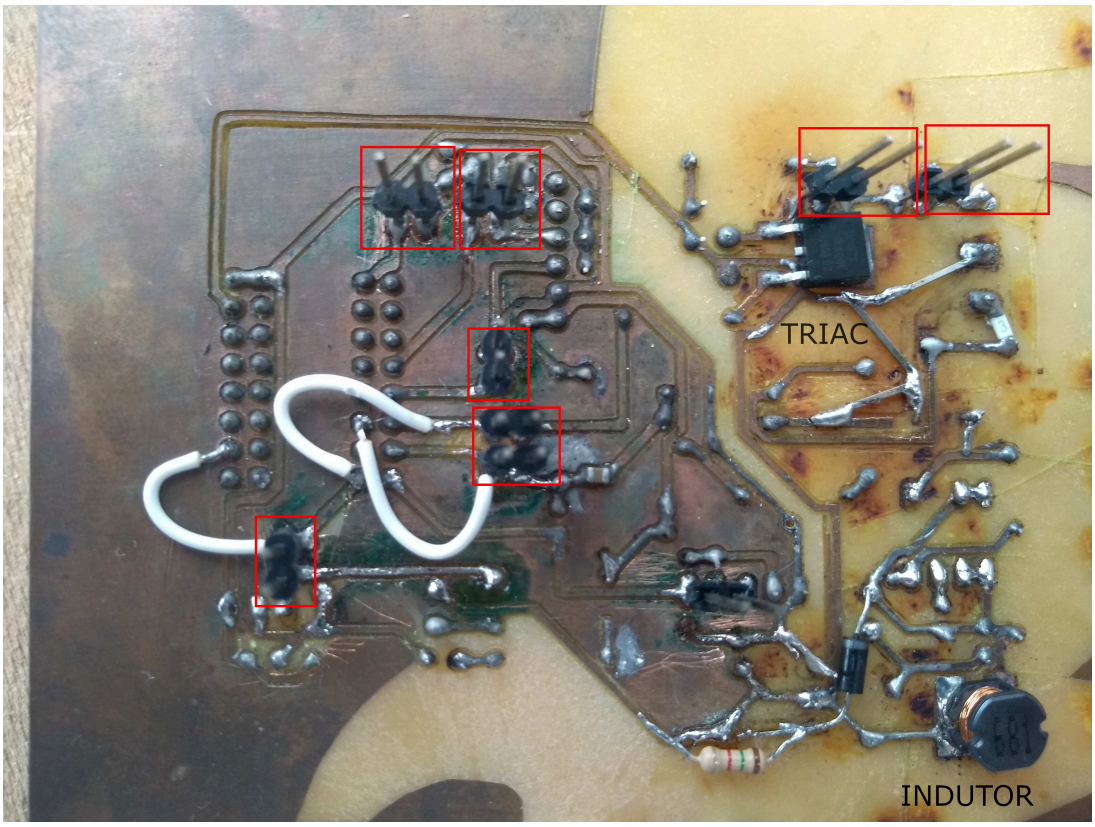
Esses pontos de injeção de falha foram projetados como um conector que interrompe a trilha em que foi instalada e expõe esses pontos de conexões permitindo que a falha seja injetada por um circuito externo. Na Figura 33 é apresentada a parte de cima placa finalizada e, Figura 34, a parte de baixo dela. Nessa Figura os conectores colocados para a injeção de falha estão destacados em vermelho.

Figura 33 – Parte de cima do dispositivo de monitoramento de consumo elétrico.



Fonte: Autor (2018)

Figura 34 – Parte de baixo do dispositivo de monitoramento de consumo elétrico.

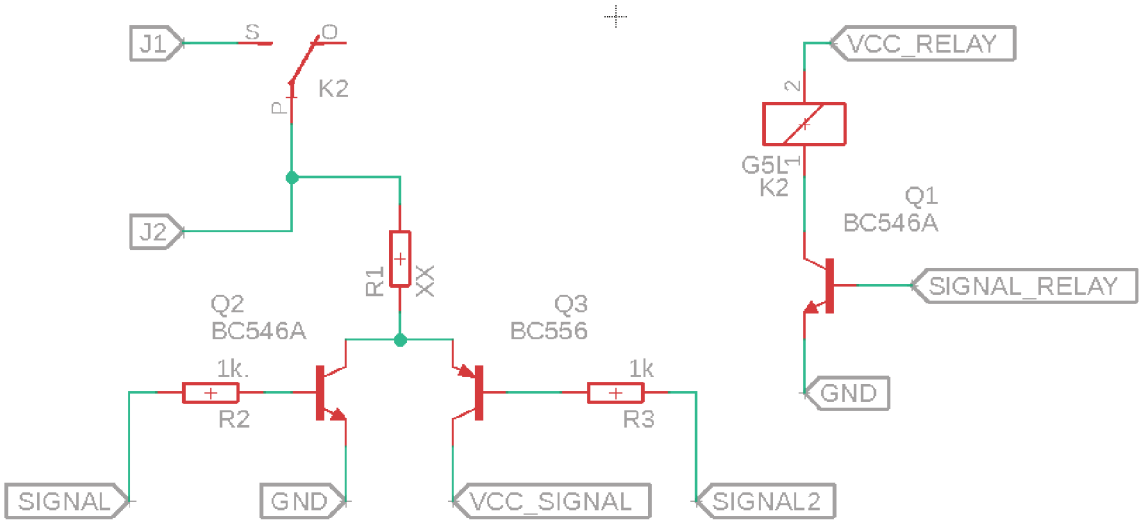


Fonte: Autor (2018)

4.2 DISPOSITIVO DE INJEÇÃO AUTOMÁTICA DE FALHAS

O dispositivo de injeção de falha foi projetado com dois mecanismo de injeção de falhas. Um para simular falhas de hardware permanentes tanto no modo de falha em curto como no modo de falha aberto. Outro para simular falhas transitórias que podem ocorrer devido a ruídos ou ao funcionamento incorreto de algum componente. A Figura 35 mostra o circuito implementado para a injeção de falha.

Figura 35 – Circuito de injeção de falhas.



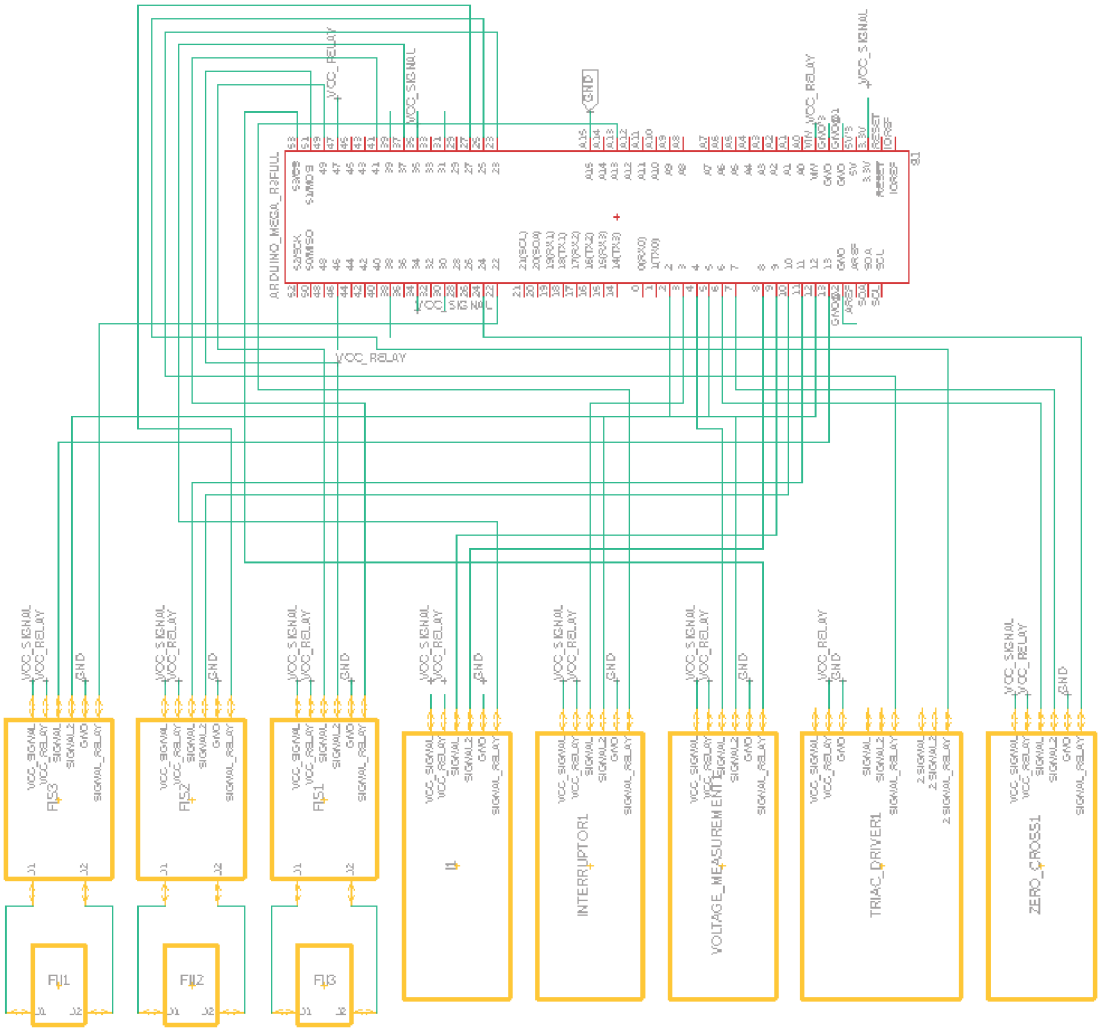
Fonte: Autor (2018)

