UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE PLUGUE PARA TOMADA ELÉTRICA CONTROLADO REMOTAMENTE

DYEGO ALEKSSANDER MAAS

BLUMENAU
2017

DYEGO ALEKSSANDER MAAS

PROTÓTIPO DE PLUGUE PARA TOMADA ELÉTRICA CONTROLADO REMOTANTE

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Francisco Adell Péricas, Mestre - Orientador

BLUMENAU
2017

PROTÓTIPO DE PLUGUE PARA TOMADA ELÉTRICA CONTROLADO REMOTANTE

Por

DYEGO ALEKSSANDER MAAS

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas, Mestre – Orientador, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof. Marcos Rodrigo Momo, Mestre – FURB

Blumenau, 11 de dezembro de 2017

Dedico este trabalho a minha família e meus amigos, que me apoiaram e suportaram ao longo de todos esses anos. Sua ajuda foi sem dúvidas decisiva para a minha conclusão do curso de graduação.

AGRADECIMENTOS

Agradeço primeiramente à minha família, pelo apoio constante.
Aos meus amigos, que me mantiveram no curso correto.
Ao meu amigo Alexandre, que me ajudou nas horas difíceis.
E ao meu orientador, sempre tão atencioso.

O único caminho para desvendar os limites do possível é aventurar-se além dele, através do impossível.

Arthur C. Clarke

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de plugue para tomada elétrica que pode ser controlado remotamente, permitindo ao usuário ligar e desligar remotamente dispositivos eletroeletrônicos, além de acompanhar relatórios do consumo de energia elétrica dos seus dispositivos. Desta forma, visa permitir ao usuário utilizar sua energia elétrica de forma mais eficiente, reduzindo desperdícios. O protótipo emprega o protocolo Message Queuing Telemetry Transport (MQTT) para realizar a comunicação entre o servidor e um ou mais plugues, e foi desenvolvido com base no módulo ESP8266. O plugue faz uso do padrão arquitetural Event Sourcing e da biblioteca Marten para registrar todos os eventos lógicos do plugue, aproveitando os recursos de armazenamento não-relacional do banco de dados PostgreSQL. Permite, assim, a execução de consultas temporais do estado lógico do plugue em qualquer momento do tempo desde a sua primeira ativação, e a geração de uma linha do tempo consistente que inclui todos os eventos significativos. Para testar o recurso de monitoramento de consumo utilizou-se equipamentos que apresentam três tipos de carga elétrica: capacitiva, resistiva e indutiva. O trabalho atingiu seus objetivos, resultando no desenvolvimento de um protótipo funcional que permite ao usuário controlar múltiplos plugues e ligar ou desligar dispositivos de forma imediata ou agendada, através de um portal Web com design responsivo compatível com dispositivos móveis. Por fim, disponibiliza relatórios de consumo de energia elétrica e uma linha do tempo do plugue.

Palavras-chave: IoT. MQTT. Event sourcing. Plugue elétrico. ESP8266.

ABSTRACT

This work presents the development of a prototype of a prototype of remotely controlled plug for electrical outlets, that allows the user to remotely turn on and off electronic devices, besides following reports on his devices' electrical consumption. In this way, it aims to enable the user to use his electrical power in more efficient ways, also reducing waste. The prototype employs the Message Queuing Telemetry Transport (MQTT) protocol to perform communication between the server and one or more plugs, and its development was based on the ESP8266 module. The plug uses the Event Sourcing architectural pattern and the Marten library to record every logical event of the plug, taking advantage of the non-relational storage resources of the PostgreSQL database. It allows, therefore, the execution of temporal queries on the plug's logical state at any moment in time since the plugs first activation, and the generation of a consistent timeline that includes every significant event. To test the consumption monitoring, three types of electrical loads were used: capacitive, resistive and inductive. The work achieved its objectives, resulting in a functional prototype that allows the user to control multiple plugs and turn on and off devices in both immediate and scheduled ways, through a Web portal with a responsive design compatible with mobile devices. Lastly, it provides reports on electrical consumption and a timeline for the plug.

Key-words: IoT. MQTT. Event sourcing. Electrical plug. ESP8266.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

6LoWPAN – IPV6 over Low-powered Wide Personal Area Network

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface

CoAP – Constrained Application Protocol

CSS – Cascade Style Sheets

DTLS – Datagram Transport Layer Security

EPE – Empresa de Pesquisa Energética

IoT – Internet of Things (Internet das Coisas)

LED – Light-emitting Diode

LwM2M – Lightweight Machine-to-Machine

MQTT – Message Queuing Telemetry Transport

OASIS – Advanced Open Standards for the Information Society

REST – REpresentational State Transfer

RMS – Root Mean Square

SMS – Short Message Service

SSL – Secure Sockets Layer

TLS – Transport Layer Security

URN – Universal Resource Name

SUMÁRIO

1 INTRODUÇÃO

Existem atualmente mais de 40 usinas hidrelétricas em construção ou planejadas para a Bacia do Rio Tapajós, em plena floresta amazônica (GREENPEACE, 2016, p. 12). Segundo o estudo de impacto ambiental realizado pela CNEC Worley Parsons, a maior dentre elas, a usina de São Luiz dos Tapajós, com potencial de geração de 8.040 MW, deve inundar quase 400 km^2 da floresta (CNEC Worley Parsons, 2014, volume 13, parte I, p. 149 apud GREENPEACE, 2016, p. 10), além de outros 2.200 km^2 de desmatamentos indiretos, como resultado da abertura de estradas e de outras obras relacionadas à construção da barragem e do influxo populacional para a região (BARRETO, 2014, apud GREENPEACE, 2016, p. 10).

Por outro lado, segundo a Abesco (2015, p. 1), o Brasil desperdiçou 53 TWh no ano de 2015, valor 5,29% maior do que o registrado em 2009, e equivalente a 60% da produção total da Usina Hidrelétrica de Itaipu. Ainda de acordo com a Abesco (2015, p. 1), "as duas maiores causas do desperdício são a falta de um programa de gestão energética dentro das empresas e o uso de equipamentos defasados na indústria."

Já no setor residencial, que representa 9,6% do consumo nacional de eletricidade, o desperdício está também associado a equipamentos que permanecem ligados em modo de "espera" (EPE, 2016, p. 23).

Grande parte dos equipamentos disponíveis hoje em dia possuem um modo de "espera" para ser utilizado, durante o qual apresentam um consumo de energia denominado *standby* (DANTAS, 2014, p. 19). O autor também alerta que alguns tipos de aparelhos consomem "mais energia por estarem "desligados" sobre o modo em *standby* do que em sua função plena de utilização."

