UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E ELETRÔNICA

Lucas Fernandes Andrade

Adaptador Inteligente para Automação Residencial baseado em Internet das Coisas

Florianópolis
2018

LUCAS FERNANDES ANDRADE

ADAPTADOR INTELIGENTE PARA AUTOMAÇÃO RESIDENCIAL BASEADO EM INTERNET DAS COISAS

Monografia submetida ao curso de Engenharia Eletrônica da Universidade Federal de Santa Catarina como requisito para aprovação da disciplina EEL7806 - Projeto Final TCC.
Orientador: Prof. Eduardo Luiz Ortiz Batista

FLORIANÓPOLIS
2018

RESUMO

Diante o cenário de diminuição dos custos de dispositivos eletrônicos conectados à internet, há um aumento da busca por soluções voltada à área de automação residencial. Tal contexto impulsionou o presente trabalho, cujo objetivo é o desenvolvimento de um adaptador de tomada conectado à internet. Esse adaptador fornece ao usuário o controle "ligar e desligar" dos dispositivos conectados via *website*, além do monitoramento do consumo de energia do equipamento conectado ao mesmo. O enfoque do trabalho se deu no projeto da infraestrutura para a comunicação entre dispositivo, aplicação web e nuvem, com auxílio de ferramentas para internet das coisas disponibilizadas no âmbito da Amazon Web Services (AWS IoT). Essa, por sua vez, possui uma infraestrutura que permite a segurança da comunicação dos dispositivos utilizando certificados X.509. Em relação à segurança da aplicação web, foi implementado um serviço de autenticação com criptografia de uma via. Por fim, a respeito do hardware, empregou-se o módulo ESP8266 para acesso Wi-Fi, além de um transformador de corrente toroidal para a medição de corrente.

Palavras-chave: Internet das coisas. Automação residencial. React. Mobx. Express. Node. ESP8266.

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Andrade, Lucas Fernandes
Adaptador Inteligente para Automação Residencial Baseado em Internet das Coisas / Lucas Fernandes Andrade ; orientador, Eduardo Luiz Ortiz Batista, 2018.
86 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Eletrônica, Florianópolis, 2018.

Inclui referências.

1. Engenharia Eletrônica. 2. Internet das Coisas. 3. Automação Residencial. 4. Node. 5. React. I. Batista, Eduardo Luiz Ortiz. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Eletrônica. III. Título.

Lucas Fernandes Andrade

ADAPTADOR INTELIGENTE PARA AUTOMAÇÃO RESIDENCIAL BASEADO EM INTERNET DAS COISAS

Esta monografia foi julgada no contexto da disciplina EEL7806 - Projeto Final TCC e aprovada em sua forma final pelo Curso de Engenharia Eletrônica.
Florianópolis, 29/11/2018.

Jefferson Luiz Brum Marques, Dr.
Coordenador do Curso

Banca Examinadora:

Eduardo Luiz Ortiz Batista, Dr.
Orientador

Richard Demo Souza, Dr.

Walter Antonio Gontijo, M.Sc.

AGRADECIMENTOS

Primeiramente gostaria de agradecer aos meus pais, Maria e Paulo, pelo carinho e apoio durante todos esses anos. Ao meu orientador Eduardo Batista, pela ajuda e esforço durante o desenvolvimento do trabalho. Também gostaria de agradecer aos meus colegas de curso, pelos momentos divertidos e cafés compartilhados. Por fim, gostaria de agradecer à Luiza pela companhia e jogatinas nas horas vagas.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

ADC Conversor Analógico-Digital (*Analog-to-digital Converter*)

API Interface de Programação de Aplicações (*Application Programming Interfacel*)

AWS *Amazon Web Services*

CRUD Criar

GPIO Portas de entrada/saída de propósito geral (*General-purpose input/output*)

HTTP Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol*)

I2C Circuito Inter-integrado (*Inter-Integrated Circuit*)

IoT Internet das Coisas (*Internet of Things*)

JSON Notação de Objeto Javascript (*JavaScript Object Notation*)

JWT Token Web JSON (*JSON Web Token*)

MAC Controle de acesso de mídia (*Media Access Control*)

MQTT Transporte de telemetria de enfileiramento de mensagens (*Message Queuing Telemetry Transport*)

PCB Placa de circuito impresso (*Printed Circuit Board*)

PWM Modulação por Largura de Pulso (*Pulse Width Modulation*)

RAM Memória de Acesso Aleatório (*Random Access Memory*)

RISC Computador com um Conjunto Reduzido de Instruções (*Reduced Instruction Set Computer*)

RMS Raiz do valor quadrático médio (*Root Mean Square*)

SDK Kit de Desenvolvimento de Software (*Software development kit*)

SPI Interface Periférica Serial (*Serial Peripheral Interface*)

UART Transmissor/Receptor Universal Síncrono e Assíncrono (*Universal Asynchronous Receiver-Transmitter*)

USB Barramento Serial Universal (*Universal Serial Bus*)

Sumário

CAPÍTULO 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

A recente redução de custos de sistemas eletrônicos em geral, aliada a uma crescente capacidade de comunicação desses sistemas via internet, tem motivado o desenvolvimento de soluções para automação residencial conectadas à internet. Dentre as soluções mais comuns, há a substituição das tomadas residenciais por tomadas com conexão à internet. Entretanto, essa proposta envolve uma instalação complexa e arriscada para a maioria dos usuários. Nesse contexto, o emprego de adaptadores para o controle de dispositivos domésticos visa a solução do problema de instalação que essa alternativa apresenta, uma vez que a instalação do adaptador é executada de maneira *plug and play*, evitando-se assim a necessidade da remoção das tomadas residenciais já exis-

tentes.

Em acréscimo, a conexão direta dos adaptadores aos eletrodomésticos facilita a aquisição de informações como, por exemplo, o consumo de energia. Essa é uma informação relevante para a maioria dos consumidores, tendo em vista que ela permite a noção do consumo específico de cada eletrodoméstico, facilitando o controle dos gastos de energia residencial.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O presente trabalho tem como objetivo desenvolver um adaptador para tomada que pode ser controlado via sítio web. Assim, o usuário poderá ligar ou desligar o aparelho conectado ao adaptador a partir de um relé que pode ser acionado via internet. O equipamento também deve ser capaz de obter as informações relativas ao consumo de energia do dispositivo a ele conectado, enviando os dados obtidos para um sistema que roda em nuvem.

1.2.2 Objetivos Específicos

Os objetivos específicos citados a seguir representam as etapas necessárias para realização do projeto:

- Desenvolvimento de uma aplicação Web utilizando biblioteca React;
- Implementação de um sistema de autenticação para a aplicação Web;

- Implementação de um servidor HTTP via Node.js;
- Desenvolvimento do *firmware* para o microcontrolador ESP8266;
- Implementação da comunicação MQTT dipositivo - nuvem;
- Implementação da comunicação nuvem - servidor HTTP;
- Desenvolver esquemático do *hardware*;
- Desenvolver *layout* da placa de circuito impresso;
- Confecção da placa de circuito impresso;
- Desenvolver e imprimir o modelo 3D do adaptador de tomada;
- Testar funcionamento do *hardware*;
- Avaliação do consumo do adaptador;
- Avaliação do sensor de corrente.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

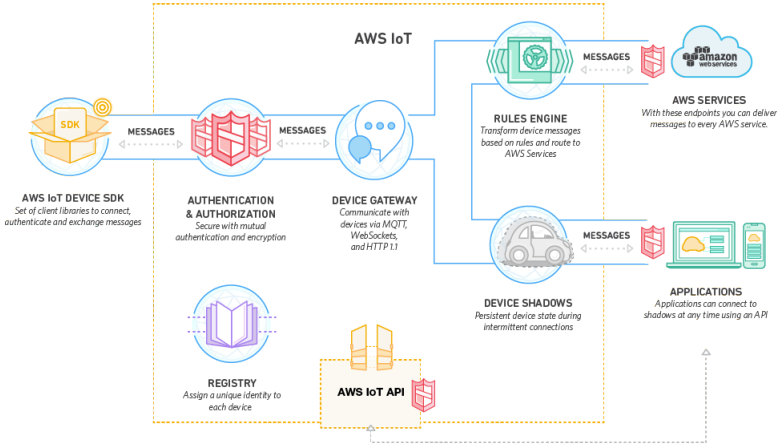
Este capítulo é dedicado à descrição da fundamentação teórica do projeto desenvolvido. Nesse contexto, as ferramentas de software utilizadas são inicialmente descritas, seguida pela descrição do hardware utilizado. Por fim, a Seção 2.7 descreve trabalhos correlatos ao presente projeto.

2.1 Amazon AWS IoT

A plataforma de IoT em nuvem utilizada para o desenvolvimento do presente projeto é a Amazon AWS IoT [1]. Tal plataforma apresenta diversos mecanismos de suporte ao desenvolvimento de aplicações em IoT, tais como soluções de comunicação e segurança, dentre outras. Para compreensão inicial, o fluxograma básico da comunicação entre o dispositivo e as aplicações pode ser visualizado na Figura 2.1. Os

blocos referentes ao fluxograma serão explicados detalhadamente a seguir.

Figura 2.1: Fluxograma da comunicação com a AWS IoT.



Fonte: (AMAZON, 2017)

2.1.1 SDK do dispositivo AWS IoT

Acompanhando o fluxograma da Figura 2.1, nota-se que o primeiro passo é a comunicação do dispositivo com a AWS IoT. Para tal, a Amazon AWS disponibiliza diversos kits de desenvolvimento de software, mas até o presente momento nenhum deles pode ser aplicado diretamente ao módulo Wi-fi ESP8266. Contudo, o sistema operacional de código livre chamado Mongoose OS (descrito na Seção 2.2) possui bibliotecas para a comunicação com a AWS IoT nativamente, por esse motivo o sistema foi aplicado ao ESP8266 no presente projeto.

2.1.2 Autenticação e autorização

Anterior a qualquer comunicação entre dispositivos de IoT e a nuvem, uma autenticação segura e robusta precisa ser realizada. Para isso, a AWS IoT disponibiliza ferramentas baseadas em métodos de criptografia mútua, sendo eles compatíveis com o SigV4 [2] e com certificados X.509 [16].

2.1.3 Gateway do dispositivo

Ainda a respeito da comunicação dos dispositivos com a nuvem, os seguintes protocolos podem de ser utilizados em conjunto com a AWS IoT:

- WebSocket [18];
- MQTT [19];
- HTTP 1.1 [17].

Para o presente trabalho, o protocolo MQTT (Ver Seção 2.5) foi empregado, por ser um protocolo que necessita de baixa largura de banda quando comparado a HTTP 1.1 [20].

2.1.4 Sombras do dispositivo

As *sombras do dispositivo* (do termo original em inglês *Thing Shadow*) são uma maneira de persistir à informação do estado de um dispositivo no momento em que ocorre a falha de comunicação. As *sombras* armazenam a informação do último estado que o dispositivo em questão possuía antes da falha. Assim sendo, sempre que uma aplicação executa a requisição do estado do dispositivo, a informação armazenada pela *sombra* é retornada.

Nas *sombras*, ao definir o estado de um dispositivo, ele é armazenado como estado futuro. Caso o dispositivo retome a conexão, o estado atual será substituído pelo estado futuro.