Na Figura 35, pode-se observar que o circuito é formado por um relê, o qual pode conectar ou desconectar os pinos expostos no dispositivo alvo da falha. Esse é o mecanismo de injeção permanente de falha e pode simular falhas com o modo de falha aberto ou em curto, Figura 29. Onde a interface FII2 injeta falhas do modo aberto e a interface FII1 injeta falha de curto no Triac.

O mecanismo de injeção de falhas transitória que é composto de um resistor em série de um circuito push-pull compostos de dois transistores, Figura 35. Essa topologia foi permite que sejam injetadas falhas com sinal lógico alto e com sinal lógico baixo. Além disso, a escolha do resistor em série permite decidir qual corrente será retirada ou enviada ao sistema alvo, podendo assim escolher qual a amplitude do ruído aplicado. Esse mecanismo não foi implementado para os circuitos que possuem interface direta com alta tensão.

O dispositivo de injeção automática de falhas conta também com um controlador de quais as falhas serão injetadas. Para isso foi escolhida a inclusão de um Arduino Mega por esse possuir doze canais PWM, que podem ser utilizados no circuito de push-pull para injetar ruídos. A Figura 36 mostra as conexões entre o Arduino e os circuitos de falha.

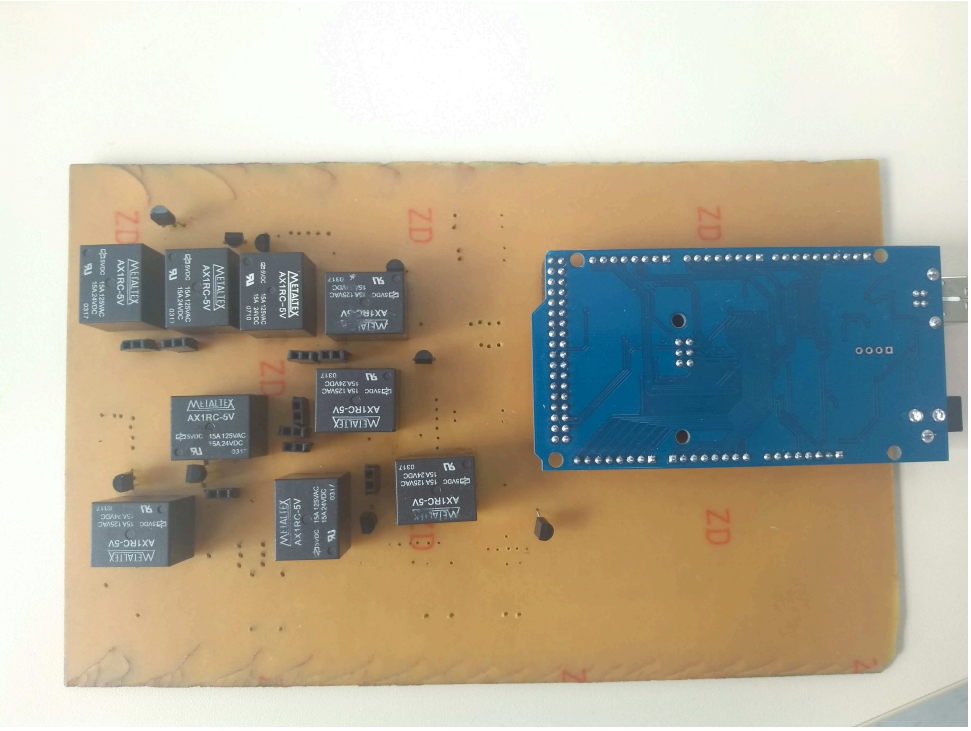
Figura 36 – Conexões entre os circuitos de falha e o Arduino.



Fonte: Autor (2018)

Na Figura 37 é apresentada a placa finalizada. Esse dispositivo de injeção automática de falhas foi projetado para que possa ser usado em estudos futuros, no entanto nos experimentos que serão apresentados apenas o mecanismo de injeção de falhas permanente de hardware foi utilizado.

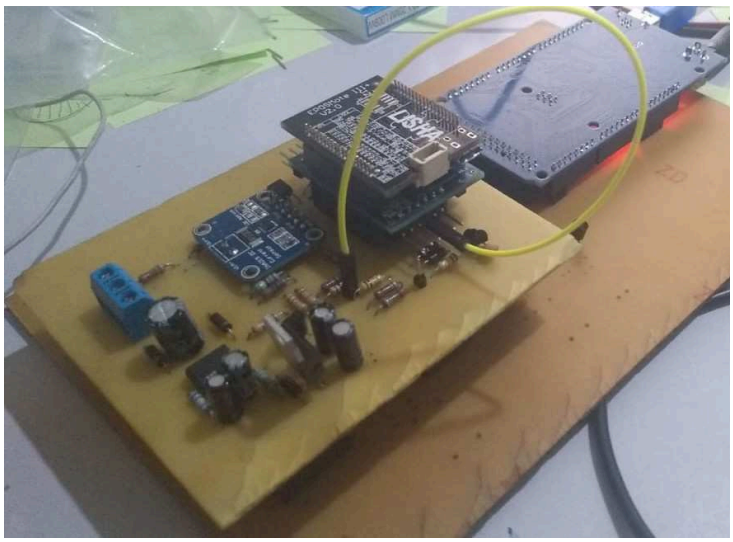
Figura 37 – Dispositivo de injeção de falha.



Fonte: Autor (2018)

A placa de injeção de falha e o protótipo modificado do dispositivo monitor de corrente foram projetados de forma compatível e com conectores alinhados, conforme ilustrado na Figura 38.

Figura 38 – Conexão entre os dispositivos.



Fonte: Autor (2018)

4.3 TESTE FUNCIONAL

Esse teste tem o objetivo de verificar se todas as funcionalidades do sistema operam de acordo com os requisitos. Nesse caso, não foi considerado o desempenho apresentado. Para isso, foi executado uma série de experimentos demonstrando o funcionamento de cada parte do sistema.