O consumo de energia dos aparelhos que ficam ligados continuamente em *standby* pode representar até 12% do consumo de uma residência (RODRIGUES, 2009, apud ABREU, 2015, p. 1).

> Nos últimos anos, o avanço incessante das TIC (Tecnologias de Informação e Comunicação) e redes de sensores, novas aplicações para melhorar a eficiência energética emergem constantemente. Por exemplo, em espaços de escritórios, temporizadores e sensores de movimento provém uma ferramenta útil para detectar e responder aos ocupantes ao mesmo tempo em que disponibilizam feedback para encorajar mudanças de comportamento (MORENO *et al.*, 2014, p. 9585, tradução nossa).

Diante deste cenário, onde o desperdício de eletricidade é evidente, propõe-se implementar um protótipo de tomada elétrica, cujo fornecimento de energia elétrica possa ser controlado remotamente, monitorando o consumo dos equipamentos conectados a ela, permitindo assim um uso mais consciente dos recursos energéticos.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um plugue para tomada elétrica que possa, através do emprego de sensores e atuadores, ser controlado remotamente e que permita o monitoramento do consumo elétrico dos equipamentos conectados.

Os objetivos específicos são:

a) desenvolver um protótipo de plugue para tomada elétrica controlado remotamente;
b) permitir ativar e desativar, via aplicativo móvel, o fornecimento de energia elétrica para os aparelhos conectados ao plugue;
c) disponibilizar em um aplicativo móvel relatórios do consumo de energia elétrica dos aparelhos conectados ao plugue.

1.2 ESTRUTURA

A presente monografia divide-se em quatro capítulos: introdução, fundamentação teórica, desenvolvimento do protótipo de tomada inteligente e conclusões. O capítulo 2 visa fornecer um embasamento teórico a respeito dos principais assuntos abordados neste trabalho, elucidando conceitos como a Internet das Coisas, além dos principais protocolos de comunicação utilizados nesse meio. Elabora também sobre o módulo ESP8266 e a placa de prototipagem NodeMCU, além do padrão arquitetural Event Sourcing. Para finalizar, apresenta três trabalhos correlatos a este. Em seguida, o capítulo 3 expõe os principais pontos do desenvolvimento do protótipo como: os requisitos, a arquitetura e o desenvolvimento do protótipo. Além disso, também são apresentadas as ferramentas utilizadas, a implementação e operacionalidade da aplicação, bem como são analisados os resultados obtidos. O capítulo 4 encerra a monografia expondo as conclusões obtidas neste trabalho e sugere extensões que podem ser desenvolvidas a partir deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo descreve brevemente os assuntos que fundamentarão o estudo a ser realizado: Internet das coisas, protocolos de rede de baixo consumo de banda e microcontroladores de baixo consumo energético. Assim, o capítulo está subdividido em quatro partes. A seção 2.1 descreve o conceito da Internet das Coisas, elaborando sobre seu propósito e arquitetura. Já a seção 2.2 apresenta o módulo ESP8266 e placa de prototipagem NodeMCU. A seção 2.3 elabora sobre os principais protocolos de comunicação utilizados em redes com dispositivos restritos. A seguir, a seção 2.4 explana sobre o padrão arquitetural Event Sourcing. Por fim, a seção 2.5 descreve três trabalhos correlatos.

2.1 INTERNET DAS COISAS

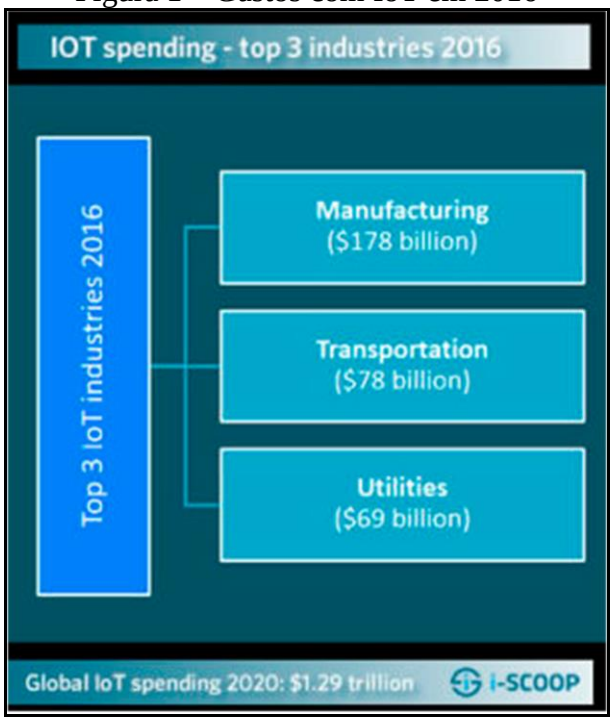
O termo Internet das Coisas (em inglês Internet of Things – IoT) foi cunhado pelo MIT Auto-ID Labs em 1999, ocasião em que trabalhava no campo da identificação, usando sensores sem fio para localizar e reconhecer o estado de objetos, e tecnologias de identificação por radiofrequência (ZHU *et al.*, 2010).

O conceito da IoT é considerado a terceira onda da tecnologia da informação, sucedendo a Internet e a rede de comunicação móvel (ZHU *et al.*, 2010). A IoT é composta basicamente por três componentes: os nós de borda (as coisas), nós *gateway* e servidores ou *datacenters* (LEVY; WALLIS, 2014).

Microcontroladores de ultra-baixa-potência são os motores computacionais que controlam os nós de borda da IoT (LEVY; WALLIS, 2014). Dependendo da aplicação, os nós podem coletar dados, enviar dados, ou ambos, e devem fazê-los com o menor consumo possível, seja de banda de dados ou consumo energético (LEVY; WALLIS, 2014).

A quantidade de dispositivos conectados cresce a cada ano, e as projeções para a quantidade de dispositivos conectados no ano 2020 variam de 20 a 30 bilhões (I-SCOOP, 2016). A Figura 1 apresenta os gastos com IoT no ano de 2016 pelas indústrias de manufatura, transporte e utilidades.

Figura 1 – Gastos com IoT em 2016



Fonte: i-Scoop (2016).