2.1.5 Mecanismo de Regras

O Mecanismo de Regras é o possibilitador da integração de mensagens dos dispositivos com serviços oferecidos pela AWS. Para o projeto em questão, a ação de inserção de mensagens em uma tabela do banco de dados DynamoDB (descrito na Seção 2.3) será utilizada.

2.2 Mongoose OS

Mongoose OS é um sistema operacional de código livre que facilita o desenvolvimento de *firmware* aplicado a internet das coisas (IoT). Ele permite a programação de microcontroladores via uma interface que utiliza o navegador Web ou linha de comando. Essa ferramenta possibilita a integração de dispositivos IoT com diversos servidores MQTT (Normalmente referidos como *brokers*) públicos e privados, como por exemplo:

- AWS IoT [1];
- Mosquitto [3];
- HiveMQ [15].

O Mongoose OS também permite a programação de diversos microcontroladores conectados a internet, como os seguintes:

- ESP32;
- ESP8266;
- TI CC3200;
- STM32.

A programação do dispositivo em questão pode ser feita usando linguagens C e Javascript. Essa última foi utilizada no presente trabalho, visando manter a mesma linguagem tanto no *front end* (parte responsável pelo visual da aplicação Web) e *back end* (parte responsável pelo gerenciamento dos dados, no caso, o servidor HTTP), quanto no do dispositivo.

Por fim, a implementação das criptografias necessárias para a comunicação via AWS IoT também é facilitada pelo uso do Mongoose OS.

2.3 DynamoDB

DynamoDB é um banco de dados não-relacional disponível na AWS que permite alta escalabilidade. Por meio dessa característica, que se deve ao escalonamento automático disponível na AWS, caso um produto que utiliza o banco de dados passe a ter uma rápida adoção, a quantidade de recursos da Amazon destinadas a aplicação será ajustada para que sirva a necessidade.

Outro fator importante do uso do DynamoDB é a segurança e backup, uma vez que ambos passam a ser controlados pela AWS, que possui alta quantidade de recursos e investimentos na área.

O DynamoDB também apresenta integração com o serviço de regras da AWS IoT. Sendo assim, para o presente

Dados transferidos	Definição de preço
Transferência de dados para DENTRO	
Todas as transferências de dados para dentro	0,00 USB por GB
Transferência de dados para FORA	
Até 1GB/mês	0,00 USD por GB
Próximos 9,999TB/mês	0,09USB por GB
Próximos 40TB/mês	0,085USB por GB
Próximos 100TB/mês	0,07USD por GB
Mais que 150TB/mês	0,05 USD por GB

Tabela 2.1: Tabela preços do DynamoDB

projeto os dados referentes a potência do dispositivo são enviados diretamente a AWS IoT, que por sua vez registra tais dados no DynamoDB via o serviço de regras, sem a necessidade do envio ser feito primeiramente para o servidor HTTP.

O DynamoDB é um serviço pago a partir de uma certa quantidade de armazenamento ou acesso ao banco, porém o nível gratuito abrange 25GB por mês de armazenagem de dados, até 200 milhões de solicitações por mês e 2,5 milhões de solicitações de stream por mês. Após essa quantidade de uso, o preço do armazenamento de dados é de 0,25 dólares para cada GB adicional, enquanto que o preço para a transferência de dados pode ser visto na Tabela 2.1 (sendo que a transferência de dados "para dentro"representa a transferência para outros serviços internos a AWS).

Alternativas de bancos de dados não relacionais gratuitas existem, como por exemplo o MongoDB [8]. Contudo a alta escalabilidade, integração com AWS IoT e a alta segurança mantida pela Amazon foram as justificativas para a escolha do DynamoDB.

2.4 Aplicacao Web

Essa seção é dedicada a introduzir os conceitos básicos das tecnologias aplicadas no presente trabalho para o desenvolvimento da aplicação web e servidor. Para tal, primeiramente na Subseção 2.4.1 a biblioteca React, empregada na criação da interface do usuário é apresentada. Posteriormente, na Subseção 2.4.2 a ferramenta Node.js aplicada na elaboração do servidor HTTP foi introduzida. Por fim, as tecnologias empregadas no serviço de autenticação do usuário foram esclarecidas nas Subseções 2.4.3 e 2.4.4.

2.4.1 Biblioteca React

A interface da aplicação web foi desenvolvida com auxílio da biblioteca React. A elaboração de interfaces com React é baseada na criação de componentes e na composição dos mesmos para a formação de interfaces mais complexas. Uma da vantagens do emprego do React é a eficiência em manipular o Modelo de Documento por Objetos (do inglês Document Object Model - DOM) de uma página. Isso se deve ao fato de que sempre que um dado relativo a um componente for alterado, apenas a diferença entre o estado anterior e posterior a alteração é enviada a DOM. Sendo assim, as alterações nos estados dos componentes são mais eficientes.

2.4.2 Node

A Node.js é uma ferramenta que permite a utilização da linguagem JavaScript para a programação do servidor HTTP. O JavaScript originalmente foi desenvolvido para ser uma linguagem de programação Web para o *client-side*, sendo

executada no navegador do usuário. Por fim, o Node.js permite a unificação das linguagens do *client-side* com o *server-side*, o que facilita o desenvolvimento da aplicação Web em questão.

2.4.2.1 Express

O Express.js é um framework de código livre para a ferramenta anteriormente mencionada, Node.js, que possui vários métodos que facilitam a implementação de servidores e o desenvolvimento de APIs, que por consequência auxilia no uso de métodos HTTP e *endpoints*.

2.4.3 JSON Web Token

O JSON Web Token é um padrão (RFC 7519 [5]) que define um método de transmitir documento JSON (Notação de Objetos JavaScript - *JavaScript Object Notation*). Tal padrão foi criado para garantir uma comunicação segura entre duas partes. Para realizar essa função, o JWT emprega o método de assinatura criptográfica, na qual os métodos utilizados são ou HMAC [6] SHA256 [7] ou RSA [13].

A estrutura do JSON Web Token é dividida em *Header*, *Payload* e *Signature*. Na primeira parte, a informação contendo o tipo da criptografia e o tipo do *token* é enviada em um documento JSON. No *Payload* devem ser enviadas informações relativas ao usuário, como o nome, o tipo de permissões e o grau de acesso do mesmo. Por fim, a assinatura é formada pela concatenação dos outros atributos junto a chave secreta ou do certificado RSA, o que depende do tipo de criptografia empregada, sendo que cada atributo a ser concatenado é codificado para base64 [9].

2.4.4 BCrypt

BCrypt é um método de criptografia de senhas criado com o intuito de tentar resolver alguns problemas ligados à segurança do usuário. Tendo em vista que a velocidade de processamento das máquinas evolui com velocidade muito superior à velocidade em que os humanos geram senhas mais complexas e longas , estratégias são necessárias para a devida aplicação de segurança para as senhas.

O BCrypt [30] é uma criptografia do tipo *hash*, onde é aplicado um *Salt* (dado que auxilia na geração da *hash*), sendo assim a criptografia possui apenas uma via, o que torna impossível reverter a mesma. A *hash* é guardada no banco de dados, o que evita com que a senha seja guardada em extenso. Contudo, apenas aplicar uma função *hash* criptográfica não resolveria o problema de um ataque de força bruta, uma vez que com o acesso ao banco uma lista de palavras poderia ser usada para comparar as *hash* geradas, e assim serem descobertas as senhas.

O método principal utilizado pela função BCrypt é a de possuir um parâmetro para ajustar a velocidade da verificação da senha de acordo com o avanço dos processadores, o que impede ataques de força bruta.

2.5 MQTT

O MQTT é um protocolo de comunicação baseado na arquitetura de publicação/assinatura, que apresenta baixo custo em termos de largura de banda, o que justifica ser muito utilizado para aplicações na internet das coisas.

Para uma mensagem ser enviada por um dispositivo, o

mesmo deve publicar um tópico. Os tópicos são rotas que podem ser separadas por barras dependendo da necessidade (Exemplo: 'sensor/temperatura'). As mensagens são recebidas por dispositivos que assinaram o tópico em questão, e uma vez que uma mensagem é enviada a um tópico, todos os dispositivos que tiverem assinado e escutando ao tópico receberão a mensagem.

O meio que intermedeia a comunicação entre a publicação e a assinatura é nomeado *Broker*. Para o presente trabalho, o *Broker* empregado é a AWS IoT.

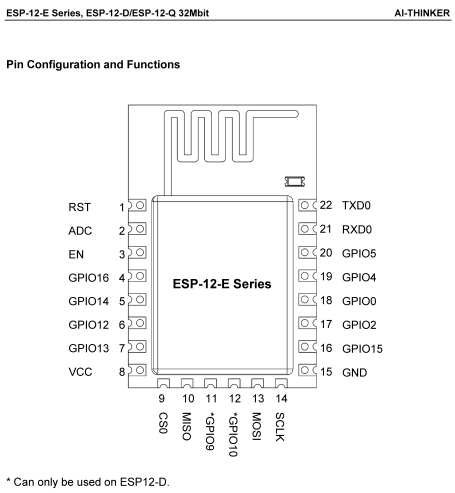
2.6 Hardware

Na presente Seção será introduzido primeiramente o microcontrolador implementado no projeto, em seguida, o chip utilizado para a programação do microcontrolador e por fim, o estudo do tipo de sensor utilizado para mensurar o consumo de energia do dispositivo conectado ao adaptador de tomada.

2.6.1 ESP8266 ESP-12

O ESP8266 é um microcontrolador de baixo custo fabricado pela empresa Espressif, com capacidade de comunicação Wi-Fi. Esse dispositivo possui arquitetura RISC 32 bits com *clock* de 80MHz, podendo chegar até 160MHz. O mesmo possui ainda saídas SPI, I2C, UART e PWM, além de uma porta com conversor analógico digital ADC. Quanto a memória, ele possui 32KBytes de RAM para instruções e 96KBytes de RAM para dados. A pinagem do dispositivo pode ser vista na Figura 2.2.

Figura 2.2: Pinagem do microcontrolador ESP8266 ESP-12.



Fonte: (ESP8266 COMMUNITY WIKI, 2018)

O ESP8266 é um microcontrolador que possui um baixo consumo de energia quando está trabalhando nos modos *Deep-sleep* e *Light-sleep*. A Tabela 2.2 Apresenta um panorama detalhado do consumo dependendo do modo de operação do dispositivo.

A partir de tal tabela, é possível notar então que o consumo do dispositivo pode variar de aproximadamente 0,5uA até 170mA.