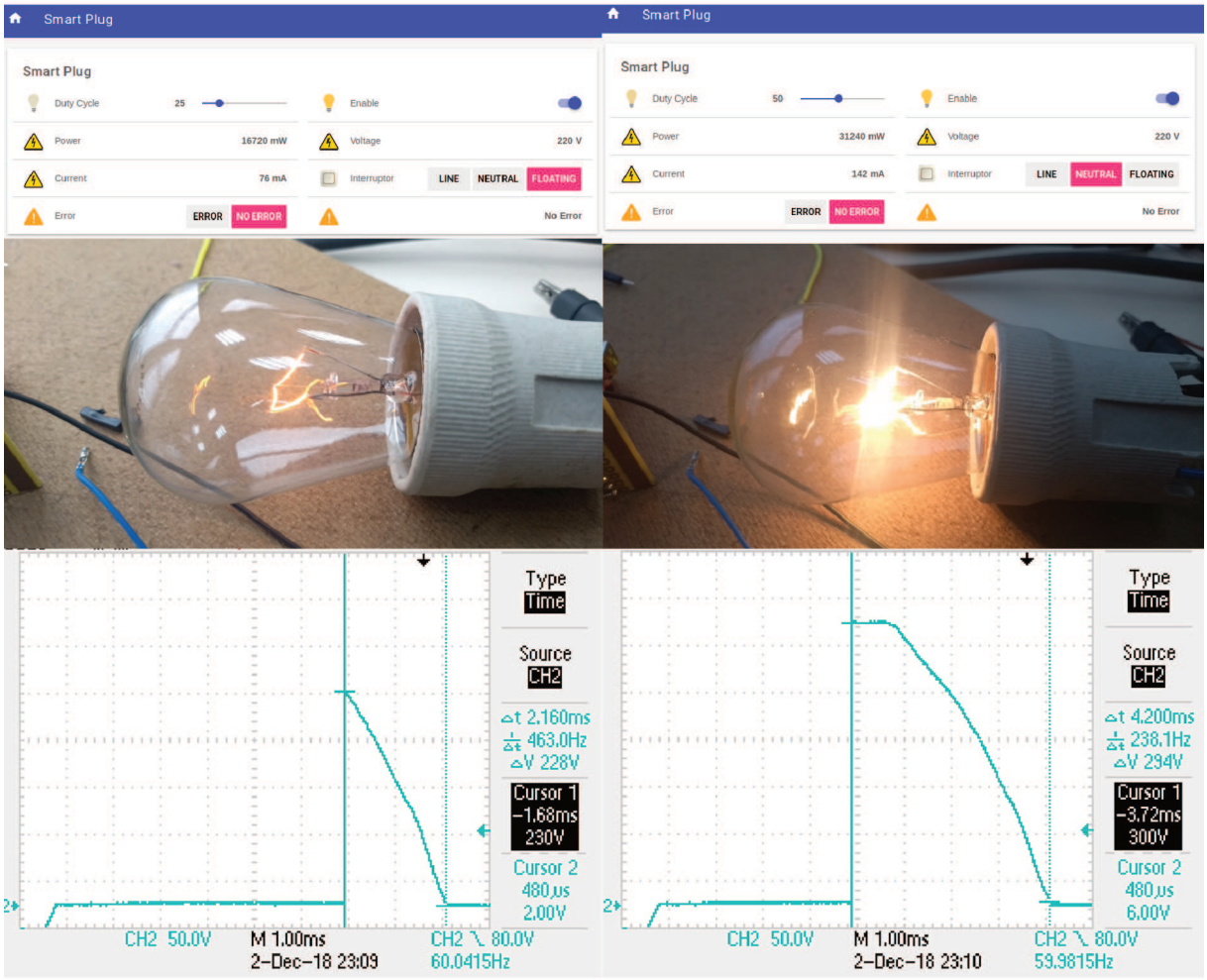
4.3.1 Controle da carga

A interface de usuário foi utilizado para testar o correto funcionamento da carga e para enviar diferentes comandos de funcionamento. A carga usada nos experimentos foi uma lâmpada de 40W. Foi também adquirida a forma de onda da tensão na carga com um osciloscópio com o objetivo de analisar a dimerização da carga. A Figura 39 mostra a intensidade da lâmpada, o formato de onda e as leituras apresentadas na interface de usuários para os duty cycles de 0%, como o fundo de escala do osciloscópio é menor que a tensão de pico a pico, nas imagens é mostrado apenas o semiciclo superior. A Figura 40 mostra os duty cycles de 25 e 50% e a Figura 41 para 75 e 100%. Destaca-se que nessas Figuras foi verificado o tempo da porção ativa, sendo que esses foram 2,16 ms, 4,2 ms, 6,24 ms e 8,24 ms para os duty cycles de 25,50,75 e 100% respectivamente.

Figura 39 – Resultados do experimento de varição do duty cycle da carga.

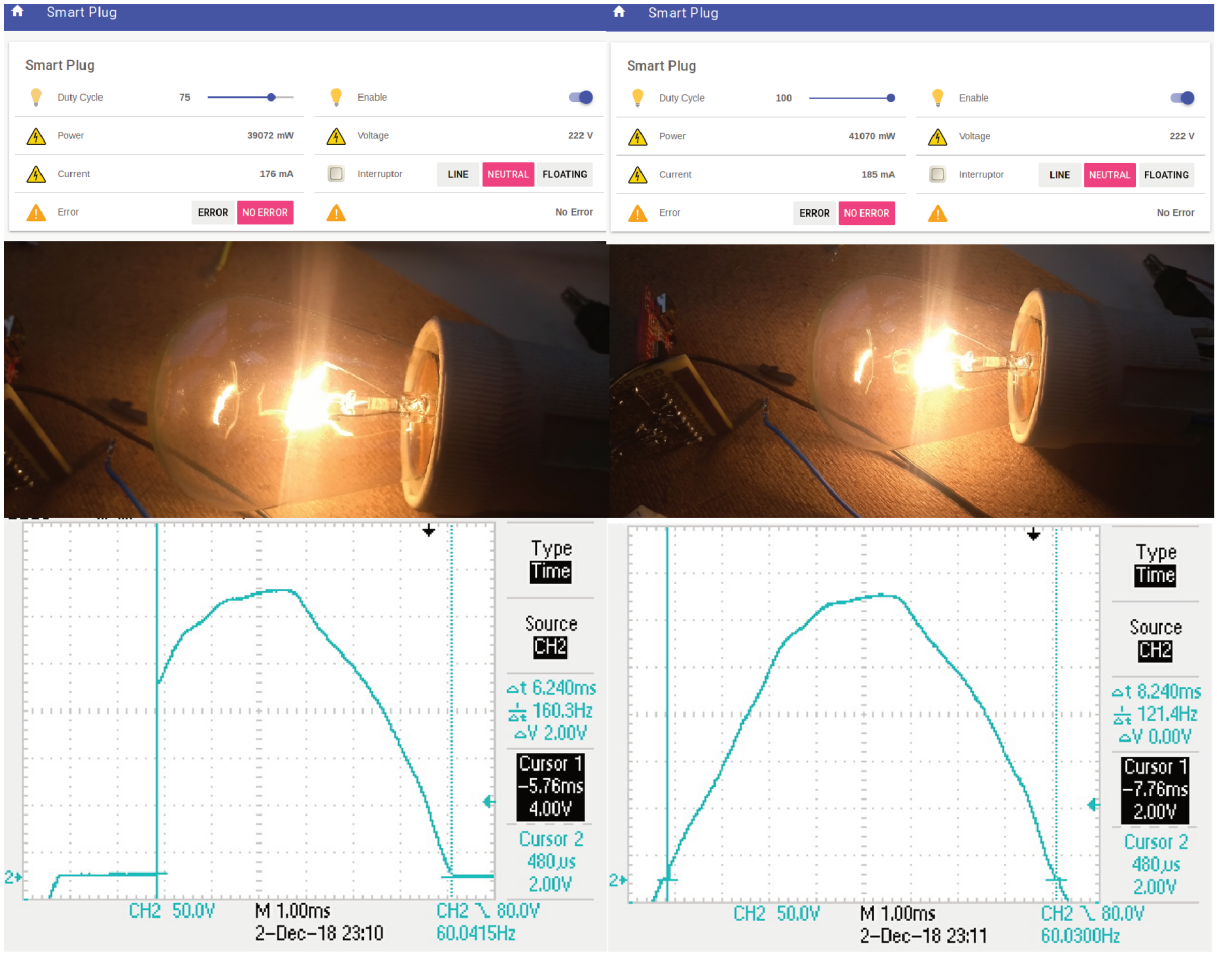
Fonte: Autor (2018)

Figura 40 – Resultados do experimento de varição do duty cycle da carga.



Fonte: Autor (2018)

Figura 41 – Resultados do experimento de varição do duty cycle da carga.



Fonte: Autor (2018)

Foi realizado também um teste para verificar o funcionamento do botão de enable disponibilizado na interface. Para isso, ele foi desabilitado e habilitado com uma configuração de 50% de duty cycle. Verificou-se que ao desabilitar o formato de onda apresentava 0V ao redor da carga e ao habilitar o formato de onda ao redor da carga possuía o mesmo formato do apresentado na Figura 40 com duty cycle de 50%.