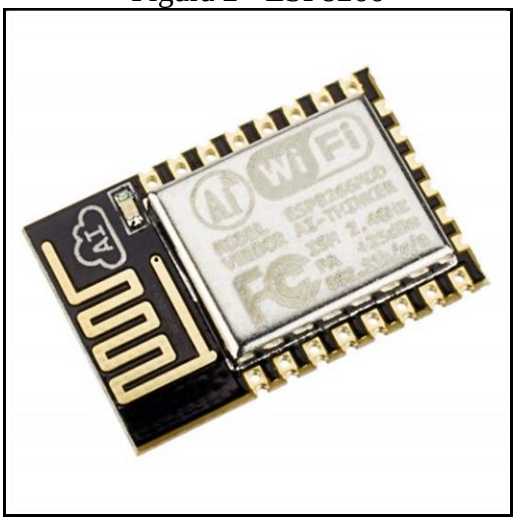
O crescimento projetado para a quantidade de dispositivos conectados até 2020 implica também no crescimento do tráfego de dados na Internet sobre o protocolo IP, em grande parte fomentado pelo aumento nas comunicações máquina-máquina, que devem atingir globalmente a marca de 194,4 *exabytes* mensais, 11,6 *exabytes* somente na América Latina (CISCO, 2016).

As aplicações de tecnologias da IoT na área da saúde, por exemplo, podem trazer grandes benefícios para o público. Algumas das aplicações de dispositivos conectados na saúde incluem rastreadores de atividade, inaladores conectados, sensores ingeríveis e lentes de contato conectadas (DAVIS, 2017).

2.2 ESP-8266

Segundo Kolban (2016, p. 27), desde 2014 o módulo ESP8266 tem sido um dos mais utilizados para prototipagem de soluções relacionadas a IoT, tendo sido entregues dezenas de milhões de unidades. Este módulo tem como principal vantagem o suporte nativo a Wi-Fi, sendo muitas vezes utilizado como módulo Wi-Fi para outros módulos. A Figura 2 apresenta um módulo ESP8266.

Figura 2 - ESP8266



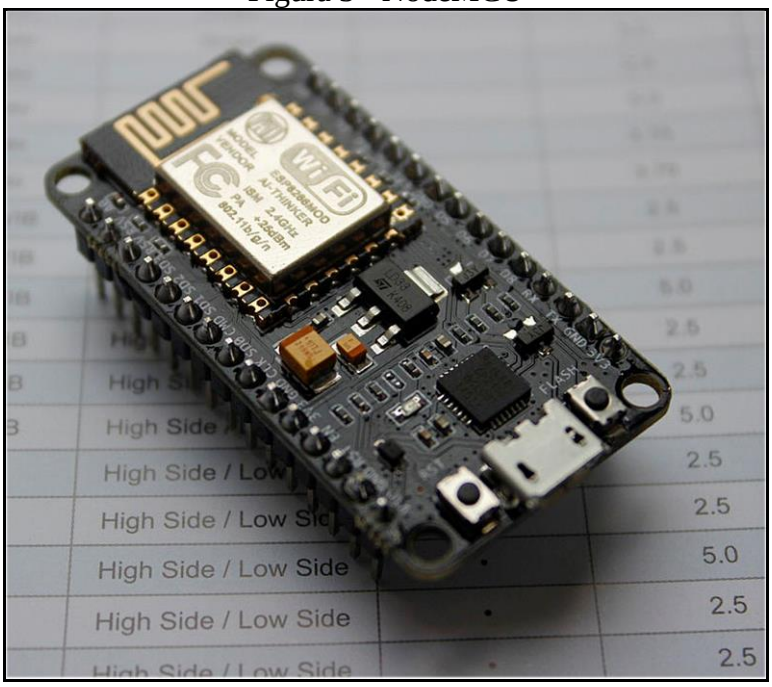
Fonte: Addicore (2017).

Como o ESP8266 apresenta suporte nativo a Wi-Fi e ao protocolo TCP/IP, representa uma boa alternativa ao Arduino Uno. Outra característica que conta a seu favor é o baixíssimo custo. O módulo pode ser encontrado por apenas US$ 2,00 e uma placa de desenvolvimento a partir de US$ 3,00, uma fração do preço de uma placa de prototipagem Arduino Uno (KOLBAN, 2016, p. 26).

Existem vários modelos de placas no mercado, como a ESP-1 ou ESP-12, mas o processador ESP8266 é o mesmo em todas as placas. O elemento que muda é a quantidade de pinos General Purpose Input/Output (GPIO) que são expostos, além do tipo de pinos para conexão e a quantidade de memória disponível (KOLBAN, 2016, p. 26).

A Figura 3 apresenta a placa de desenvolvimento de código aberto NodeMCU, baseada no módulo ESP8266. A placa possui uma entrada Micro-USB para alimentação e download de *firmware*, além de pinos para prototipagem.