2.6.2 CH340g

O CH340g é um chip para conversão USB - Serial de baixo custo, o qual trabalha com tensão de alimentação 3.3V e 5V. Tal chip também pode necessitar da adição de um oscilador externo a cristal de 12MHz para sua operação. No caso

Parâmetro	Típico	Unit
Tx 802.11b, CCK 11Mbps, $P_{OUT} = +17dBm$	170	mA
Tx 802.11b, OFDM 54Mbps, $P_{OUT} = +15dBm$	140	mA
Tx 802.11n, MCS7 , $P_{OUT} = +13dBm$	120	mA
Rx 802.11b, 1024 bytes packet length, -80dBm	50	mA
Rx 802.11g, 1024 bytes packet length, -70dBm	56	mA
Rx 802.11n, 1024 bytes packet length, -65dBm	56	mA
Modem-Sleep	15	mA
Light-Sleep	0.5	mA
Power save mode DTIM 1	1.2	mA
Power save mode DTIM 3	0.9	mA
Deep-Sleep	10	uA
Power OFF	0.5	uA

Tabela 2.2: Tabela de consumo ESP8266

do presente trabalho, o mesmo foi utilizado com o intuito de programar o microcontrolador ESP8266 explicado anteriormente, e também caso o produto final tenha função de atualizar o *firmware* do dispositivo.

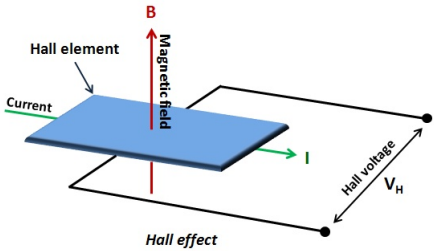
2.6.3 Sensor de Corrente

Para a tarefa de analise da corrente consumida pelo equipamento à ser inserido no adaptador inteligente, dois tipos de alternativas foram encontradas: a primeira delas é o chip ACS712 e a segunda é um transformador de corrente toroidal.

O ACS712 é um sensor de corrente baseado no efeito hall, o qual se baseia no princípio de que ao aplicar uma corrente em um material condutor e um campo magnético no mesmo, uma diferença de tensão é gerada nas bordas desse condutor de maneira perpendicular à corrente e ao campo magnético,

como pode ser visto na figura 2.3.

Figura 2.3: Explicação do efeito Hall.



Fonte: (EMBEDDED LAB, 2018)

A curva da tensão de saída pela corrente detectada pelo sensor pode ser vista na Figura 2.4. Por meio dessa curva é possível a observação do aspecto linear da saída do chip com uma variação de 50mV/A.

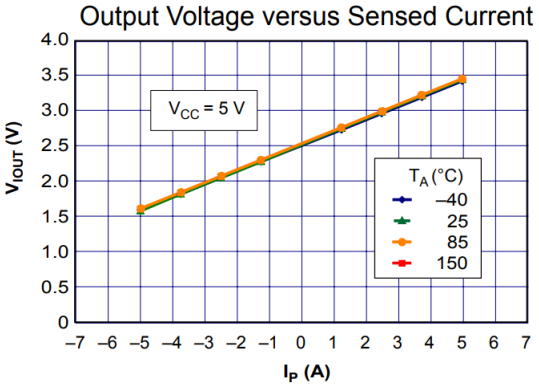
Uma vez que o conversor analógico digital do ESP8266-12 possui 10 bits e 3.3V de alimentação, o tamanho do passo pode ser calculado pela fórmula:

$$Tamanho\ do\ Passo = \frac{V_{source}}{2^N} = \frac{3.3V}{2^{10}} = \frac{3.3V}{1024} = 3.3mV \tag{2.1}$$

Em (2.1), N representa a quantidade de bits do conversor analógico digital. Sendo assim, a resolução do ACS712 é suficiente para ser medida satisfatoriamente pelo ESP8266.

Apesar da análise do circuito se demonstrar uma boa solução para a aplicação, ao ser analisado o *datasheet* foi observado que o dispositivo possui isolamento básico para tensão até 354V de pico, e o isolamento reforçado para tensões de

Figura 2.4: Curva Tensão de saída por corrente medida do ACS712.



Fonte: (EMBEDDED LAB, 2018)

até 184V de pico. Assim, para a nossa aplicação de medição do consumo da rede de aproximadamente 220V RMS, o circuito possui apenas isolamento básico.

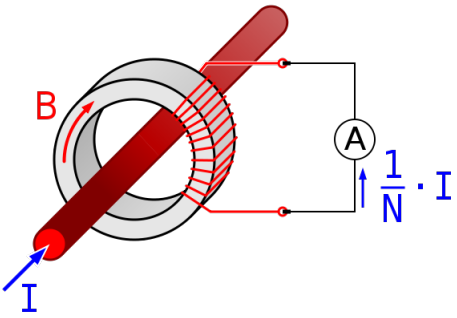
A segunda alternativa para a análise da corrente do dispositivo é o transformador de corrente toroidal HMCT103C (Figura 2.5). O funcionamento desse tipo de sensor é explicado pelo princípio da lei de Faraday da indução, tendo em vista a Figura 2.6 temos que uma corrente alternada gerado no condutor primário induz um campo magnético que varia a sua volta, esse por sua vez induz uma força eletromagnética no condutor secundário.

A Tabela 2.3 apresenta os parâmetros técnicos do HMCT103C.

Figura 2.5: Transformador de corrente HMCT103C.

Fonte: (AUTO CORE ROBÓTICA, 2018)

Figura 2.6: Funcionamento do transformador de corrente toroidal.



Fonte: (TEST GUY, 2018)

Modelo	HMCT103C
Relação de Transformação	1000:1
Corrente de entrada	0-5A
Corrente de saída	0-5mA
Resistência de Amostagem	200 Ω
Tensão de amostragem	1V
Não Linearidade	≤0,3%
Frequência de trabalho	20Hz-20KHz
Temperatura de operação	-55°C∼+85°C

Tabela 2.3: Parâmetros técnicos HMCT103C

A partir dessas informações, pode-se observar que a frequência de trabalho do dispositivo está de acordo com a frequência da rede elétrica, em que para a nossa região é de aproximadamente 60Hz. Além disso, temos a relação de corrente na saída do dispositivo, que é de 1mA para cada 1A na entrada.

Ao comparar ambas as alternativas explicadas acima, em relação ao ACS712 o transformador de corrente toroidal tem a vantagem de não ser um elemento intrusivo, posto que o ACS712 não possui isolamento reforçado para tensões acima de 184V de pico. Por esse motivo o HMCT103C foi o dispositivo empregado no presente trabalho para a função de medir a corrente.

2.7 Trabalhos Correlatos

Diversos trabalhos da literatura técnica têm sido dedicados a desenvolver soluções para o controle de adaptadores para automação residencial. Em [21], por exemplo, Yaowaluk e Wanchalerm desenvolveram uma proposta com o mesmo objetivo do presente projeto, ou seja, um adaptador com função de liga/desliga e monitoramento do consumo. Porém, apesar de ter sido utilizado um módulo Wifi (ESP-WROOM02) a conexão era feita direta com o Access Point e o dispositivo, sem utilizar o serviço de nuvem.

Em outra proposta [22] buscou-se o controle de dispositivos para automação residencial por comunicação Bluetooth. Contudo, a alternativa também não envolve a utilização da nuvem. Por fim, em [23] utiliza-se o serviço de nuvem para o desenvolvimento de aplicações Web, mas a aplicação é focada em AVAC (aquecimento, ventilação e ar condicionado) e no

controle do seu status com a mínima intervenção humana.

CAPÍTULO 3

DESENVOLVIMENTO DO PROJETO

Este capítulo é dedicado a apresentação das etapas de desenvolvimento do presente trabalho. Primeiramente, na Seção 3.1, as etapas do projeto do *hardware* são apresentadas,assim como os detalhes do esquemático dos circuitos e design do *layout* da placa de circuito impresso. Em seguida, na Seção 3.2 os processos de desenvolvimento da aplicação Web, do servidor HTTP, assim como a estrutura do banco de dados estão descritos com detalhes. A Seção 3.3, por sua vez, demonstra os procedimentos de desenvolvimento do *firmware*, incluíndo a comunicação MQTT, o credenciamento do adaptador com a AWS, e a aplicação das sombras do dispositivo com a AWS IoT. Por último, a Seção 3.4 demonstra o cálculo realizado para medir a potência do dispositivo conectado ao adaptador de tomada.

3.1 Desenvolvimento do Hardware

O projeto de desenvolvimento do hardware foi dividido em:

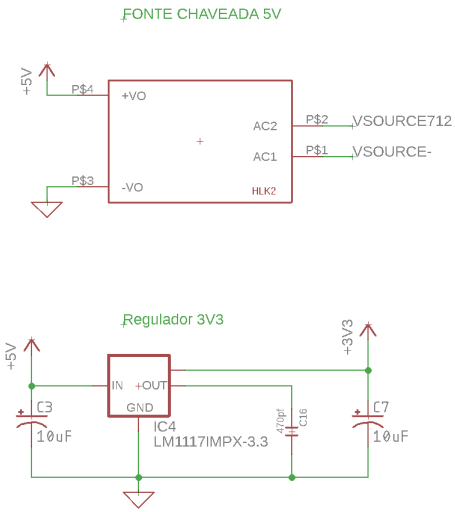
- Fontes de alimentação;
- Conversor USB - Serial;
- Circuito de ativação;
- Sensor de Corrente.

Cada item da lista será descrito nas seções em sequência.

3.1.1 Fontes de Alimentação

Tendo em vista que o adaptador inteligente será conectado diretamente à rede, uma fonte chaveada TSP-05 é utilizada, a qual possui uma tensão de 5Vdc de saída para uma tensão de entrada de 100-240Vac. Essa tensão será utilizada para conectar o relé do circuito de ativação. Além da fonte chaveada, o regulador de tensão LM1117 deu-se como necessário por regular a tensão de 5V para 3.3V, cuja utilidade é de alimentar o microcontrolador ESP8266. O esquemático para as fontes de alimentação pode ser visto na Figura 3.1.

Figura 3.1: Esquema das fontes de alimentação.



Fonte: Elaborada pelo autor

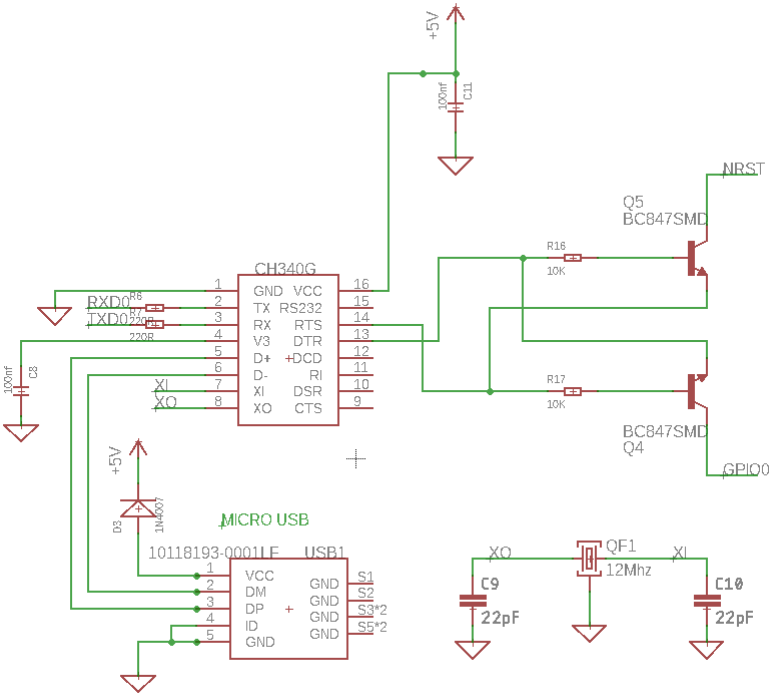
3.1.2 Conversor USB - Serial

Para a programação do ESP8266 do adaptador inteligente, o conversor USB Serial CH340g foi adicionado ao circuito junto a um conector USB Micro, visando permitir a conexão com o computador. O circuito de tal conversor pode ser visualizado na Figura. Para o funcionamento do chip, adicionou-se um oscilador externo de 12MHz e um circuito para fazer a lógica de programação do ESP8266. Essa lógica foi baseada na utilizada no circuito da placa de desenvolvimento de código aberto NodeMCU [12], na qual os terminais RTS e DTR ajustam os pinos de NRST e GPIO do módulo Wi-Fi, conforme descrito na Tabela 3.1.