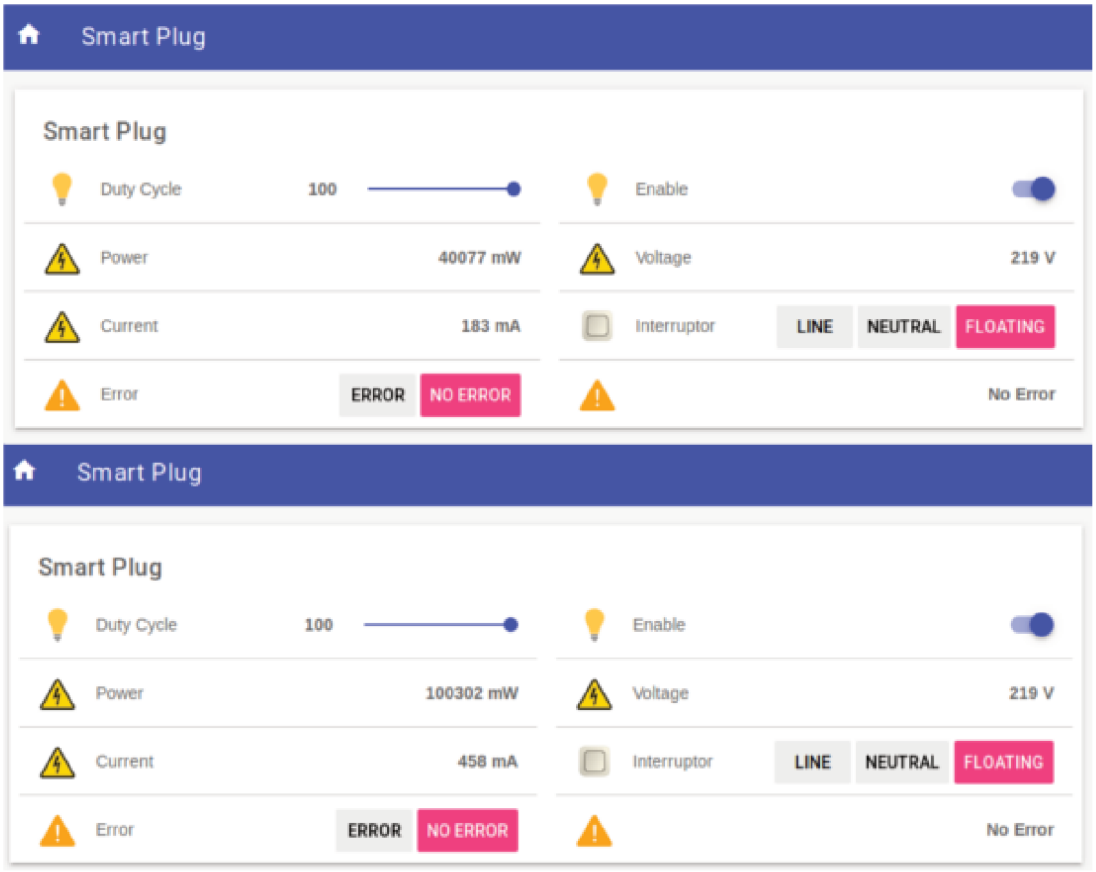
Esses testes permitiram demostrar que o controle da carga apresenta o comportamento esperado do dispositivo. Esses testes também demostram o funcionamento da interface e do sistema de comunicação entre o gateway e o dispositivo de monitoramento de consumo elétrico.

4.3.2 Aquisição de dados

Esse experimento teve o objetivo de verificar o funcionamento da leitura de tensão e corrente do sistema. Para isso, foram utilizados como cargas uma lampada de 40W e uma lampada de 100W. O objetivo é verificar se com 100% do duty cycle o

dispositivo lê a potência nominal de ambas as cargas. A Figura 42 mostra os valores informados na interface de usuário com a carga de 40W e com a carga de 100W.

Figura 42 – Interface de usuário com carga de 40W nominal e 100W nominal.

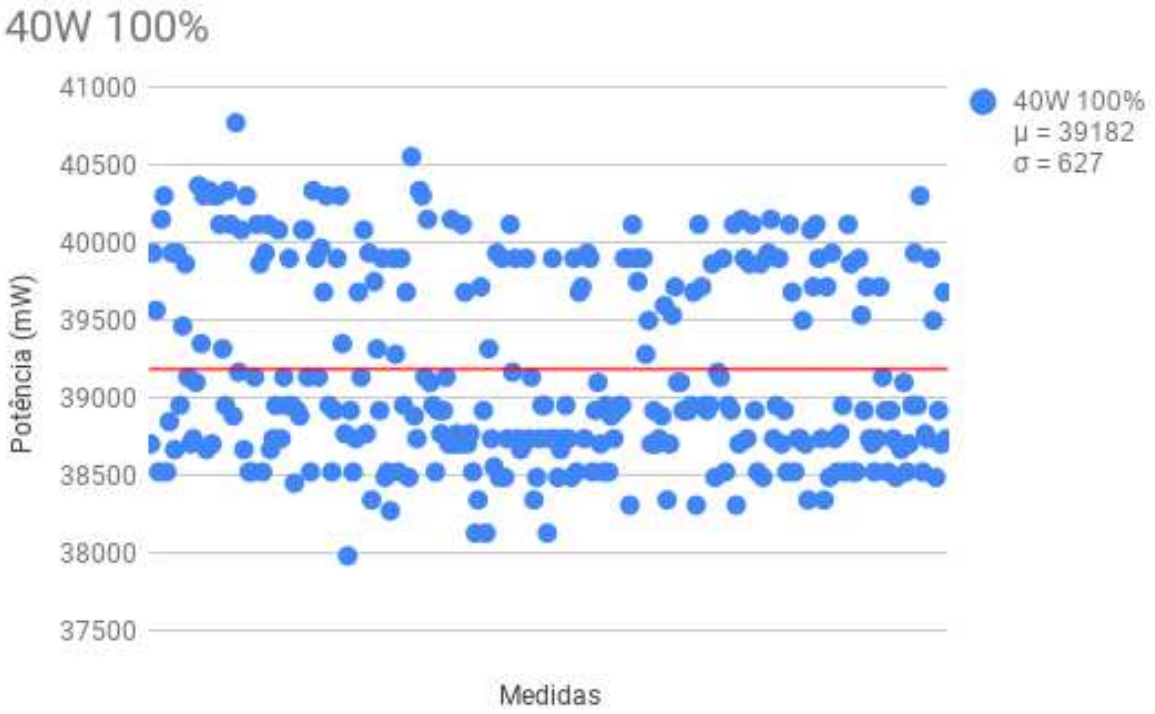


Fonte: Autor (2018)

4.4 TESTE DA MEDIÇÃO DE POTÊNCIA

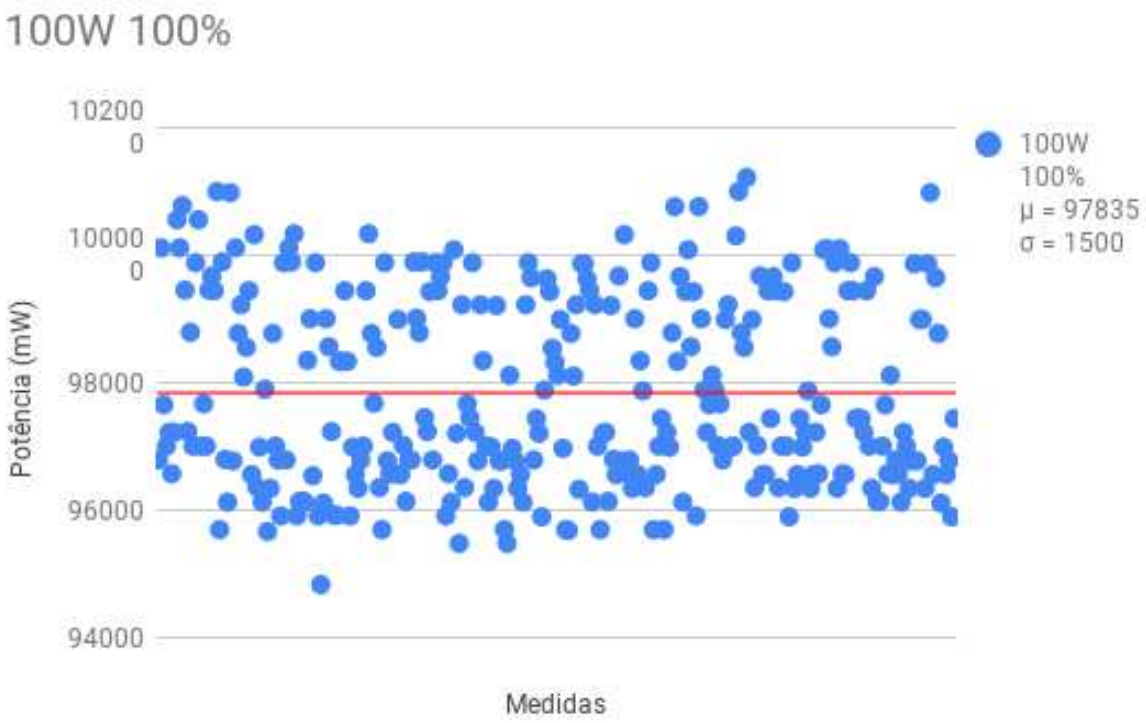
O objetivo desse teste é verificar o desempenho da medição de potência do dispositivo. Para isso, foram utilizadas duas cargas resistivas de 40W e 100W e adquirido dados da potência durante 5 minutos em intervalos de 1 segundo e duty cycle de 100%. A Figura 43 apresenta os resultados encontrados para a carga de 40W e a Figura 44 para a carga de 100W.