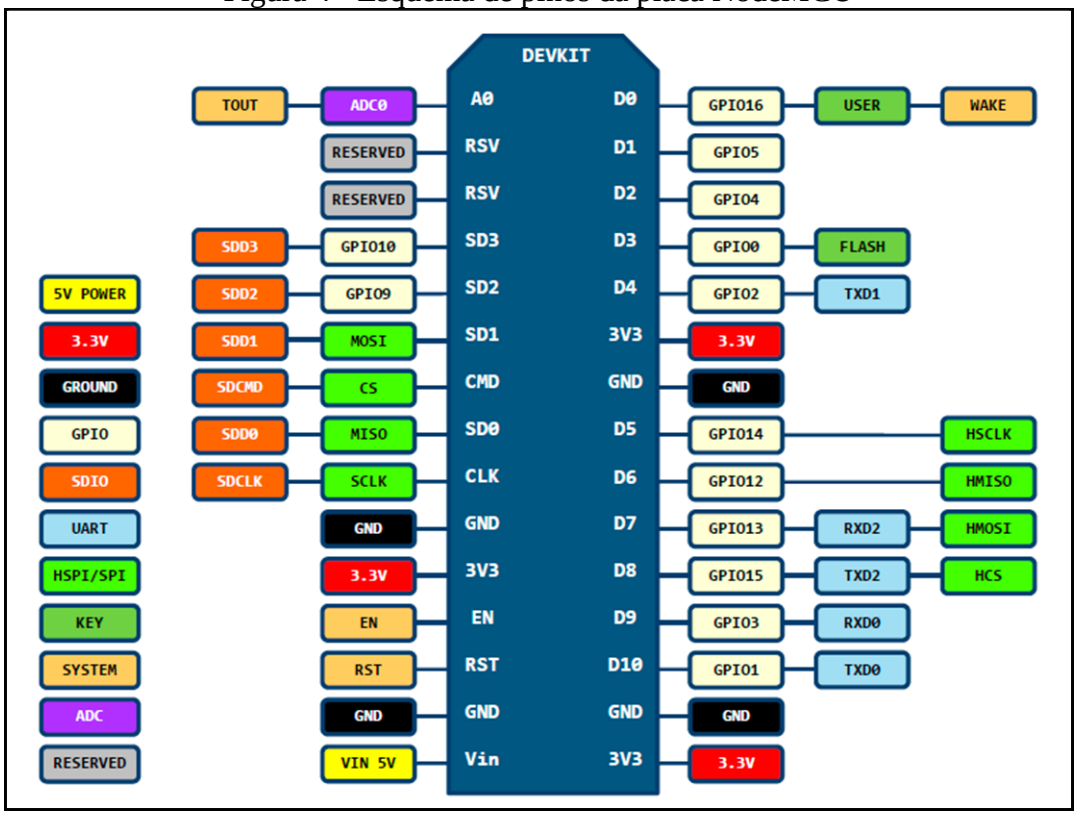
Figura 3 - NodeMCU



Fonte: NodeMCU (2017).

A Figura 4 mostra um detalhamento do esquema dos pinos da placa de desenvolvimento. Nela, pode-se ver que a placa disponibiliza 13 pinos digitais do tipo General Purpose Input/Output (GPIO), que podem ser utilizadas para controlar outros componentes eletrônicos. Uma diferença do ESP8266 em relação ao Arduino Uno é a tensão de funcionamento da placa. Enquanto o Arduino funciona com uma tensão elétrica de 5V, o ESP8266 utiliza uma tensão de 3,3V.

Figura 4 - Esquema de pinos da placa NodeMCU



Fonte: NodeMCU (2017).

2.3 PROTOCOLOS DE COMUNICAÇÃO

Nos últimos anos, surgiram vários protocolos de comunicação voltados a IoT. Os dois protocolos mais promissores são o Message Queue Telemetry Transport (MQTT) e o Constrained Application Protocol (CoAP) (JAFFEY, 2014).

Além desses dois, o protocolo Lightweight Machine-to-Machine (LwM2M) surgiu para suprir a necessidade de um protocolo focado no gerenciamento de dispositivos na IoT (AVSYSTEM, 2017).

2.3.1 Constrained Application Protocol

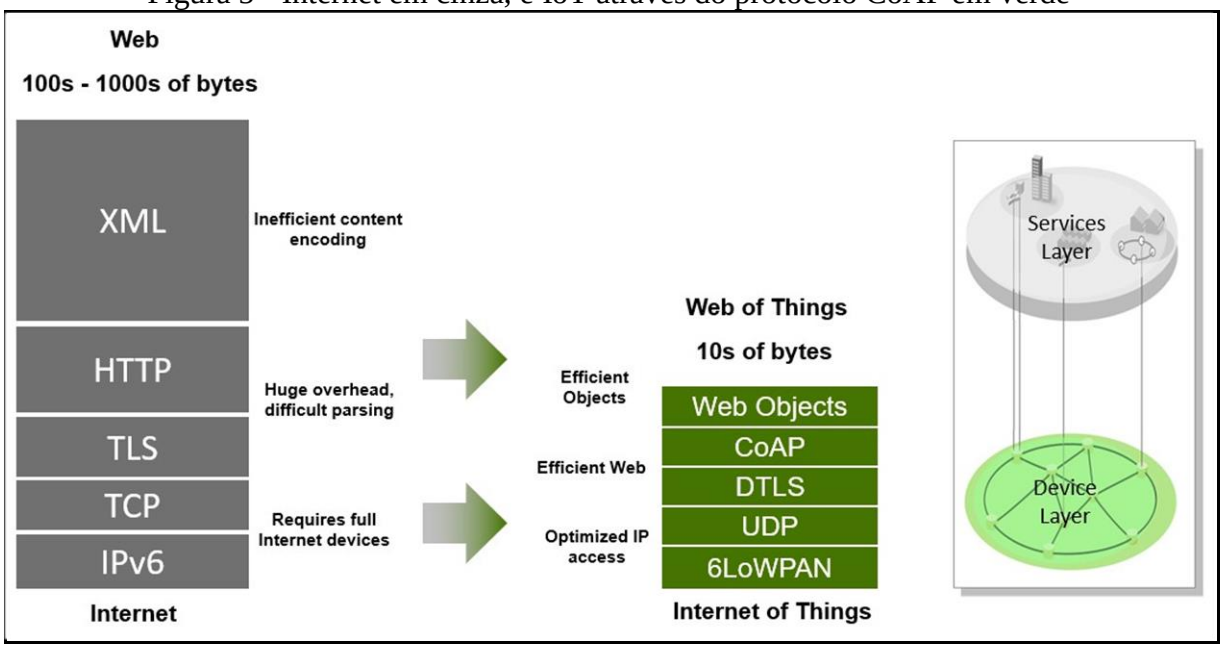
O CoAP é um protocolo de transferência de documentos que, ao contrário do HTTP, é desenhado para atender às necessidades de dispositivos restritos (JAFFEY, 2014). O protocolo é especializado em atuar em nós e redes sujeitas a restrições no âmbito da IoT, sendo desenhado para desempenhar comunicação máquina-máquina em aplicações como automação residencial e energia inteligente (COAP, 2014).

De acordo com Jaffey (2014, p. 1), para reduzir o consumo de banda, “mapas de bits e mapeamentos de *strings* para inteiros são usados extensivamente para salvar espaço”. Essa estratégia garante que a análise e geração dos pacotes sejam simples e exijam poucos recursos em sistemas restritos.

O protocolo segue um modelo cliente/servidor e oferece as operações GET, PUT, POST e DELETE entre cliente e servidor, e uma das suas principais características é a interoperabilidade com o protocolo HTTP, através do uso de *proxies* responsáveis por adaptar as requisições entre um formato e o outro. Ao contrário do protocolo HTTP, que opera em cima do protocolo TCP, o CoAP opera com o protocolo UDP. Isso permite que cliente e servidor se comuniquem através de datagramas que não dependem de conexão. Assim, o protocolo não garante a entrega das mensagens, e tratamentos de erro e estratégias de tolerância são de responsabilidade da aplicação (JAFFEY, 2014). Ainda segundo Jaffey (2014, p. 1), em termos de segurança, o protocolo UDP permite o mesmo nível de segurança que o HTTPS através do emprego da Datagram Transport Layer Security (DTLS).