DTR	RTS	RST	GPIO0
1	1	1	1
0	0	1	1
1	0	0	1
0	1	1	0

Tabela 3.1: Relação sinais DTR, RTS, RST e GPIO

Figura 3.2: Esquema do conversor USB - Serial.



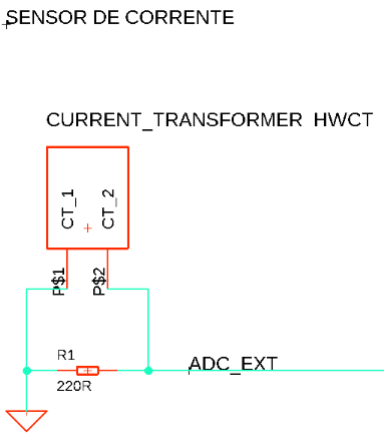
Fonte: Elaborada pelo autor

3.1.3 Sensor de Corrente

Como explicado na Seção 2.6.3, o dispositivo HMCT103C foi escolhido para a função de medir a corrente do dispositivo conectado ao adaptador de tomada. O sensor foi conectado a um resistor de 220Ω como ilustrado na Figura 3.3. Uma vez que a saída do transformador de corrente gera $1mA$ para cada $1A$ medido, ao adicionar um resistor de 220Ω temos $200mV$ para cada ampere, segundo a lei de ohm. Sendo assim, tendo em vista que o tamanho do passo do conversor analógico digital do ESP8266 é de $3.3mV$, temos que a corrente mínima a ser medida é dada por:

$$Corrente\ Mínima = \frac{Tamanho\ do\ passo}{Tensão\ na\ saída}.A \tag{3.1}$$

Figura 3.3: Circuito do sensor de corrente.

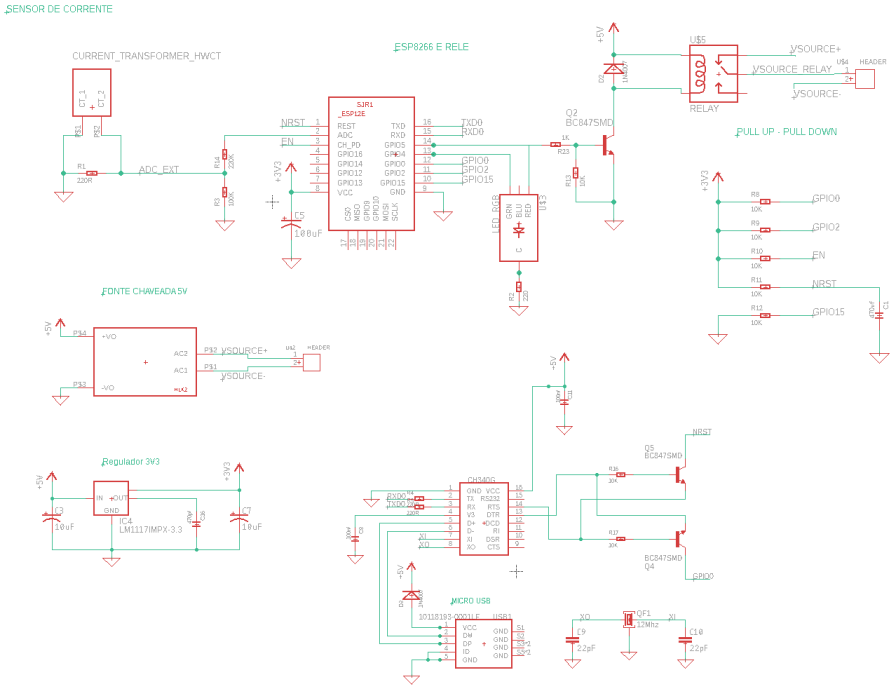


Fonte: Elaborada pelo autor

3.1.4 Circuito Completo

O esquemático do circuito completo pode ser visualizado na Figura 3.4. Nele, além dos blocos de circuito explicados anteriormente, o esquemático completo contém o circuito para o acionamento do relé e um LED RGB, empregado com o intuito de fornecer uma resposta com relação a ativação do relé.

Figura 3.4: Esquemático do circuito completo.



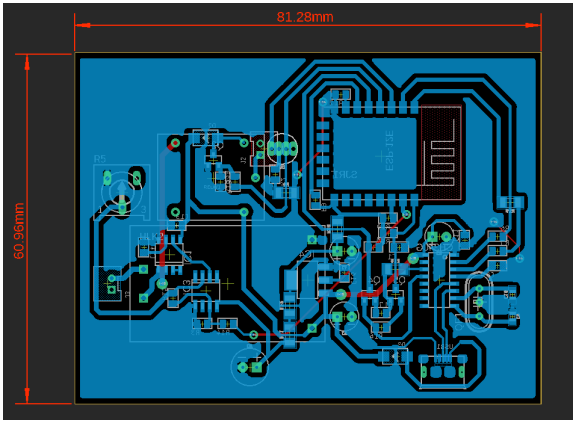
Fonte: Elaborada pelo autor

3.1.5 Projeto da Placa de Circuito Impresso

O projeto do layout da placa de circuito impresso foi realizado através do software Eagle, assim como o esquemático do circuito. Com a criação de um layout para a placa foi possível alcançar duas alternativas de projeto. A primeira delas possui o circuito ACS712 para medição de corrente e teve o visível diferencial por ser realizada em apenas uma *layer*, facilitando sua produção. Esse primeiro layout pode ser visualizado na Figura 3.5.

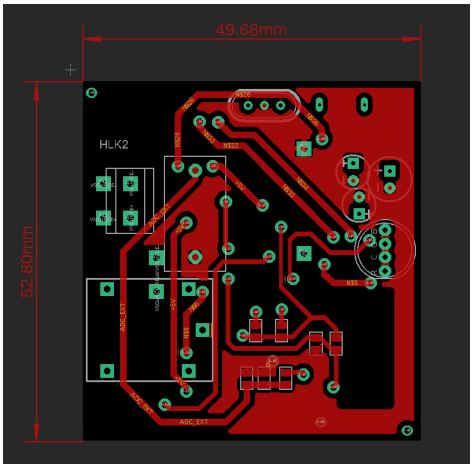
Entretanto, com a verificação de alguns erros relativos a distância entre os pinos de alta tensão e a mudança do tamanho de *pads* de componentes, foram realizados ajustes no circuito, e por consequência o layout da placa foi totalmente reprojetado, o que resultou na criação de uma segunda placa em duas *layers*. Esse segundo e último layout da *top layer* pode ser visto na Figura 3.6, enquanto o layout final da *bottom layer* pode ser visualizado na Figura 3.7.

Figura 3.5: Primeira PCB de uma camada.



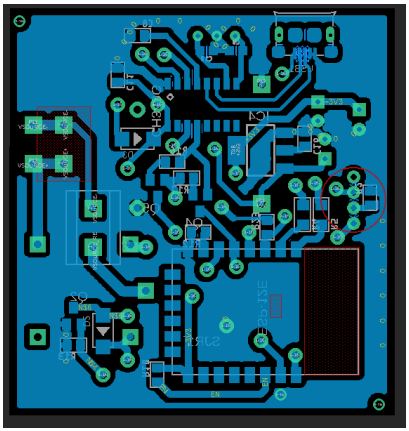
Fonte: Elaborada pelo autor

Figura 3.6: Segunda PCB *top layer*.



Fonte: Elaborada pelo autor

Figura 3.7: Segunda PCB *bottom layer*.

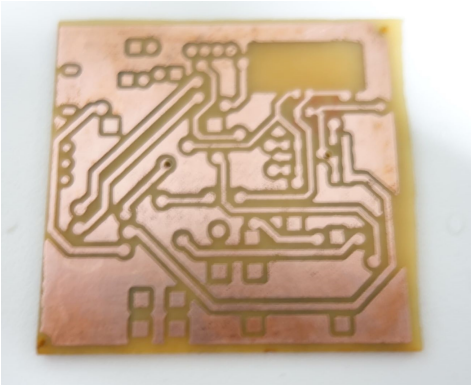


Fonte: Elaborada pelo autor

O protótipo físico da placa antes da colocação dos com-

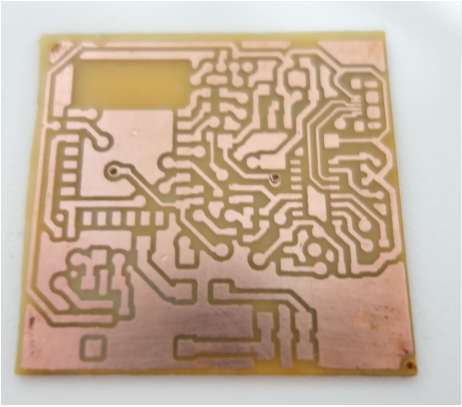
ponentes pode ser observado nas Figuras 3.8 e 3.9. Após a perfuração e solda dos componentes a placa final pode ser vista nas Figuras 3.10 e 3.11.

Figura 3.8: PCB antes da colocação de componentes (*Top Layer*).



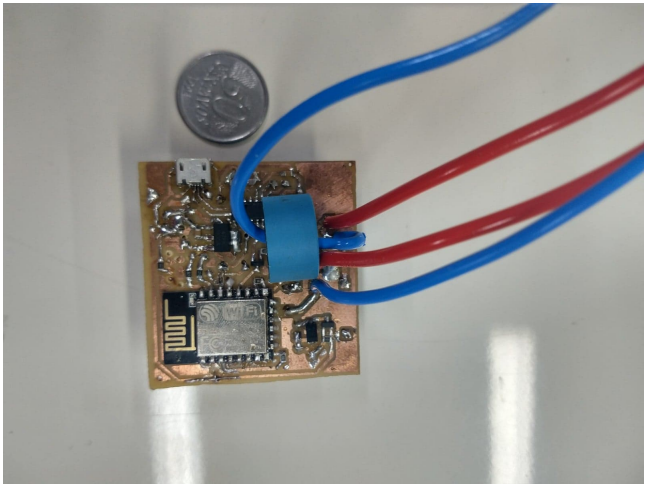
Fonte: Elaborada pelo autor

Figura 3.9: PCB antes da colocação de componentes (*Bottom Layer*).