Figura 43 – Dispersão dos dados da medição de potência com uma carga de 40W e duty cycle de 100%.



Fonte: Autor (2018)

Figura 44 – Dispersão dos dados da medição de potência com uma carga de 100W e duty cycle de 100%.



Fonte: Autor (2018)

Conforme informado na seção 3.2.4.7, a Thread responsável pela aquisição de dados analógicos tem um período de 1 ms. Dessa forma, a aquisição da corrente ocorre com o mesmo período, sendo assim em cada subciclo a corrente será medida oito vezes no pior caso e nove vezes no melhor caso. Isso causa a variação encontrada na leitura de corrente. No entanto, para aumentar a precisão, o sistema armazena a leitura de quatro subciclos e faz o cálculo de RMS em um período maior que um subciclo.

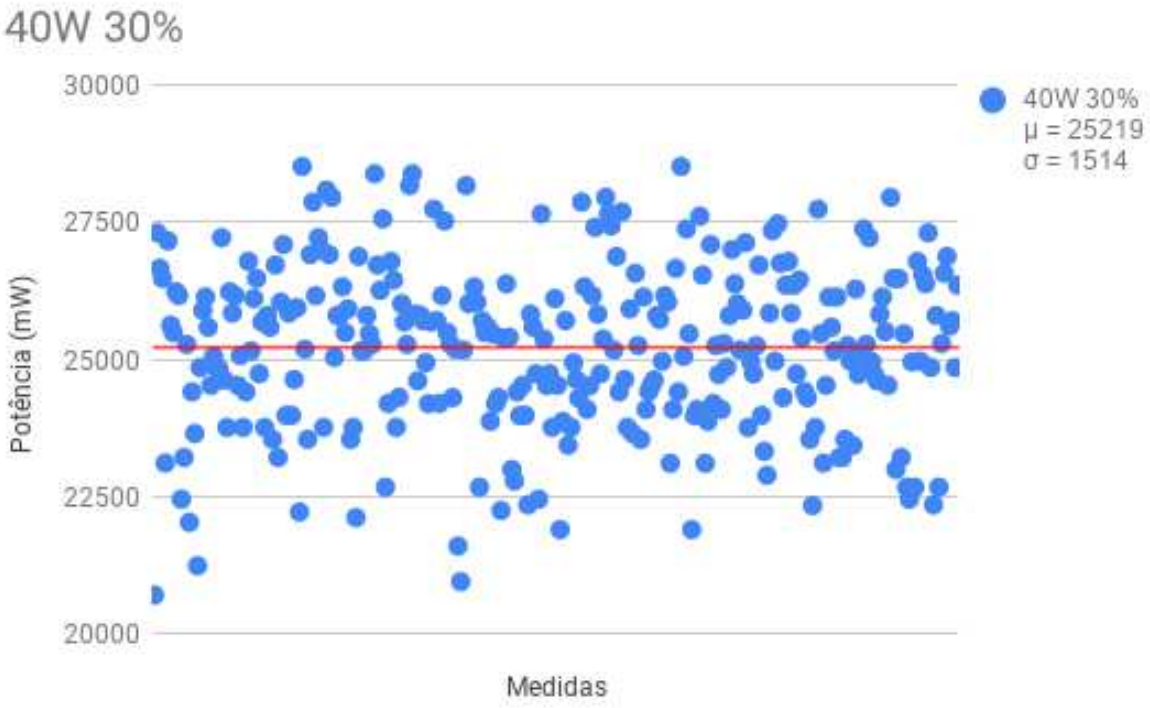
O valor de quatro subciclos foi escolhido como o menor valor que apresentou bons resultados empiricamente. No entanto, esse valor é um balanceamento entre a variação da leitura da potência, o consumo de memória RAM do microcontrolador e o atraso do valor lido. Trabalhos futuros podem ser realizados para encontrar o número de subciclo mais adequado ao sistema.

Outra característica desse sistema é possuir uma variação maior da potência lida para duty cycles menores. Isso acontece porque a forma de onda encontrada quando a carga é dimerizada possui uma parte em que a corrente será zero e depois a corrente apresentará o comportamento senoidal esperado, no entanto com o sistema medindo a corrente a cada 1 ms quando menor o duty cycle menor o número de

medidas diferentes de zero. Isso causa uma maior variação do valor medido.

Para verificar esse efeito foi feito o mesmo teste proposto acima com a carga de 40W no entanto com o duty cycle em 30%. A Figura 45 mostra o resultado desse teste. Comparando-se as Figuras 45 e 43 nota-se que o desvio padrão encontrado na medição da potência para o caso com 30% de duty cicle é maior que o encontrado para a mesma carga com 100% do duty cycle.

Figura 45 – Dispersão dos dados da medição de potência com uma carga de 40W e duty cycle de 30%.



Fonte: Autor (2018)

4.5 TESTE DE INJEÇÃO DE FALHAS

O objetivo do teste de injeção de falhas é verificar se o sistema é capaz de identificar as falhas. Para isso foi usada a placa de injeção de falhas que causam distúrbios no dispositivo e consultado se o sistema reporta o erro ao gateway.

Pra cada falha foram realizados dois teste da injeção de falha. O primeiro é a injeção da falha antes de o sistema iniciar para verificar se o dispositivo é capaz de identificá-la. A segunda maneira foi a injeção periódica da falha em intervalos de 30 segundos, para as falhas em que o sistema não consegue se recuperar essa injeção foi realizada apenas uma vez.

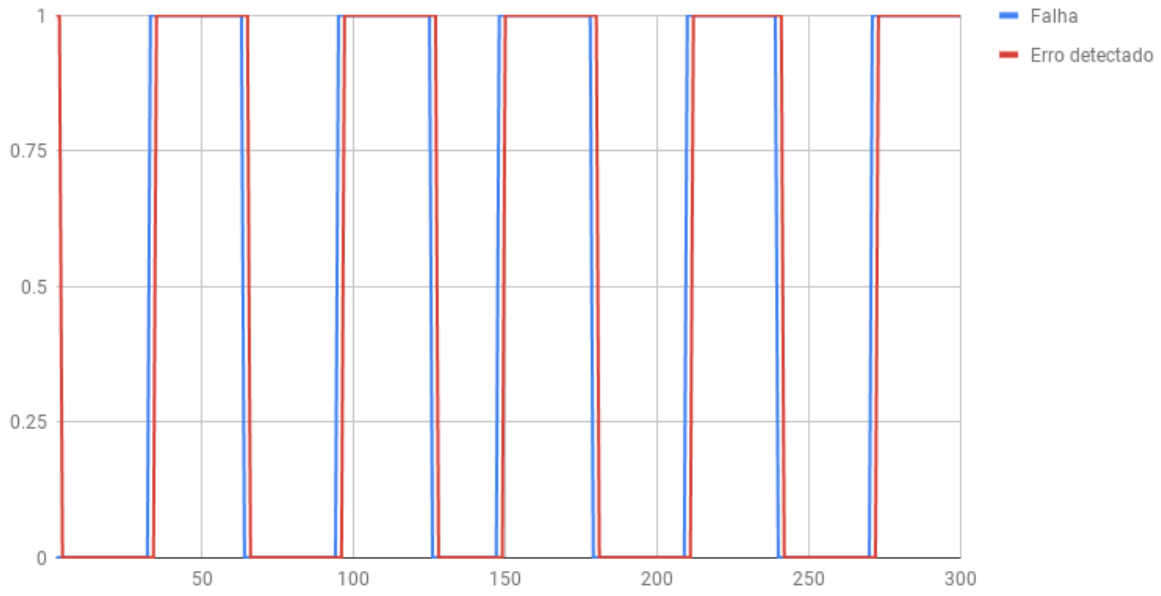
4.5.1 Falha do Zero Cross

O resultado do teste de injeção de falha no Zero Cross é apresentado na Figura 46. Nesse gráfico, um valor igual à um, indica que a falha foi detectada ou causada e, um valor igual a zero, indica que a falha foi retirada. Para a aquisição dos dados foi utilizado um programa em Python que gerencia o Arduíno da placa requisitando que ele gere a falha e verifica a informação do gateway se o dispositivo está com erro. Esse programa encontra-se no Anexo F.