A Figura 5 apresenta uma comparação entre as pilhas de protocolos empregadas na Web e aquela empregada em IoT através do protocolo CoAP. Nela podemos observar que a escolha de uma pilha de protocolos mais enxuta permite ao protocolo CoAP operar com um consumo de banda de dezenas a centenas de vezes menor do que o protocolo HTTP.

Figura 5 - Internet em cinza, e IoT através do protocolo CoAP em verde



Fonte: Shelby (2013).

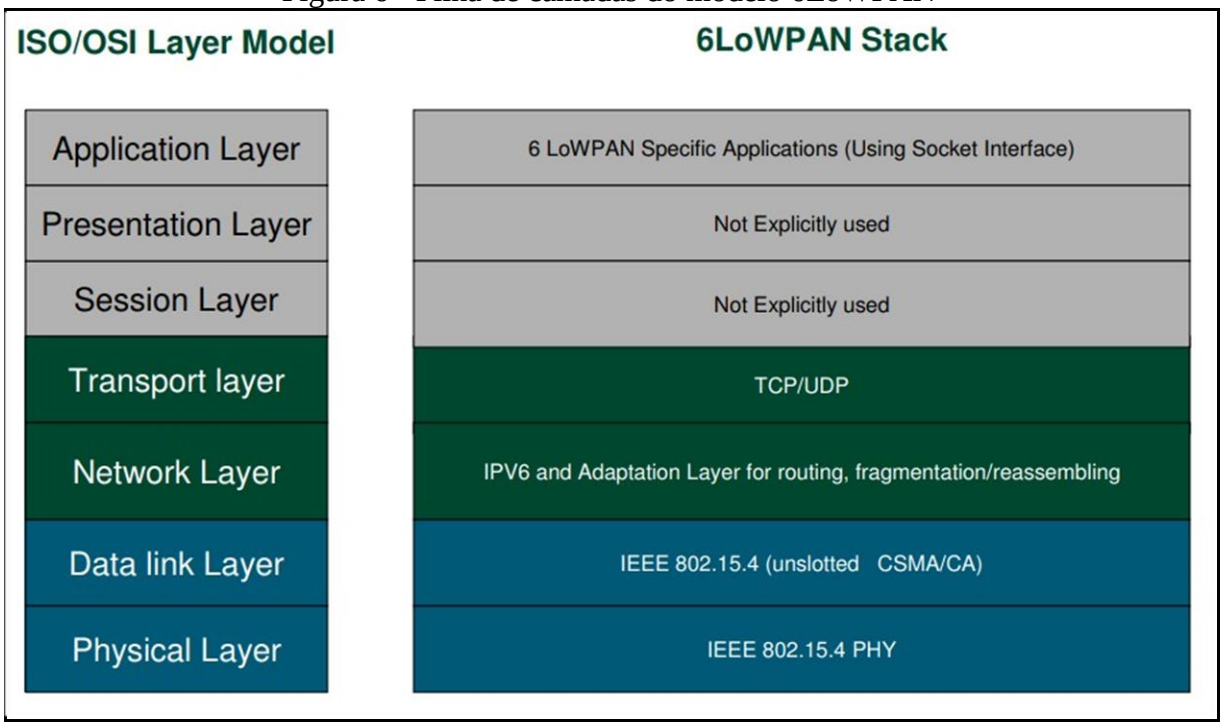
No protocolo CoAP um cliente pode observar um recurso do servidor. Para isso, ele pode realizar uma operação GET com o indicador “observe” ativo. Desta forma, o servidor manterá uma conexão ativa com este cliente, informando o cliente de cada mudança de estado daquele recurso na medida em que elas acontecem (JAFFEY, 2014).

Além das redes Wi-Fi, o transporte dos pacotes pode acontecer em redes 6LoWPAN, como pode ser visto na Figura 5. A 6LoWPAN simplifica o IPV6 ao adotar formatos de

cabeçalho compactos, permitindo seu uso por dispositivos restritos (SHELBY; BORMANN, 2011).

De acordo com a Texas Instruments (2017, p. 1), a "6LoWPAN é uma rede em malha sem fio de baixo consumo onde cada nó possui seu próprio endereço IPv6, permitindo que ele se conecte diretamente à Internet utilizando padrões abertos." Na Figura 6, pode-se ver como a pilha de camadas do modelo 6LoWPAN se compara ao modelo OSI.

Figura 6 - Pilha de camadas do modelo 6LoWPAN



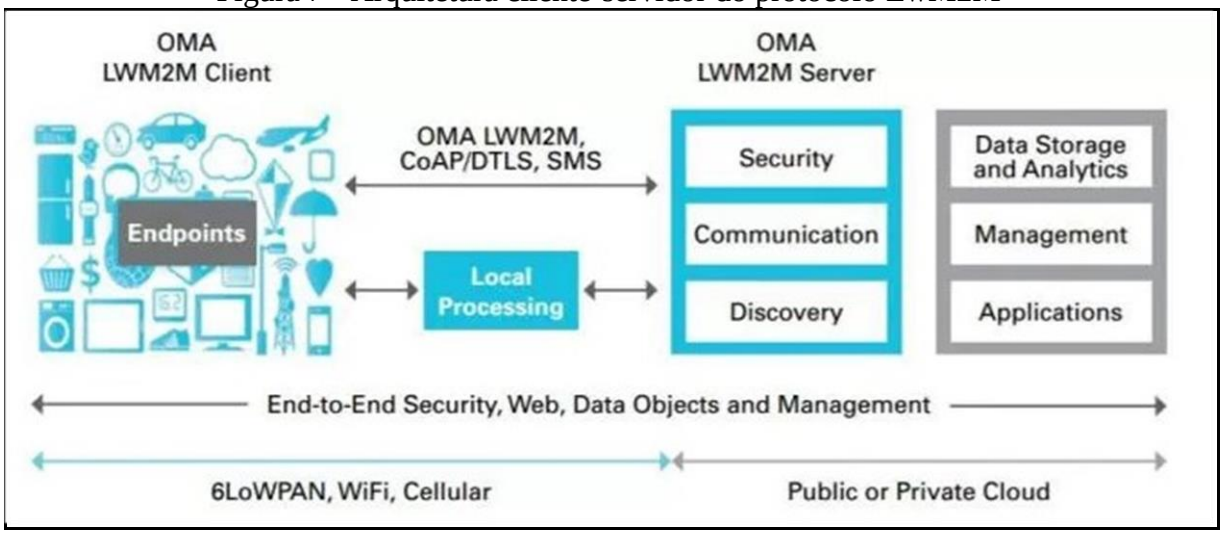
Fonte: Mindtek (2009, p. 5).