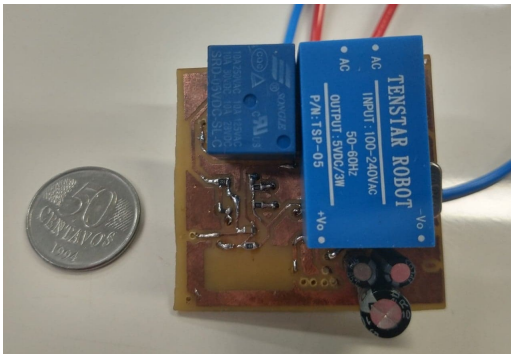
Fonte: Elaborada pelo autor

Figura 3.10: PCB após a colocação de componentes (*Bottom Layer*).



Fonte: Elaborada pelo autor

Figura 3.11: PCB após a colocação de componentes (*Top Layer*).



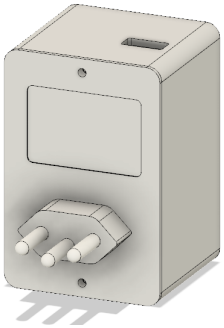
Fonte: Elaborada pelo autor

A mudança de layout refletiu em uma diminuição de área de 49,54cm^2 para 26,23 cm^2, o que representa uma diminuição de aproximadamente 53%.

3.1.6 Design do Modelo 3D do adaptador de tomada

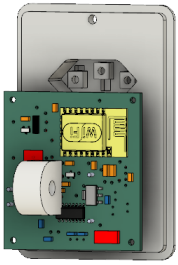
O design do modelo 3D do adaptador de tomada foi realizado com o auxílio do software AutoCad 2019, com acesso via conta do tipo estudante. A parte frontal do modelo renderizado pode ser vista nas Figuras 3.12 e 3.13, enquanto a parte traseira pode ser vista nas Figuras 3.14 e 3.14. O desenho dimensional do modelo está apresentado no Apêndice A.

Figura 3.12: Modelo 3D renderizado do adaptador de tomada (Parte Frontal).



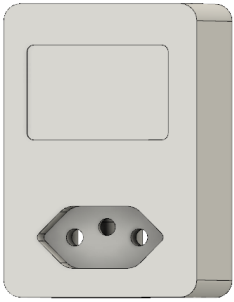
Fonte: Elaborada pelo autor

Figura 3.13: Modelo 3D renderizado do adaptador de tomada (Parte Frontal Aberta).



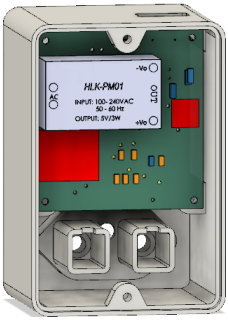
Fonte: Elaborada pelo autor

Figura 3.14: Modelo 3D renderizado do adaptador de tomada (Parte Traseira).



Fonte: Elaborada pelo autor

Figura 3.15: Modelo 3D renderizado do adaptador de tomada (Parte Traseira Aberta).



Fonte: Elaborada pelo autor

A partir do seu modelo, a caixa do adaptador foi impressa por uma impressora 3D modelo Ender 3. As fotos do modelo impresso se encontram nas figuras 3.18, 3.18 e 3.18.

Figura 3.16: Modelo 3D impresso do adaptador de tomada (Parte Interna).

Fonte: Elaborada pelo autor

Figura 3.17: Modelo 3D impresso do adaptador de tomada (Parte Externa).

Fonte: Elaborada pelo autor

Figura 3.18: Modelo 3D impresso do adaptador de tomada.

Fonte: Elaborada pelo autor

3.1.7 Lista de componentes

A lista de componentes gerada para o projeto pelo software Eagle pode ser visualizada na Tabela 3.2.

Dispositivo	Encapsulamento	Código	Valor	Quantidade
ESP8266-12EESP12E	ESP8266-ESP12E	SJR1		1
SMD-RES-10K-1%-1/8W(0805)	R0805	R3	100K	1
CERAMIC-100NF-50V-10%-X7R(0805)	C0805	C8, C11	100nf	2
100UF-POLAR-RADIAL-2.5MM-25V-20%	CPOL-RADIAL-2.5MM-6.5MM	C5	100uF	1
MICRO-USB-SMD-B-(10118193-0001LF)	MICRO-USB5+6P-SMD-0.65-B	USB1		1
SMD-RES-10K-1%-1/8W(0805)	R0805	R8, R9, R10, R11, R12, R13, R16, R17	10K	8
10UF-POLAR-RADIAL-2.5MM-KIT-25V-20%	CPOL-RADIAL-2.5MM-5MM-KIT	C3, C7	10uF	2
10M7AHC49T	HC49U	QF1	12Mhz	1
SMD-RES-1K-5%-1/8W(0805)	R0805	R23	1K	1
DIODE-GEN-PURPOSE-1KV-1A(DO-214AC)	DO-214AC	D2, D3	1N4007	2
SMD-RES-1K-5%-1/8W(0805)	R0805	R2	220	1
SMD-RES-10K-1%-1/8W(0805)	R0805	R14	220K	1
SMD-RES-13R-5%-1/4W(1206)	R1206	R4, R5	220R	2
SMD-RES-220R-5%-1/8W(0805)	R0805	R1	220R	1
22PF-0603-50V-5%	603	C9, C10	22pF	2
CERAMIC-100NF-50V-10%-X7R(0805)	C0805	C16	470pf	1
CERAMIC-100NF-50V-10%-X7R(0805)	C0805	C1	470uf	1
BC847SMD	SOT23	Q2, Q4, Q5		3
CH340G	SOIC16	U$15		1
CURRENT_TRANSFORMER_HWCT		U$1		1
HEADER	H2-3.5-7.0X7.0MM	U2, $U4$		2
HLK-PM013W	HLK-PM01	HLK2		1
LED_RGB	LED4-2.0-D8.0MM	U$3		1
LM1117IMPX-3.3	SOT223	IC4		1
RELAY		U$5		1

Tabela 3.2: Lista de componentes

3.2 Desenvolvimento da Aplicação Web e Servidor

O desenvolvimento das aplicações Web e do servidor foram divididos nas seguintes etapas:

- Criação de usuário;
- Serviço de Autenticação;
- Criação do dispositivo;
- Lista de dispositivos;
- Informação do dispositivo;
- Banco de dados.

Cada etapa será explicada nas subseções seguintes.

3.2.1 Criação de Usuário

3.2.1.1 Front End

Para o presente projeto, um serviço de autenticação para o login do usuário em plataforma web foi desenvolvido. O funcionamento da autenticação se dá primeiramente no ato de registro e criação de uma conta de usuário e senha na aplicação web, do utilizador do produto. A tela referente ao preenchimento do formulário pode ser vista na Figura 3.19.

Para o desenvolvimento do *front end* da criação do usuário três bibliotecas, foram necessárias em conjunto com o React e MobX. São elas:

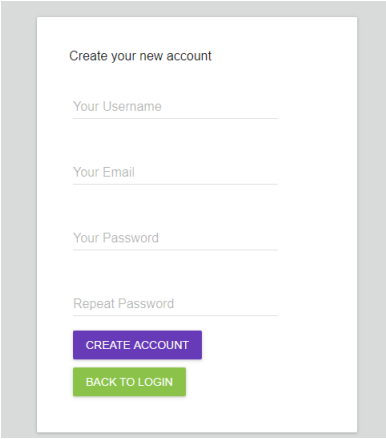
- validatorjs [25];
- axios [26];
- react-notify-toast [27];
- material-ui [28].

A biblioteca validatorjs foi implementada com o objetivo de facilitar a verificação de consistências no formulário preenchido. Assim, é possível criar regras para cada campo do formulário e verificar se os parâmetros preenchidos estão de acordo com as regras criadas. Dessa forma, para o formulário em questão foram criadas regras para que todos os campos sejam obrigatórios, para que o campo Email seja preenchido exclusivamente através de email válido, e para que a senha tenha no mínimo 6 caracteres, também de acordo com o campo de repetir senha (por fins de verificação).

A biblioteca read-notify-toast foi inserida apenas para mostrar um *feedback* visual no topo da tela, para informar ou o erro ou o sucesso da operação. Já o material-ui foi empregado para que os componentes da aplicação estejam de acordo com os padrões do Google Material Design [10].

Por fim, o axios foi a biblioteca aplicada para realizar a requisição de HTTP POST para o servidor, enviando as informações para a criação do usuário no corpo da requisição.

Figura 3.19: Tela de criação de usuário.



Fonte: Elaborada pelo autor

3.2.1.2 Back End

Para o desenvolvimento do *back end* da criação do usuário, as seguintes bibliotecas foram utilizadas no servidor em Node.js:

- dynamodb [31];
- joi [29];

- bcryptjs[30].

A biblioteca dynamodb implementa as operações de CRUD (Criar, Ler, Atualizar e Remover) para o banco de dados DynamoDB da AWS, assim como a criação de tabelas. A mesma foi empregada junto à biblioteca joi, para a definição do modelo do usuário no banco de dados, sendo definido com um objeto Javascript, como parâmetro da função de definição do modelo. Isso pode ser observado a seguir:

```
const User = dynamo.define('User', {
   hashKey: 'username',
   timestamps: true,
   schema: {
      username: Joi.string().alphanum().min(3).max(30)
         .required(),
      email: Joi.string().email(),
      password: Joi.string()
   }
})
```

Sendo assim, o modelo foi definido de acordo com a Seção 3.2.3. O usuário é criado pela requisição HTTP POST para o servidor, nele um *middleware* é utilizado para a verificação dos parâmetros inseridos. Posteriormente, a senha do usuário é empregada como parâmetro da função responsável por gerar o *Salt* do algoritmo BCrypt (biblioteca 'bcryptjs'). Por fim, o *Salt* é transformado em *hash* para ser armazenado no banco de dados junto ao nome do usuário.

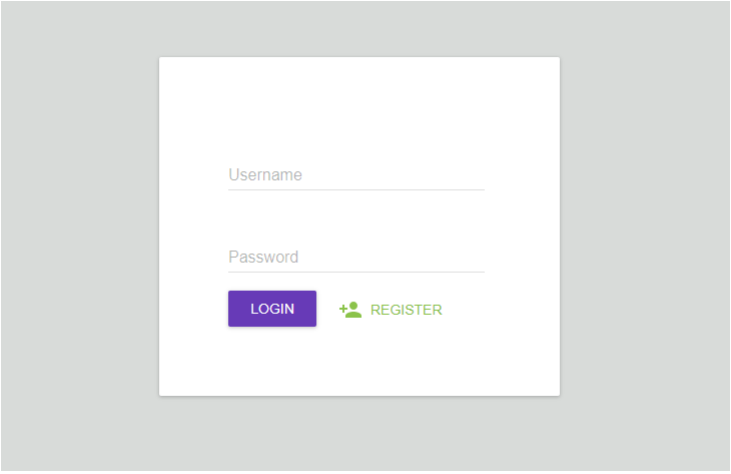
3.2.2 Serviço de Autenticação

3.2.2.1 Front End

Para o *Front end* do serviço de autenticação (Login) as mesmas bibliotecas de validação de formulário e criação de componentes são empregadas. Ao preencher o formulário (Figura 3.20), uma requisição HTTP POST é realizada para o *endpoint* '/auth/login'. Em seguida, de maneira assíncrona a resposta é aguardada pela aplicação.