Como pode ser visto na Figura 46, existe um atraso entre a emissão do sinal de falha e a informação do mesmo na interface. Destaca-se que esse atraso é de um a dois segundos no entanto ele inclui é contabilizado a partir do momento em que o programa envia a requisição de falha ao dispositivo de injeção.

Os resultados apresentados na Figura 46 demostram que o algoritmo utilizado para a detecção da falha foi efetivo em encontrar todas as vezes que a mesma foi injetada durante os cinco minutos em que o experimento decorreu, assim como foi efetivo em recuperar-se do erro. O algoritmo também foi efetivo ao encontrar a falha quando o sistema foi iniciado já em estado de falha.

Figura 46 – Injeção e detecção da falha de Zero Cross no tempo.

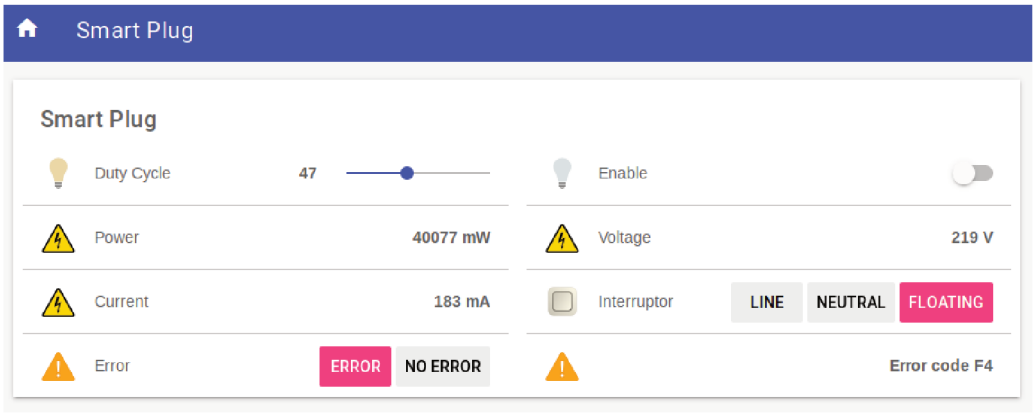


Fonte: Autor (2018)

4.5.2 Falha de Triac em curto

A análise da falha de Triac em curto seguiu o mesmo procedimento de experimentos que a falha de Zero Cross. No entanto, ao invés de injetar e recuperar a falha a cada 30 segundos, como o sistema não se recupera dessa falha isso pode ser feito apenas uma vez. Foi observado que a falha foi corretamente identificada, para isso foi utilizado a interface de usuário. A Figura 47 mostra o erro mostrado por ela. O dispositivo também foi submetido a injeção da falha antes da inicialização, nesse caso a falha foi identificada assim que o sistema foi iniciado.

Figura 47 – Interface de usuário mostrando o código de erro da falha do Triac em curto.



Fonte: Autor (2018)

4.5.3 Falha de Triac aberto

A falha de Triac aberto foi, da mesma forma que a Triac em curto, injetada uma vez com o sistema em operação normal e uma vez antes da inicialização. Em ambos os casos o algoritmo identificou corretamente a falha. No entanto o dispositivo apresentou falsos positivos quando o ciclo de trabalho foi menor que 100%.

Isso acontece porque o algoritmo considera que quando o Triac está conduzindo, ou sem carga, a tensão na entrada do circuito de retorno será a tensão do neutro e quando o Triac estiver aberto com a carga conectada a tensão de entrada do mesmo será a do fase. O algoritmo usa, portanto, a informação do circuito de retorno para separar as situações onde o dispositivo está sem carga ou o Triac está em falha. No entanto, quando o ciclo de trabalho é diferente de 100% a tensão de entrada no circuito de retorno tem um período como fase e um como neutro, isso faz com que a saída do circuito de entrada AC seja uma onda quadrada com um ciclo de trabalho menor que 50% e a rotina de software implementada para a detecção do estado de entrada AC identifica essa saída erroneamente como fase causando assim a identificação do erro sem o mesmo estar presente.

4.5.4 Falha no monitor de tensão

O procedimento de injeção de falha no monitor de tensão foi o mesmo utilizado na injeção de falha no Zero Cross, sendo executado dois experimentos, um em que a falha é injetada e retirada a cada 30 segundos e um em que a falha é injetada antes da inicialização. Em ambos os casos as falha foram identificadas corretamente. Destaca-se que o comportamento temporal da falha teve o mesmo comportamento apresentado na figura 46 e para a aquisição dos dados foi utilizado o mesmo programa disponível no Anexo F.

5 CONCLUSÃO

Esse trabalho teve como objetivo a apresentação e validação de uma solução para o desenvolvimento de uma tomada conectada capaz de monitorar o consumo de equipamentos. Ela também possui as funcionalidade do controle de cargas elétricas, identificação do estado de interruptores a ela conectadas, suporte a autodiagnóstico e é capaz de se comunicar com um gateway com capacitada de interagir com o usuário.

Inicialmente foi executado um levantamento de requisitos que auxiliou na tomada de decisões para o desenvolvimento do projeto. Foi então desenvolvido o hardware de um dispositivo capaz de ser alimentado diretamente pela rede, controlar as cargas pelo uso de Triacs, se comunicar através do protocolo ZigBee, identificar sinais de interruptores conectados diretamente com a rede e ler a potência de cargas ele conectadas. Após a definição do hardware o software foi projetado com base no EPOS e com técnicas de orientação a objeto objetivando modularidade e boa manutenibilidade do software.

As soluções desenvolvidas tiveram como objetivo produzir uma aplicação de baixo custo e boa confiabilidade, conforme os requisitos levantados. Para isso, o projeto explorou técnicas de integração de sistemas de autodiagnóstico no dispositivo e a injeção de falhas para teste e avaliação do sistema. Optou-se ainda pela utilização do openHab como sistema supervisório com o objetivo de facilitar a integração desses dispositivos em ambientes com uma grande variedade de dispositivos.

A validação foi feita através da construção de um protótipo preparado para a injeção de falhas e uma placa de circuito que auxilia na injeção automática delas. Foram realizados experimentos de funcionalidade e confiabilidade apresentando resultados como a medição de corrente, tensão e potência das cargas conectadas, a correta identificação de falhas de hardware que o sistema esta sujeito, controle da tensão aplicada as cargas conectadas e a identificação do estado de interruptores. Os experimentos demostraram também a interação dessas com a interface de usuário. Foi apresentado a viabilidade técnica da construção desse dispositivo em uma placa de circuito impresso de 39,3x30mm .

Dessa forma ao analisar os resultados apresentados, destaca-se que o dispositivo alcançou os requisitos estabelecidos de confiabilidade, coleta, atuação, valor e interface gráfica.

O modelo de dispositivo proposto pode ser uma solução viável para a aplicação em ambientes, onde busca-se prover uma melhoria no conforto. Assim como, utilizar as informações de consumo fornecidas pelo dispositivo para de incentivar a redução da utilização de energia elétrica.