2.3.2 Lightweight M2M

A Open Mobile Alliance (OMA) desenvolveu o protocolo LwM2M com o intuito de permitir o gerenciamento de dispositivos na IoT, além de suportar outras aplicações Máquina-Máquina, como redes celulares (AVSYSTEM, 2017). Ainda segundo a AVSystem (2017, p. 1), o protocolo foi "[…] desenhado para ser transportado tanto através de UDP assegurado por DTLS em redes IP, como através de SMS diretamente em redes de telefones celulares."

Os primeiros desenvolvimentos na área da IoT careceram de padronização, resultando em múltiplas soluções incompatíveis entre si e em um ecossistema fragmentado. Segundo Fuller (2016, p. 1), "[...] a fragmentação é uma grande barreira à ubiquidade, talvez até maior do que a segurança." Ao disponibilizar interfaces padronizadas para desacoplar diferentes componentes de sistema, o LwM2M visa resolver problemas de fragmentação técnica com um mecanismo capaz de atender as necessidades de dispositivos M2M (TRACY, 2016). A Figura 7 apresenta a arquitetura cliente-servidor empregada no protocolo.

Figura 7 - Arquitetura cliente-servidor do protocolo LwM2M



Fonte: Open Mobile Alliance (2016, p. 1).

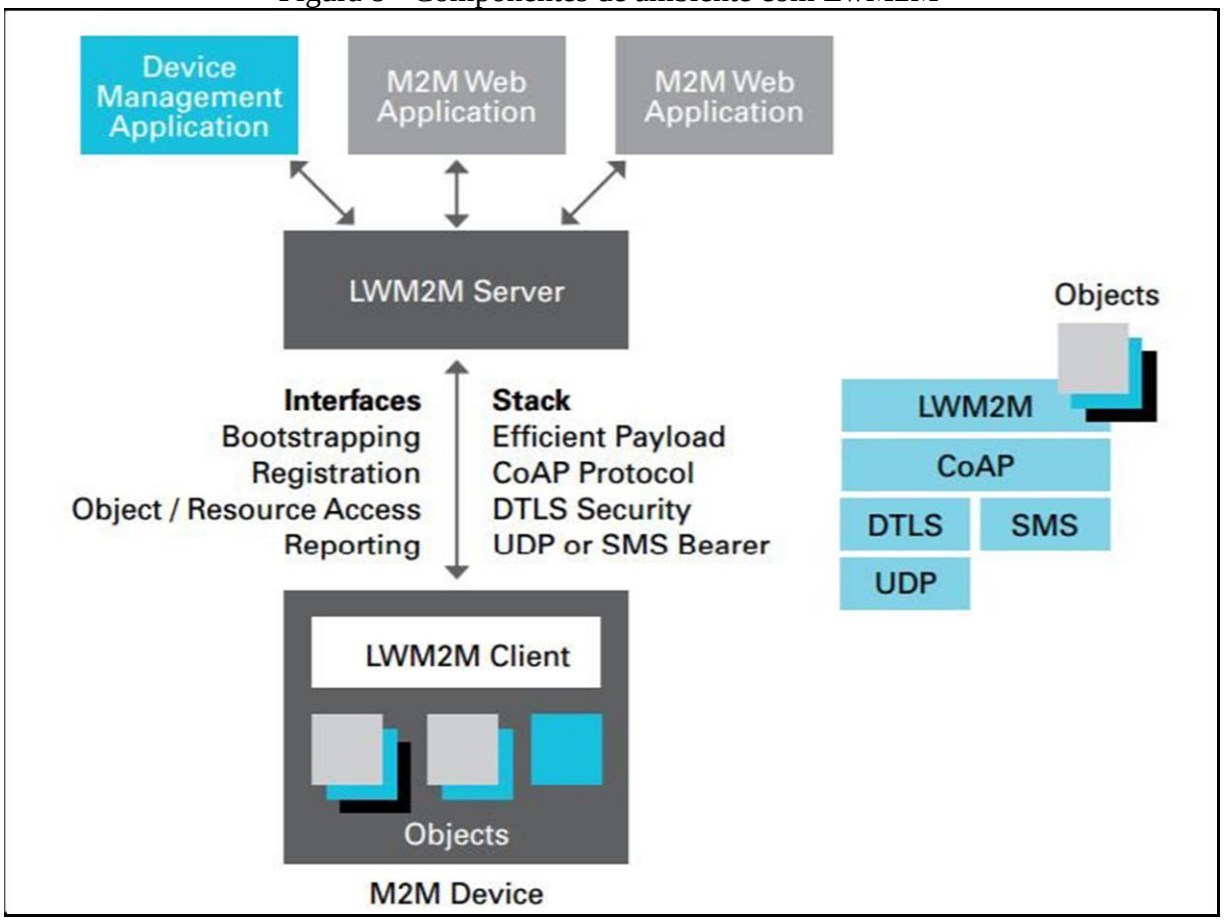
Ambientes de rede envolvendo ao protocolo LwM2M consistem em três tipos de entidades: clientes, servidores e servidores de *bootstrap*. Os clientes ficam localizados nos dispositivos e se comunicam com o servidor, de modo que podem ser gerenciados e ter seus recursos monitorados. Esses recursos são expostos através de um modelo de dados padronizado, definido no cliente. Cada cliente LwM2M possui identificação única, que independe do endereço na rede, através de um Universal Resource Name (URN) definida pelo fabricante do dispositivo. Esse URN é chamado Endpoint Client Name (AVSYSTEM, 2017).

Os servidores de *bootstrap* têm o propósito de inicializar o modelo de dados, incluindo conexões com os servidores LwM2M. Eles atuam com um conjunto diferente de comandos, e podem ser contatados pelos clientes durante a primeira inicialização, ou toda vez que o dispositivo for reiniciado (AVSYSTEM, 2017).

Por último, os servidores LwM2M mantêm as conexões com os clientes LwM2M e podem ler e escrever no modelo de dados exposto pelos clientes. Cada cliente pode estar conectado a múltiplos servidores, e cada servidor pode acessar uma parte diferente do modelo de dados do cliente (AVSYSTEM, 2017).

A Figura 8 apresenta um exemplo da disposição de todas as partes que envolvem um ambiente LwM2M, considerando os três tipos de entidades apresentados.

Figura 8 - Componentes de ambiente com LwM2M



Fonte: Open Mobile Alliance (2016, p. 1).