A resposta do servidor contém um Json Web Token, o mesmo é guardado no Local Storage do navegador para a persistência do usuário. Após esses processos, ao JWT recebido pelo servidor é adicionado o parâmetro 'Authorization' das requisições HTTP da biblioteca axios, com o propósito de sempre que for enviar uma requisição ao servidor a informação do token do usuário conectado ser enviada em conjunto.

Figura 3.20: Tela de login.



Fonte: Elaborada pelo autor

3.2.2.2 Back End

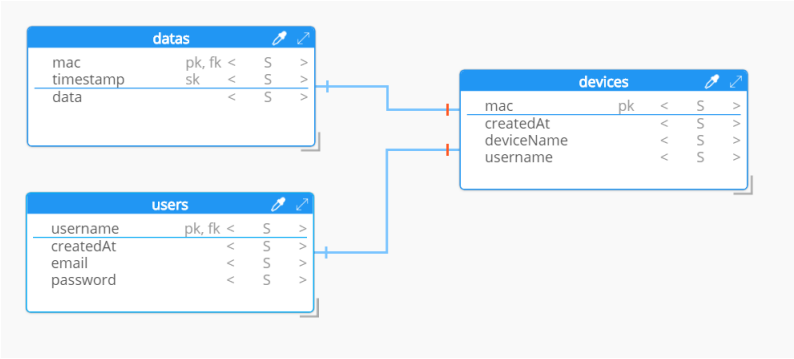
Ao realizar o Login, o servidor recebe as informações do usuário via requisição HTTP POST, o que antecede a realização de uma pesquisa no banco de dados DynamoDB da AWS, para obter o modelo do usuário. O processo posterior é a verificação do servidor, com o DynamoDB, da já existência do usuário no banco. Caso for verificada que já existe a Hash guardada no processo de criação do usuário e a senha inserida pelo mesmo, essas informações são adicionadas no parâmetro da função de comparação do algoritmo BCrypt. Caso a comparação não retorne erro, o nome do usuário é adicionado a uma assinatura de JSON Web Token criptografada, com uma chave privada do servidor. O token da assinatura JWT então é retornada ao *front end*, onde deve ocorrer o registro do mesmo para a persistência da autenticação.

A persistência da autenticação é feita aplicando um *middleware* a todas as rotas que requerem autenticação. Esse *middleware* verifica os *Headers* da requisição HTTP e faz a verificação do token recebido pela *front end* junto à chave secreta do servidor. Caso o token seja inválido, o servidor retorna uma resposta 401 informando falha de autenticação.

3.2.3 Banco de dados

O banco de dados empregado no presente trabalho foi o DynamoDB disponível na AWS. O diagrama entidade relacionamento do modelo (Realizado com auxílio do software Hackolade) aplicado ao banco de dados em questão pode ser visto na Figura 3.21.

Figura 3.21: Diagrama entidade relacionamento do banco de dados.



Fonte: Elaborada pelo autor

Os dados foram organizados a partir de uma divisão em três tabelas:

- datas;
- users;
- devices.

A tabela *datas* armazena as informações do estado atual do dispositivo, cujas informações são processadas diretamente pelas regras adicionadas na AWS IoT, além de adicionadas ao campo data como um objeto Javascript, contendo a informação do status do dispositivo (Ligado ou desligado) e a potência de consumo atual. Tratando-se de uma tabela utilizada pelo *front end* para gerar o gráfico de consumo. Também são registradas as informação referentes ao tempo atual e o endereço MAC do dispositivo, o qual é empregado como chave estrangeira para a tabela *devices*.

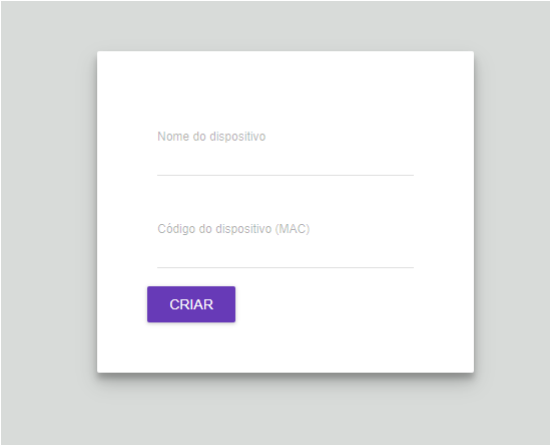
Na tabela *devices* constam as informações do endereço MAC como chave primária (sendo que o mesmo deve ser único), o nome do dispositivo como 'deviceName', o nome do usuário que possui o dispositivo como 'username' e o tempo em que usuário registrou esse dispositivo no campo 'createdAt'. Essa tabela tem a função de registrar todos os dispositivos que foram uma vez inscritos no website e quem os possui.

Por último, a tabela *users* possui o campo 'username', como chave primária (tendo em vista que o usuário deve ter registro único) e estrangeira, o campo 'createdAt' que informa o momento que o usuário foi criado para registro da informação do momento que o usuário foi criado, o campo email para o registro de email e, por fim, o campo password que registra um *Hash* gerado pelo algorítimo BCrypt, para autenticar o usuário no momento do login.

3.2.4 Registro do dispositivo

Uma vez que o usuário esteja autenticado no *website*, o mesmo pode adicionar dispositivos em uma lista, ao pressionar o botão de *adicionar dispositivos* o qual leva à tela apresentada na Figura 3.22. Nessa tela, o usuário deve adicionar o nome do dispositivo e o endereço MAC do mesmo: ambas as informações são enviadas para o servidor que salva o registro na tabela 'devices', junto ao nome do usuário.

Figura 3.22: Tela de registro de dispositivos.



Fonte: Elaborada pelo autor

3.2.5 Lista de dispositivos

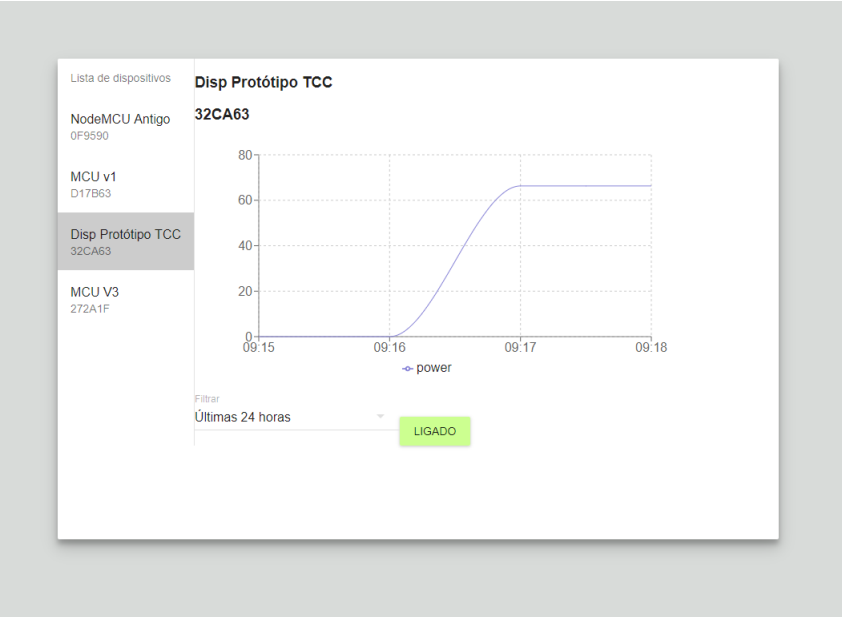
Posto que o dispositivo foi registrado com sucesso, uma lista com os dispositivos adicionados aparecerá, o que pode ser observado na Figura 3.23. O usuário então poderá selecionar qual dos dispositivos deseja verificar na barra lateral esquerda, para assim que o selecionar ser possível observar o gráfico de Potência Consumo ao longo do Tempo.

Na seleção ainda é possível escolher o estado do dispositivo conectado ao adaptador (por meio do botão de *ligado* ou *desligado*), e assim filtrar os resultados do gráfico ou pelas últimas 24 horas ou pela última hora.

Para o *front end* montar a lista de dispositivos, o mesmo envia uma requisição para o servidor via a biblioteca 'axios' para o *endpoint* '\device\list'. Uma vez que o servidor recebe a requisição, o *middleware* de autenticação verifica se o usuário é valido, com a qual as informações do usuário serão

posteriormente transmitidas para o método *listByUser*, do módulo *DeviceManager*. Com essas informações, o servidor realiza uma busca pelo nome do usuário na tabela *devices* e obtém o endereço MAC e o 'deviceName'. Por fim essas informações retornam para o *front end* e são mostradas na tela.

Figura 3.23: Lista de dispositivos.



Fonte: Elaborada pelo autor

3.3 Desenvolvimento do Firmware

O *firmware* do ESP8266 foi desenvolvido em Javascript com auxílio do Mongoose OS, assim é mantida uma persis-

tência de linguagem em todo espectro do desenvolvimento. Tratando-se do *firmware*, destacam-se as seguintes etapas:

- Credenciamento com AWS IoT;
- Comunicação MQTT;
- Aplicação de sombras do dispositivo da AWS IoT.

As etapas serão explicadas individualmente a seguir.

3.3.1 Credenciamento com AWS IoT

Para realizar o credenciamento do dispositivo com a AWS primeiramente é necessário criar uma AWS IoT *policy*. As *policies* são objetos no formato JSON, que são armazenados pela Amazon com o intuito de adicionar permissões específicas para cada tipo de acesso. Sendo assim, são impossibilitados acessos indesejáveis. Para o projeto em questão a *policy* empregada para os certificados dos dispositivos foi a padrão do Mongoose OS:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ],
  "Version": "2012-10-17"
}
```

Uma vez que a *policy* foi criada, para cada dispositivo criado, o Mongoose OS adiciona uma chave privada, uma pública e

um certificado de autoridade único, nos padrões de Certificado X.509. O certificado também é atribuído à instância do dispositivo dentro da interface da AWS IoT, assim como a *policy* criada anteriormente.

Na *policy* padrão todos os acessos ao serviço de IoT são permitidos. Contudo, os acessos a outros serviços, como ao de banco de dados por exemplo, não são permitidos.

3.3.2 Comunicação MQTT

A comunicação via protocolo MQTT foi realizada de forma a empregar a AWS IoT como *broker*. O *firmware* do dispositivo foi configurado de forma a enviar informações relativas a consumo de potência do dispositivo e o tempo em que o dado foi medido no formato JSON. Essas informações são enviadas via protocolo MQTT para o *endpoint* da AWS, que pode ser obtido no site do mesmo. A URL possui o seguinte formato:

```
mqtts://<endpoint>.iot.us-east-1.amazonaws.com:8883
```

No endereço citado acima o <endpoint> representa o *endpoint* específico da conta da AWS. O envio das informações é feito a cada 30 segundos pelo ESP8266 no tópico 'power/MAC', na qual MAC representa o endereço MAC de cada adaptador. A mensagem enviada ao tópico em questão é realizada da seguinte maneira:

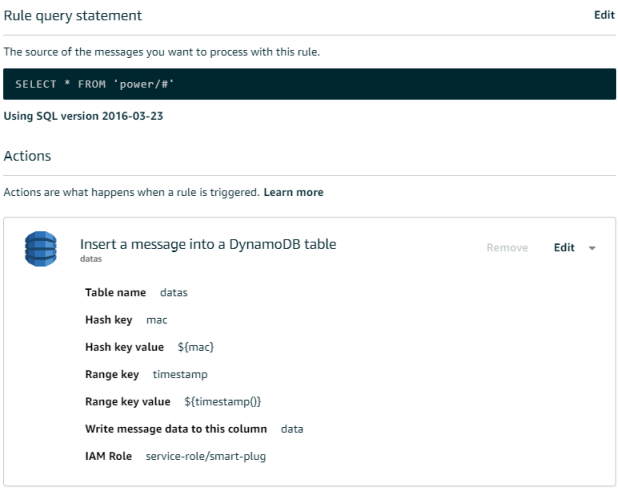
```
let message = JSON.stringify({
    mac: Cfg.get('device.id'),
    power: power
});
```

Para lidar com o recebimento de informação, no *console* da AWS IoT foi criada uma regra chamada *sendStatus*, que ao receber uma publicação no tópico 'power/#' (na qual representa qualquer informação), é inserida a mensagem diretamente no banco de dados DynamoDB, sem a necessidade do uso servidor. A regra criada separa as informações recebidas nas seguintes colunas do banco de dados:

- mac;
- data;
- timestamp.

A edição da regra realizada no console da AWS pode ser vista na Figura 3.24.

Figura 3.24: Regra 'sendStatus' da AWS.



(AMAZON, 2018)

3.3.3 Aplicação de Sombras do Dispositivo da AWS IoT

A *sombra do dispositivo*, termo descrito anteriormente neste trabalho, foi aplicada com o intuito de ativar o adaptador ao pressionar os botões do dispositivo no *website* de forma persistente, ou seja, a AWS IoT mantém o registro do último estado do dispositivo e qual o estado desejado pelo usuário. Caso o adaptador possua falha de conexão com a internet, a AWS garante a tentativa de envio até que o estado desejado seja igual ao estado atual do adaptador. Para tal, inicialmente no *front end* na tela de informações do dispositivo foi adicionado um botão contendo o estado ou 'ligado' ou 'desligado'. Assim, ao usuário pressionar o botão, uma requisição HTTP POST é feita ao servidor no endpoint '/device/setstate' com um objeto JSON contendo o endereço MAC do dispositivo e o novo estado do relé, como pode ser visto a seguir:

```
const deviceData = {
      mac: this.selectedDevice.mac,
      newState: {
            relay: newState
      }
}
```

O servidor por sua vez utiliza a SDK da Amazon 'aws-iot-device-sdk' para implementar a *sombra do dispositivo*. Ao receber a requisição do *front end*, o servidor verifica se o dispositivo recebido existe no banco de dados, cria um objeto JSON contendo o novo estado do relé no seguinte formato:

```
var relayState = {
   'state': {
```

```
      'desired': {
         'relay': newState.relay
      }
   }
}
```

Em seguida, realiza-se o registro do dispositivo com o endereço MAC recebido e atualiza-se as informações da *sombra do dispositivo*. Por fim, caso a atualização retorne um *token* não nulo do servidor da Amazon, a comunicação com a *sombra* é finalizada.

3.4 Cálculo da Potência

O cálculo de potência foi realizado de forma a se obter apenas a potência aparente, ou seja, considerando uma carga puramente resistiva. Esse tipo de cálculo não representa o consumo real, uma vez que produtos eletrônicos podem possuir um fator de potência diferente de 1, gerando uma diferença entre a potência real e a aparente. Para o cálculo, levou-se em consideração o somatório da potência aparente que o próprio adaptador consome, com a do dispositivo conectado ao adaptador. Uma vez que a medida é realizada a partir da tensão alternada da rede, para o cálculo primeiramente é verificado o valor lido pelo conversor analógico digital do ESP8266 (lembrando que trata-se de um conversor ADC de 10 bits),e sendo assim possui 1024 valores possíveis.

Para medir a tensão de pico do sensor, a função chamada getVPP foi criada. Nela é realizada a medição do conversor ADC durante 1 segundo, obtendo o valor do ponto máximo lido nesse tempo. Com esses valores a seguinte fórmula é

aplicada:

$$V_{pp} = \frac{Valor_{max} \cdot V_{maxADC}}{2^{10}} = \frac{Valor_{max} \cdot 1}{1024} \tag{3.2}$$

Posteriormente, o valor de tensão de pico é convertido para tensão RMS, sendo assim:

$$V_{RMS} \approx V_{pp} \cdot 0.707. \tag{3.3}$$

Por conseguinte, a tensão medida pelo sensor é empregada no cálculo da corrente RMS consumida pelo adaptador somada ao dispositivo conectado no mesmo. O cálculo leva em consideração o fator de conversão do sensor e o resistor ligado no mesmo, sendo assim:

$$I_{RMS} = \frac{V_{RMS} \cdot 1000}{R} = \frac{V_{RMS} \cdot 1000}{220\Omega} \tag{3.4}$$

Por fim, sendo a tensão da rede da região de aproximadamente $220V_{RMS}$, temos:

$$P_{aparente} = V_{RMS} \cdot I_{RMS} = 220 \cdot I_{RMS}. \tag{3.5}$$

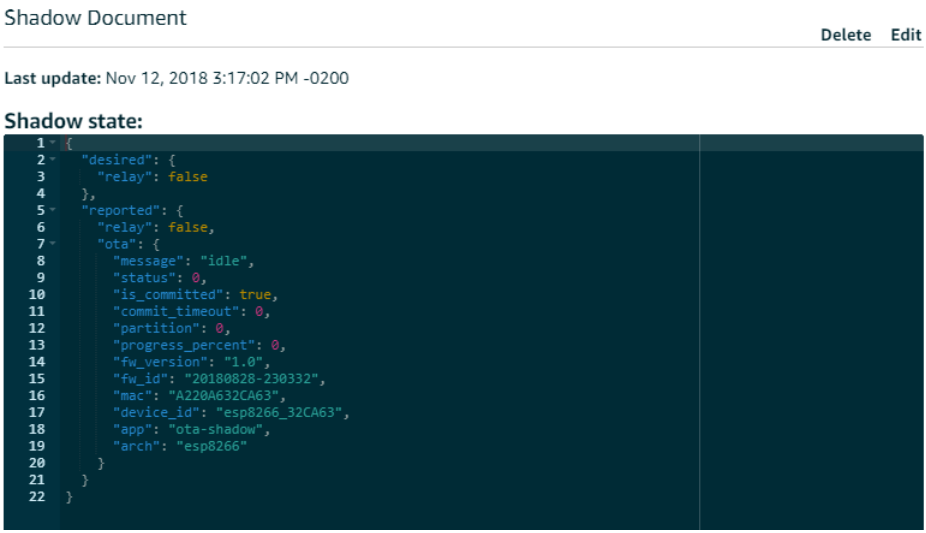
CAPÍTULO 4

TESTES E RESULTADOS

Anteriormente à construção do protótipo, foram realizados testes quanto à comunicação entre o dispositivo e a sombra do dispositivo (AWS IoT) e quanto à comunicação do servidor Node.js com a AWS via MQTT. Para isso, a interface da AWS console foi utilizada, onde primeiramente partindo da placa de desenvolvimento NodeMCU, foi criado o certificado e a *policy* do dispositivo. Após o firmware ser carregado no NodeMCU, foram realizados testes quanto à alteração do campo "desired" na interface de teste da AWS (Figura 4.1).

O campo "desired" reflete a intenção da mudança a ser realizada. Sendo assim, o valor do campo foi enviado ao dispositivo, retornando o valor alterado no campo "reported". Portanto, o comportamento esperado para a comunicação foi observado com sucesso.

Figura 4.1: Ambiente de teste de sombra do dispositivo AWS.



Fonte: Elaborada pelo autor

Posteriormente, um teste similar foi realizado, porém com o *front end* realizando a requisição de alteração do campo "desired" para o *back end*, esse por sua vez enviando a requisição a AWS via SDK 'aws-iot-device-sdk'. O envio da requisição chegou com sucesso na interface da AWS e as informações foram enviadas corretamente ao dispositivo.

Já para a análise do *hardware*, uma sequência de medições foi realizada com auxílio de um multímetro. Uma vez que o método de medição leva em conta apenas cargas puramente resistivas, uma lâmpada de 60W e uma lâmpada de 100W foram utilizadas como carga, resultando nas medidas descritas na Tabela 4.1.

Estado do adaptador	Corrente Medida (mA)	Tensão (V_{RMS})	Potência (VA)
Lampada 60W Conectada	260mA	∼220	57,2
Lâmpada 100W Conectada	420mA	∼220	92,4
Relé Ligado sem Lâmpada	4,5mA	∼220	0,99
Relé Desligado	2,3mA	∼220	0,51

Tabela 4.1: Medições do consumo do adaptador de tomada em diferentes condições

Em seguida, aforam realizadas medições usando o adaptador proposto, as quais podem ser visualizadas na Tabela 4.2. Realizando a média aritmética dos valores obtidos na coluna "Lâmpada 100W" temos uma potência aparente de 95,13VA, já para a coluna "Lâmpada 60W" temos uma potência aparente de 63,09VA.

A medida da potência apenas das lâmpadas conectadas (Sem o adaptador) também foi realizada, o resultado foi o mesmo da Tabela 4.1, ou seja, 57,2VA para a lâmpada de $60W$ e 92,4VA para a lâmpada de $100W$.

Sendo assim, o erro obtido comparando o valor obtido pelo multímetro e a média medida pelo o adaptador, para a lâmpada de 60W, foi de 10,3%. Já para a lâmpada de 100W o erro obtido foi de 2,8%.

Potência medida pelo adaptador (VA)	
Lâmpada 100W	**Lâmpada 60W**
96.66	63.51
95.96	62.14
94.58	64.20
95.96	62.82
95.27	63.51
94.58	62.82
95.27	64.20
91.82	62.82
95.27	62.13
95.96	62.82

Tabela 4.2: Tabela de medições realizadas pelo adaptador de tomada

CAPÍTULO 5

CONCLUSÃO

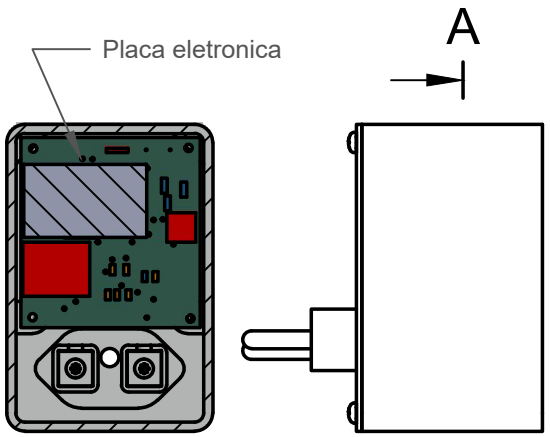
O presente projeto validou a possibilidade do desenvolvimento de um adaptador de tomada conectado a internet, com controle de acionamento e medidor de energia, utilizando um microcontrolador de baixo custo. O projeto também apresentou soluções, como a AWS IoT, que incluem nativamente técnicas para aumentar a segurança dos dispositivos conectados a internet, como o certificado x509. Também foi possível aplicar técnicas de criptografia (Bcrypt) para aprimorar a segurança a autenticação da aplicação web criada, assim como aplicar JSON Web Tokens para a persistência da autenticação. O projeto também demonstrou viável a aplicação da placa de circuito impresso a um modelo de adaptador real, obtido via impressão 3D.

5.1 TRABALHOS FUTUROS

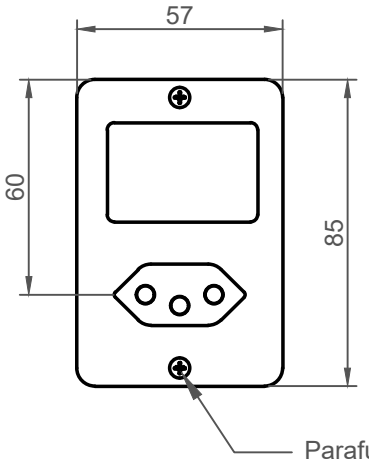
O presente trabalho possuiu um foco maior na etapa de construção da infraestrutura IoT, relacionada a comunicação dos dispositivos à nuvem e ao *website*. Sendo assim, algumas mudanças na etapa de medição são sugeridas para um trabalho futuro, uma vez que, embora a medição do consumo de potência tenha-se demonstrado possível, foi obtido um erro de 10,3% para a carga menor (lâmpada de 60W) considerando apenas cargas resistivas.

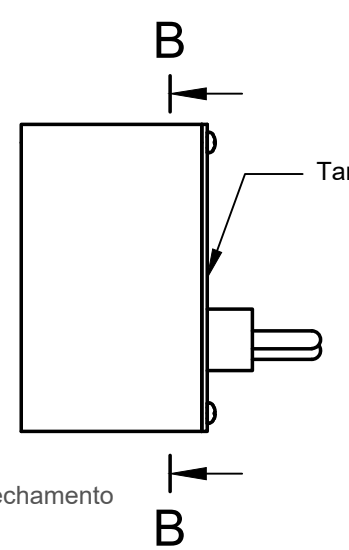
Para tais trabalhos futuros, alguns aperfeiçoamentos devem ser realizados na etapa de medição, como por exemplo, a medição da potencia real de consumo, levando em consideração tanto a onda de tensão quanto de corrente, assim como a defasagem entre elas. Além disso, é aconselhável a realização de uma pesquisa por outros métodos de medição que visem obter erros menores quanto ao consumo de potência.

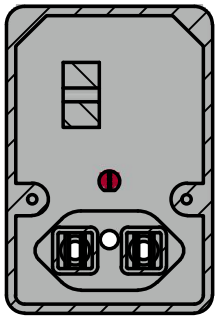
APÊNDICE A



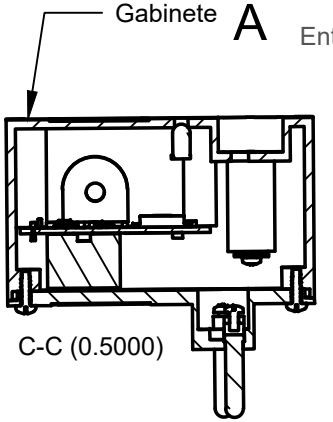
B-B (0.5000)



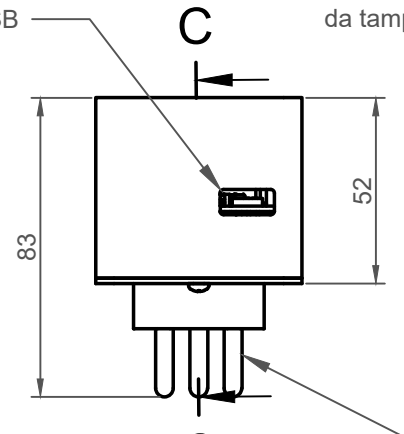


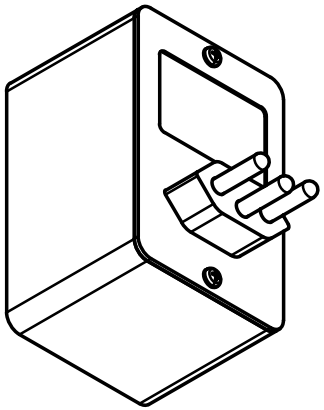


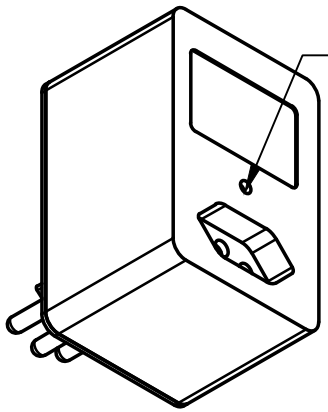
A-A (0.5000)



C-C (0.5000)







conforme NBR 14136

TCC - LUCAS FERNANDES ANDRADE			PROJEÇÃO	NÃO MUDAR A ESCALA DO DESENHO	SE NÃO ESPECIFICADO: DIMENSÕES EM MILÍMETROS
				TOLERÂNCIAS: LINEAR: ˚0,3 ANGULAR:0,5	RETIRAR REBARBAS E QUEBRAR CANTOS VIVOS:
	NOME	DATA	MATERIAL:	ACABAMENTO:	
DES.					
VERIF.					
APROV.			DESCRIÇÃO:		
MANUF.			DESENHO DIMENSIONAL		
QUALID				CÓDIGO	REVISÃO
PESO:			MONITORADOR DE ENERGIA		
				FOLHA: / ESCALA:	DATA 16/11/2018

Referências bibliográficas

[1] AMAZON WEB SERVICES (Org.). **What Is AWS IoT?** Disponível em: <https://docs.amazonaws.cn/en_us/iot/latest/developerguide/is-aws-iot.html>. Acesso em: 05 maio 2018.

[2] AMAZON WEB SERVICES (Org.). **Processo de autenticação do Signature versão 4.** Disponível em: <https://docs.aws.amazon.com/pt_br/general/latest/gr/signat version-4.html>. Acesso em: 06 maio 2018.

[3] LIGHT, Roger A. **Mosquitto: server and client implementation of the MQTT protocol.** Journal of Open Source Software, v. 2, n. 13, 2017.

[4] **Current Transformer Basics: Understanding Ratio, Polarity, and Class** Disponível em: <https://testguy.net/content/190-Current-

Transformer-Basics-Understanding-Ratio-Polarity-and-Class>. Acesso em: 02 set. 2018.

[5] JONES, Michael; BRADLEY, John; SAKIMURA, Nat. **Json web token (jwt).** 2015.

[6] KRAWCZYK, Hugo; BELLARE, Mihir; CANETTI, Ran. HMAC: **Keyed-hashing for message authentication.** 1997.

[7] DANG, Quynh H. **Secure hash standard.** 2015.

[8] MONGODB (Org.). **Introduction to MongoDB.** Disponível em: <https://docs.mongodb.com/manual/introduction/>. Acesso em: 10 jun. 2018.

[9] JOSEFSSON, Simon. **The base16, base32, and base64 data encodings.** 2006.

[10] GOOGLE (Org.). **Design: Create intuitive and beautiful products with Material Design.** Disponível em: <https://material.io/design/>. Acesso em: 13 jun. 2018.

[11] PROVOS, Niels; MAZIERES, David. **A Future-Adaptable Password Scheme.** In: USENIX Annual Technical Conference, FREENIX Track. 1999. p. 81-91.

[12] VOWSTAR. Github. **NodeMCU Devkit.** Disponível em: <https://github.com/nodemcu/nodemcu-devkit>. Acesso em: 19 jul. 2018.

[13] JONSSON, Jakob; KALISKI, Burt. Public-key cryptography standards (PKCS) 1: RSA cryptography specifications version 2.1. 2003.

[14] EMBEDDED LAB.**A BRIEF OVERVIEW OF ALLEGRO ACS712 CURRENT SENSOR** Disponível em: <http://embedded-lab.com/blog/a-brief-overview-of-allegro-acs712-current-sensor-part-1/>. Acesso em: 12 ago. 2018.

[15] HIVEMQ (Org.). **Introducing HiveMQ, the enterprise MQTT broker.** Disponível em: <https://www.hivemq.com/hivemq/>. Acesso em: 10 maio 2018.

[16] AMAZON WEB SERVICES (Org.).**Certificados X.509.** Disponível em: <https://docs.aws.amazon.com/pt_br/iot/latest/developerguid certs.html>. Acesso em: 06 maio 2018.

[17] FIELDING, Roy et al. **Hypertext transfer protocol–HTTP/1.1.** 1999.

[18] FETTE, Ian; MELNIKOV, Alexey. The websocket protocol. 2011.

[19] Standard, O. A. S. I. S. **"MQTT version 3.1. 1."** URL http://docs. oasis-open. org/mqtt/mqtt/v3 1 (2014).

[20] Yokotani, Tetsuya, and Yuya Sasaki. *"Comparison with HTTP and MQTT on required network resources for IoT."* Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016 International Conference on. IEEE, 2016.

[21] Y. Thongkhao and W. Pora, **"A low-cost Wi-Fi smart plug with on-off and Energy Metering functions,"** 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Mai, 2016, pp. 1-5. doi: 10.1109/ECTICon.2016.7561264

[22] K. Mandula, R. Parupalli, C. A. S. Murty, E. Magesh and R. Lunagariya, **"Mobile based home automation using Internet of Things(IoT),"** 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), Kumaracoil, 2015, pp. 340-343. doi: 10.1109/ICCICCT.2015.7475301.

[23] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou and C. H. Lung, **"Smart Home: Integrating Internet of Things with Web Services and Cloud Computing,"** 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, 2013, pp. 317-320. doi: 10.1109/CloudCom.2013.155.

[24] ESP8266 COMMUNITY WIKI Disponível em <https://www.esp8266.com/wiki/doku.php?id=esp8266-module-family> Acesso em: 25 de julho de 2018.

[25] Github.**A data validation library in JavaScript for the browser and Node.js, inspired by Laravel's Validator.** Disponível em: <https://github.com/skaterdav85/validatorjs>. Acesso em: 25 ago. 2018.

[26] Github.**Promise based HTTP client for the browser and node.js.** Disponível em: <https://github.com/axios/axios>. Acesso em: 20 jun. 2018.

[27] JESÚS OTERO GOMEZ. Github.**oast notifications for React.js** Disponível em: <https://github.com/jesusoterogomez/react-notify-toast>. Acesso em: 20 jun. 2018.

[28] Github. **React components that implement Google's Material Design.** Disponível em: <https://github.com/mui-org/material-ui>. Acesso em: 27 jun. 2018.

[29] Github.**Object schema validation** Disponível em: <https://github.com/hapijs/joi>. Acesso em: 08 jul. 2018.

[30] Github.**Optimized bcrypt in plain JavaScript with zero dependencies.** Disponível em: <https://github.com/dcodeIO/bcrypt.js>. Acesso em: 10 jul. 2018.

[31] Github.**DynamoDB data mapper for Node.js.** Disponível em: <https://github.com/baseprime/dynamodb>. Acesso em: 10 jul. 2018.

[32] Auto Core Robótica. Disponível em: <https://www.autocorerobotica.com.br/transformador-de-corrente-ac-hmct103c-5a5ma-nao-invasivo>. Acesso em: 10 set. 2018.