No entanto, o dispositivo apresentado possui oportunidades para melhorias

que podem ser exploradas em futuros trabalhos como, por exemplo:

- melhoria do algoritmos de detecção de falha do Triac em estado aberto,
- ampliação de quais falhas são detectáveis pelo dispositivo,
- melhoria da precisão de leitura da corrente através da escolha dinâmica do ganho do sensor,
- análise e calibração dos ganhos dos sensores de corrente e tensão,
- análise da resposta do sistema a injeção de ruídos e falhas transitórias,
- verificação do desempenho do sistema para cargas não lineares,
- análise do desempenho em um ambiente com vários dispositivo, e
- criação de uma arquitetura para implementar ao EPOS o suporte a autodiagnóstico.

REFERÊNCIAS

AGêNCIA NACIONAL DE ENERGIA ELéTRICA. **Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional**. [S.l.], 2018. Rev. 10.

ASHTON, K. That 'internet of things' thing. **RFiD Journal**, v. 22, n. 7, 2011.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. **Computer Networks**, v. 54, n. 15, p. 2787 – 2805, 2010. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128610001568>.

COSTA, J. C. B. **Ferramenta de apoio ao projeto, configuração e gestão de instalações domóticas**. Dissertação (Mestrado) — Instituto Superior de Engenharia do Porto, 2012.

DIAS, C. L. de A. **Domótica: aplicabilidade às edificações residenciais**. Dissertação (Mestrado) — Universidade Federal Fluminense, 2004.

DORF, R. **Introdução aos circuitos elétricos**. Livros Técnicos e Científicos, 2003. ISBN 9788521613671. Disponível em: <https://books.google.com.br/books?id=83NVAgAACAAJ>.

EHRHARDT-MARTINEZ, K.; DONNELLY, K.; LAITNER, J. Advanced metering initiatives and residential feedback programs: a meta-review for household electricity-saving opportunities. 01 2010.

HART, D. W. **Eletrônica de Potência**. Porto Alegre: AMGH, 2012. ISBN 9788580550450.

HSUEH, M.-C.; TSAI, T. K.; IYER, R. K. Fault injection techniques and tools. **Computer**, v. 30, n. 4, p. 75–82, April 1997. ISSN 0018-9162.

ITU. **ITU-T Y.4000: Overview of the Internet of things**. 2012. (Acessado em 24/09/2017). Disponível em: <http://handle.itu.int/11.1002/1000/11559>.

LISHA. **EPOSMote III**. 2017. (Acessado em 20/11/2017). Disponível em: <http://epos.lisha.ufsc.br/EPOSMote+III#Overview>.

LISHA. **Welcome to the EPOS Project**. 2017. (Acessado em 20/11/2017). Disponível em: <https://epos.lisha.ufsc.br/HomePage>.

LISHA. **Welcome to the EPOS Project**. 2017. (Acessado em 20/11/2017). Disponível em: <https://epos.lisha.ufsc.br/EPOS+Overview>.

MALVINO, A.; BATES, D. J. **Eletrônica**. Porto Alegre: AMGH, 2007.

MEELDIJK, V. **Electronic Components: Selection and Application Guidelines**. Wiley, 1995. (A Wiley-Interscience publication). ISBN 9780471022879. Disponível em: <https://books.google.com.br/books?id=lqQwAQAAMAAJ>.

NEXPERIA. **AN467**: Nxp's 51lpc - microcontrollers and triacs easily connected. [S.l.], 2013. Rev. 2.

NXP. **BT137S-600E**. [S.l.], 2014.

OLBRICH, T.; RICHARDSON, A.; BRADLEY, D. Built-in self-test and diagnostic support for safety critical microsystems. **Microelectronics Reliability**, v. 36, n. 7, p. 1125 – 1136, 1996. ISSN 0026-2714. Reliability Physics of Advanced Electron Devices. Disponível em: <http://www.sciencedirect.com/science/article/pii/0026271496000352>.

openHab. **Welcome to the openHAB 2 Documentation!** 2017. (Acessado em 20/11/2017). Disponível em: <http://docs.openhab.org/>.

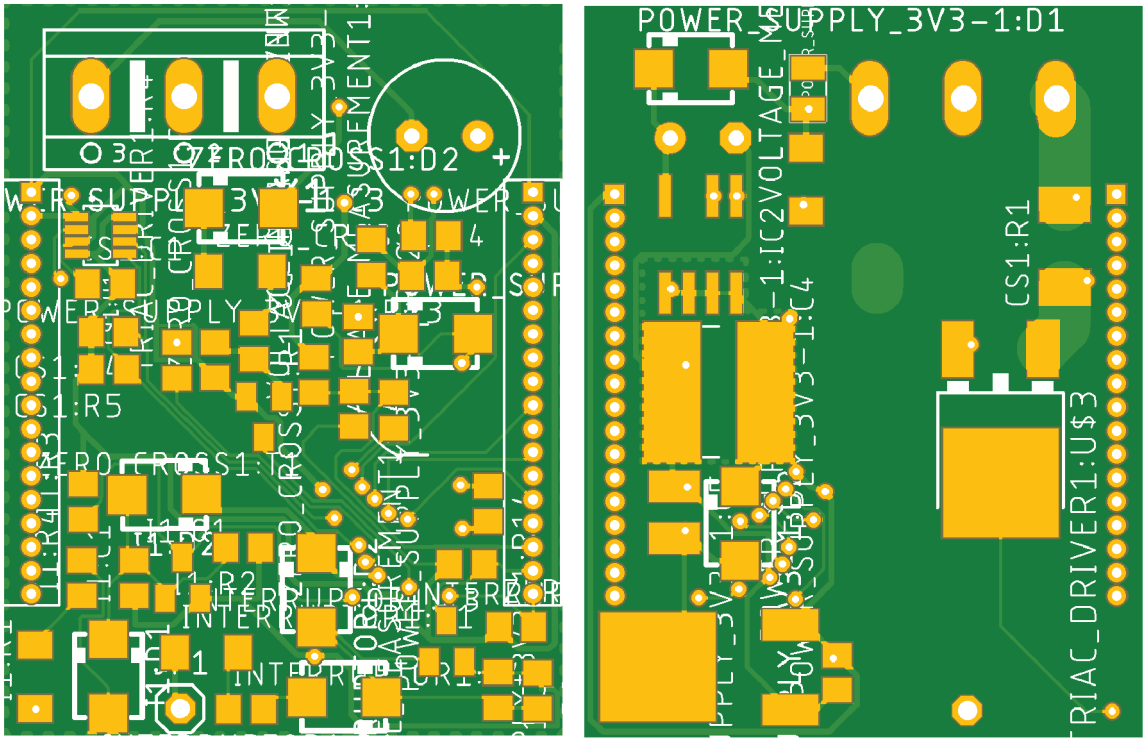
POWER INTEGRATIONS. **Application Note AN-37 LinkSwitch-TN Family**. [S.l.], 2014. Rev. F.

STMICROELECTRONICS. **Snubberless™ and logic level TRIAC behavior at turn-off**. [S.l.], 2008. Rev. 3.

ZIADE, H.; AYOUBI, R.; VELAZCO, R. **A Survey on Fault Injection Techniques**. 2003.

APÊNDICE A – MODELO 2D DA PLACA DE CIRCUITO IMPRESSO

Figura 48 – Modelo 2D da camada de cima (a) e de baixo (b) da placa.



a) cima b) baixo

Fonte: O Autor (2018)

APÊNDICE B – BOM DO DISPOSITIVO MONITOR DE CONSUME ELÉTRICO

Part	Value	Package	Description	Cost
CS1:C1	0.1uF 6V	C0805	CAPACITOR, European symbol	0.1
CS1:R1	40 m	R2010	RESISTOR, European symbol	0.01
CS1:R4	10k	R0805	RESISTOR, European symbol	0.01
CS1:R5	10k	R0805	RESISTOR, European symbol	0.01
CS1:U1		SOT23-8	INA219 - I2C Current/Power Monitor	2.07
I1:C1	1nf	C0805	CAPACITOR, European symbol	0.1
I1:D1	S1M	SMA		0.31
I1:D2	S1M	SMA		0.31
I1:Q1	BC846ASMD	SOT23	NPN Transistor	0.21
I1:R1	84k	M1406	RESISTOR, European symbol	0.01
I1:R2	100k	R0805	RESISTOR, European symbol	0.01
I1:R3	10k	R0805	RESISTOR, European symbol	0.01
I1:R4	84k	R0805	RESISTOR, European symbol	0.01
INTERRUPTOR1:C1	1nf	C0805	CAPACITOR, European symbol	0.1
INTERRUPTOR1:D1	S1M	SMA		0.31
INTERRUPTOR1:D2	S1M	SMA		0.31
INTERRUPTOR1:Q1	BC846ASMD	SOT23	NPN Transistor	0.21
INTERRUPTOR1:R1	84k	M1406	RESISTOR, European symbol	0.01
INTERRUPTOR1:R2	100k	R0805	RESISTOR, European symbol	0.01