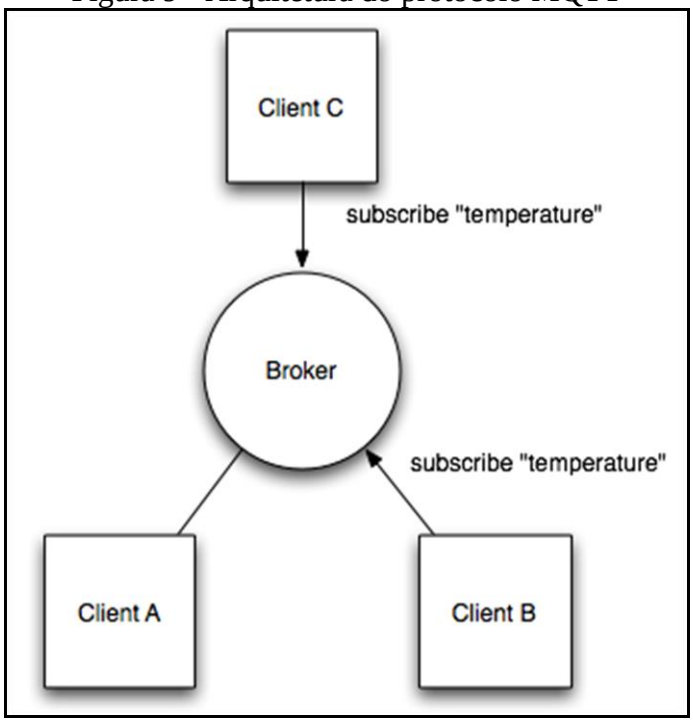
2.3.3 Message Queuing Telemetry Transport

O MQTT é um protocolo de comunicação máquina-máquina desenhado para a Internet das Coisas, e desde a sua criação em 1999, vem sendo aplicado em cenários que vão de automação residencial e aplicações móveis a comunicar leituras de sensores via *link* de satélite em cenários com conexão discada muito limitada (MQTT, 2017). O protocolo foi inicialmente desenvolvido pela IBM em 1999 e hoje é um padrão reconhecido pela Advanced Open Standards for the Information Society (OASIS, 2017).

Conforme Jaffey (2014, p. 1), esse protocolo opera no modelo cliente/servidor onde cada sensor é um cliente que se conecta a um servidor, conhecido também como *broker*, através de uma conexão TCP. O protocolo MQTT é orientado a mensagens, que possuem um endereço, conhecido como tópico. Cada cliente pode se inscrever em múltiplos tópicos, e receberá cada mensagem publicada para os tópicos inscritos. O *broker* é responsável por entregar as mensagens aos clientes interessados.

A Figura 9 exemplifica um cenário com três clientes, onde os clientes B e C se inscrevem no tópico “temperatura”. Quando o cliente A publicar uma mensagem para o tópico “temperatura” com o valor “22”, os clientes B e C recebem essa mensagem.

Figura 9 - Arquitetura do protocolo MQTT



Fonte: Jaffey (2014).

O protocolo oferece também recursos de segurança, permitindo, dessa forma, realizar conexões TCP com o *broker* encriptadas com os protocolos criptográficos Secure Sockets Layer (SSL) ou Transport Layer Security (TLS), caso em que a conexão exige informar usuário e senha (JAFFEY, 2014).

2.4 EVENT SOURCING

Segundo Fowler (2005, p. 1), o padrão arquitetural *Event Sourcing* visa garantir que todos as mudanças no estado lógico da aplicação são persistidas na forma de uma sequência de eventos, possibilitando assim, reconstruir estados lógicos da aplicação no passado.

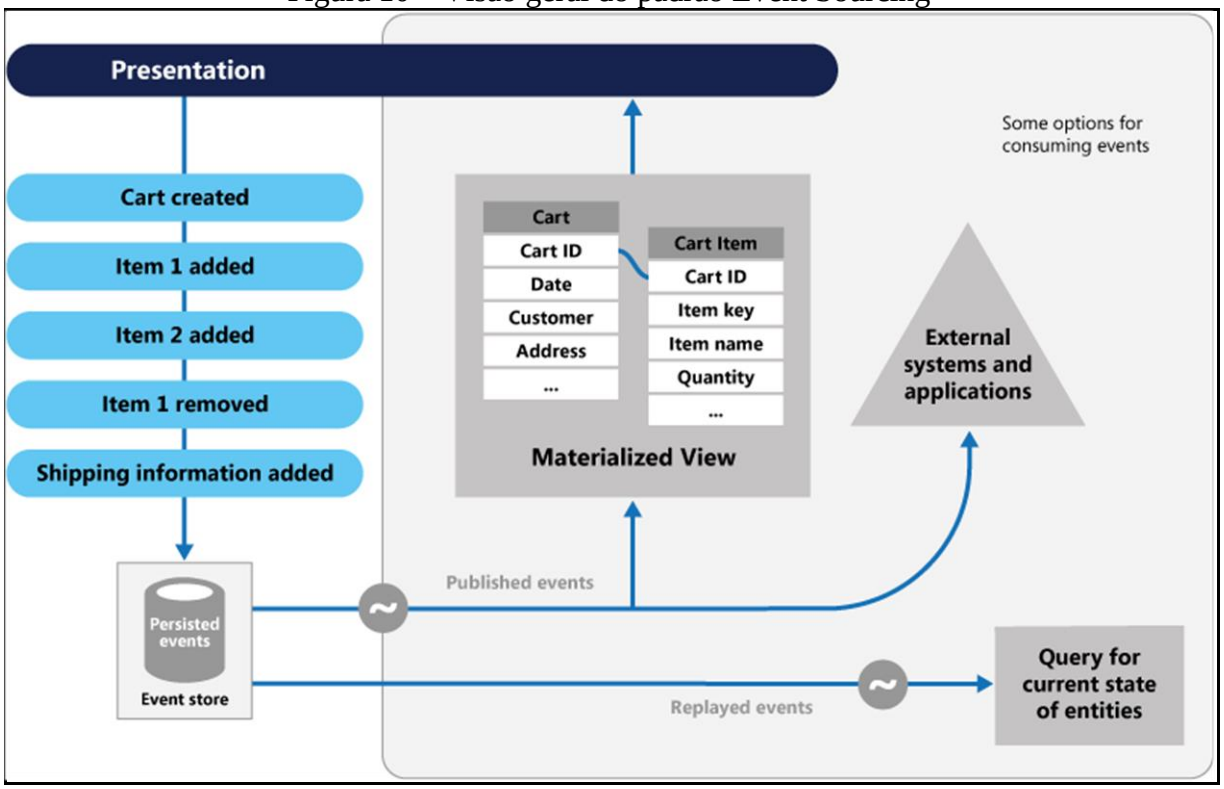
Ao invés de armazenar apenas o estado atual dos dados, armazena-se a série completa de eventos realizados sobre esses dados. Os eventos são persistidos em uma *event store* através de operações *append-only*. Dessa forma, a *event store* atua como um sistema de registro e pode ser usada para materializar objetos de domínio a partir dos eventos salvos (MICROSOFT, 2017).