INTERRUPTOR1:R3	10k	R0805	RESISTOR, European symbol	0.01
INTERRUPTOR1:R4	84k	R0805	RESISTOR, European symbol	0.01
POWER_SUPPLY_3V3-1:C1	10uF 10V	C0805	CAPACITOR, European symbol	0.1
POWER_SUPPLY_3V3-1:C2	0.1uF 10V	C0805	CAPACITOR, European symbol	0.1
POWER_SUPPLY_3V3-1:C3	4.7uf	E3,5-8	POLARIZED CAPACITOR, European symbol	0.41
POWER_SUPPLY_3V3-1:C4	10uF 10V	C0805	CAPACITOR, European symbol	0.1
POWER_SUPPLY_3V3-1:C5	100uf 10V	C1210	CAPACITOR, European symbol	1.24
POWER_SUPPLY_3V3-1:D1	S1M	SMA		0.31
POWER_SUPPLY_3V3-1:D2	US1M	SMA		0.43
POWER_SUPPLY_3V3-1:D3	S1M	SMA		0.31
POWER_SUPPLY_3V3-1:FUSE	FUSE	1206		0.24
POWER_SUPPLY_3V3-1:IC2		SO8-C	LinkSwitch-XT incorporates a 700 V power MOSFET, oscillator,	1.77
POWER_SUPPLY_3V3-1:R12	6.68k	R0805	RESISTOR, European symbol	0.01
POWER_SUPPLY_3V3-1:R13	2k49	R0805	RESISTOR, European symbol	0.01
POWER_SUPPLY_3V3-1:R14	1k5	R0805	RESISTOR, European symbol	0.01
POWER_SUPPLY_3V3-1:U$1	7905	DPAK-3		0.67
POWER_SUPPLY_3V3-1:U$9	CD75 680uh	CD75		0.97

TRIAC_DRIVER1:R4	660	R0805	RESISTOR, European symbol	0.01
TRIAC_DRIVER1:U$3	BT136S	DPAK-3		0.68
VOLTAGE_MEASUREMENT1:C1	1uf	C0805	CAPACITOR, European symbol	0.1
VOLTAGE_MEASUREMENT1:R1	600k	M1406	RESISTOR, European symbol	0.01
VOLTAGE_MEASUREMENT1:R2	2k7	R0805	RESISTOR, European symbol	0.01
VOLTAGE_MEASUREMENT1:R3	27k	R0805	RESISTOR, European symbol	0.01
VOLTAGE_MEASUREMENT1:R4	27k	R0805	RESISTOR, European symbol	0.01
X1		AK500/3	CONNECTOR	0.74
ZERO_CROSS1:D2	S1M	SMA		0.31
ZERO_CROSS1:R1	80k	R0805	RESISTOR, European symbol	0.01
ZERO_CROSS1:R2	20k	R0805	RESISTOR, European symbol	0.01
ZERO_CROSS1:R3	10k	R0805	RESISTOR, European symbol	0.01
ZERO_CROSS1:R4	80k	M1406	RESISTOR, European symbol	0.01
ZERO_CROSS1:T1	BC846ASMD	SOT23	NPN Transistor	0.21

APÊNDICE C – ITENS UTILIZADOS NO OPENHAB

```
1  String Epos_TX "Epos_TX [%s]" { serial = "/dev/ttyACM0" }
2  String Epos_RX "Epos_RX [%s]" { serial = "/dev/ttyACM0" }
3
4  Number duty_cycle "Duty Cycle"
5  Switch enabled    "Enabled"
6  Number power      "Power"
7  Number voltage    "Voltage"
8  Number current    "Current"
9  String interruptor "Interruptor"
10 Switch error      "Error"
11 String errorCode  "ErrorCode"
```

APÊNDICE D – REGRAS UTILIZADAS NO OPENHAB

```
rule "Object 1 Duty cycle sender"
when
    Item duty_cycle changed
then
    sendCommand(Epos_TX, "{1,0,0," + duty_cycle.state.toString + "}")
end

rule "Object 1 Enabled sender"
when
    Item enabled changed
then
    if (enabled.state == ON)
        sendCommand(Epos_TX, "{1,0,1,1}")
    else
        sendCommand(Epos_TX, "{1,0,1,0}")
end

rule "Pooling"
when
    Time cron "0/1 * * ? * *"
then
    sendCommand(Epos_TX, "{1,1,6}");
end

rule "Power Receive"
when
    Item Epos_RX changed
then
    var String string1 = Epos_RX.state.toString;
    string1 = string1.substring(1, string1.length()-3);
    val payload = string1.split(",");
    switch (payload.get(2))
    {
        case "2":
        {
            power.postUpdate(payload.get(3));
        }
        case "4":
        {
            voltage.postUpdate(payload.get(3));
        }
        case "5":
        {
            current.postUpdate(payload.get(3));
        }
        case "6":
        {
            power.postUpdate(payload.get(5));
            interruptor.postUpdate(payload.get(6));
            voltage.postUpdate(payload.get(7));
            current.postUpdate(payload.get(8));
```

```
52              if (payload.get(9) == "0")
53              {
54                  error.postUpdate(OFF);
55                  errorCode.postUpdate("No Error");
56              }
57              else
58              {
59                  error.postUpdate(ON);
60                  errorCode.postUpdate("Error code F" + payload.get(9));
61              }
62          }
63      };
64  end
```

APÊNDICE E – SITEMAP UTILIZADO NO OPENHAB

```
sitemap default label="Smart Plug"
{
    Frame label="Smart Plug"
    {
        Slider item=duty_cycle label="Duty Cycle [%d]" icon="light"
        Switch item=enabled    label="Enable" icon="light"
        Text   item=power      label="Power   [%d mW]"  icon="energy"
        Text   item=voltage    label="Voltage [%d V]"  icon="energy"
        Text   item=current    label="Current [%d mA]"  icon="energy"
        Switch item=interruptor label="Interruptor" icon="wallswitch" ...
            mappings=[0="LINE", 1="NEUTRAL", 2="FLOATING"]
        Switch item=error label="Error" icon="error" mappings=[ON="ERROR", OFF="NO ERROR"]
        Text   item=errorCode   label="[%s]"  icon="error"
    }
}
```

APÊNDICE F – PROGRAMA UTILIZADO NA AQUISIÇÃO DOS DADOS DOS TESTES DE INJEÇÃO DE FALHAS

```
1  import serial
2  import time
3  from datetime import datetime
4
5  ser = serial.Serial('/dev/ttyACM1', 115200)
6  arduino = serial.Serial('/dev/ttyUSB0', 9600)
7  file  = open("errorVoltage", "w")
8
9  ser.write("{1,0,0,100}")
10 time.sleep(1)
11 ser.reset_input_buffer()
12
13 i = 0
14 arduino.write("1")
15 falha = False
16 arduinoTimer = datetime.now()
17 while i < 300:
18     now = datetime.now()
19     Timediff = now - arduinoTimer
20     if Timediff.seconds > 30:
21         falha = not falha
22         arduinoTimer = datetime.now()
23         if falha :
24             arduino.write("2")
25         else:
26             arduino.write("1")
27
28     ser.write("{1,1,6}")
29     a = datetime.now()
30
31     while(ser.in_waiting == 0):
32         b = datetime.now()
33         c = b-a
34         if c.microseconds > 700000 :
35             ser.reset_input_buffer()
36             ser.reset_output_buffer()
37             ser.write("{1,1,6}")
38             a = datetime.now()
39         pass
40
41     line = ser.readline()
42     line = line[1:-3]
43     payload = line.split(',')
44     print payload,
45
46     if len(payload) == 10:
47         pot = (payload[9])
48         file.write(str(falha) + " : ")
49         file.write(pot + '\n')
```

```
50          time.sleep(1)
51          ser.reset_input_buffer()
52          i += 1
53          print (" " + str(i) + " " + str(falha))
54
55  ser.close()
56  file.close()
57  arduino.close
```