Essa abordagem resolve alguns dos problemas que emergem a partir do uso de abordagens mais tradicionais, como o *create, read, update and delete* (CRUD). Alguns dos exemplos de problemas resolvidos pelo Event Sourcing são deficiências de performance

advindas de conflitos ao atualizar os dados, sobrecarga causada pela necessidade de carregar os dados para então atualizá-los, e manter mecanismos separados para auditar alterações sobre os dados (MICROSOFT, 2017).

A Figura 10 apresenta o padrão *Event Sourcing*. Os eventos são persistidos na *event store* na sequência em que aconteceram. Pode-se ver que é possível consultar o estado atual de entidades ao executar o *replay* os eventos associados a uma entidade. A Figura 10 mostra também que é possível publicar os eventos persistidos, de modo que outras partes da aplicação, ou até mesmo outras aplicações interessadas se inscrevam nesses eventos, de modo que possam criar suas próprias visões sobre os dados ou executar outras regras de negócio.

Figura 10 – Visão geral do padrão Event Sourcing



Fonte: Microsoft (2017).

De acordo com Fowler (2005, p. 1), o fato de todas as mudanças no estado da aplicação surgirem a partir de eventos, habilita as seguintes possibilidades: a) reconstrução completa do estado da aplicação, descartando o estado atual e reconstruindo-o a partir dos eventos; b) consultas temporais, ao começar do início e executar o *replay* dos eventos até um ponto específico do tempo, reconstruindo o estado da aplicação naquele tempo exato; c) o *replay* de eventos permite a correção de um estado inválido da aplicação em consequência de um evento incorreto. Ao adicionar um evento de reversão, corrigindo o evento original, e executar o *replay* do fluxo de eventos, é possível computar as consequências dessa alteração.

Uma consideração importante acerca do padrão, é que o sistema será apenas eventualmente consistente ao criar visões materializadas ou gerar projeções a partir do *replay* dos eventos. Além disso, como a *event store* é uma fonte permante de informação, os dados dos eventos nunca devem ser atualizados (MICROSOFT, 2017).

2.5 TRABALHOS CORRELATOS

São apresentados trabalhos correlatos que se assemelham em suas características à proposta deste trabalho. A seção 2.1 descreve o Sonoff Pow (ITEAD, 2015), um interruptor Wi-Fi com suporte a medição de consumo de energia. A seção 2.2 apresenta o TP-Link HS110 (TP-LINK, 2015), um plugue que permite controlar remotamente o fornecimento de energia da tomada. Por fim, a seção 2.3 apresenta um protótipo de tomada elétrica controlada remotamente e que faz uso dos conceitos da IoT (WAKA, 2015).

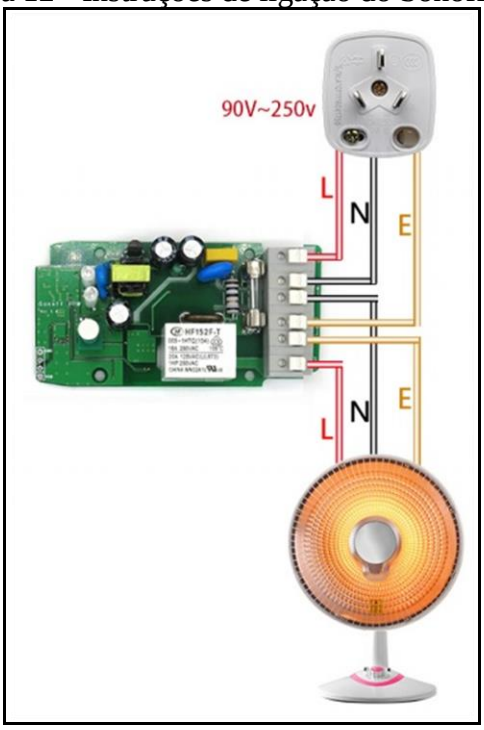
2.5.1 Sonoff Pow

O Sonoff Pow é um interruptor Wi-Fi que permite controlar o fornecimento de energia elétrica através de um aplicativo móvel, além de monitorar o consumo de energia pelos equipamentos conectados a ele (ITEAD, 2015). A Figura 11 mostra um interruptor Sonoff Pow com a tampa de segurança removida, e a Figura 12 apresenta o esquema de ligação do mesmo.

Figura 11 - Sonoff Pow

Fonte: Itead (2015).

Figura 12 - Instruções de ligação do Sonoff Pow



Fonte: Itead (2015).

O interruptor requer que os fios de alimentação do aparelho, ou de uma extensão elétrica, sejam parafusados nos soquetes de entrada do interruptor. Uma vez instalado e ligado à rede de energia, é necessário conceder acesso a uma rede Wi-Fi, o que pode ser feito através de um smartphone. O usuário precisa acessar, através do smartphone, a rede Wi-Fi disponibilizada pelo interruptor, e conceder as credencias de acesso à rede doméstica. Uma vez realizados esses primeiros passos, o interruptor pode ser controlado através do aplicativo móvel eWeLink através de uma rede Wi-Fi, 2G, 3G ou 4G. A Figura 13 apresenta três telas do aplicativo eWeLink: a tela de acompanhamento de consumo de energia em tempo real, a tela de histórico de consumo de energia, e uma tela de medição de consumo com temporizador e cronômetro.

Figura 13 - Aplicativo eWeLink

Fonte: Itead (2015).

No aplicativo, o usuário pode então acompanhar o consumo histórico de energia do equipamento conectado ao interruptor (B), assim como acompanhar o consumo em tempo real. Também é possível ligar e desligar o interruptor (A), além de agendar horários de funcionamento (C).

O interruptor oferece suporte para tensões entre 110 e 250V AC, permitindo que seja utilizado na maioria dos mercados globais. Além disso, suporta correntes de até 16A e potência máxima de 3500W (ITEAD, 2015), suficiente para a grande maioria dos aparelhos eletroeletrônicos no mercado (ENERGIA SUSTENTÁVEL, 2015).

2.5.2 TP-Link HS110

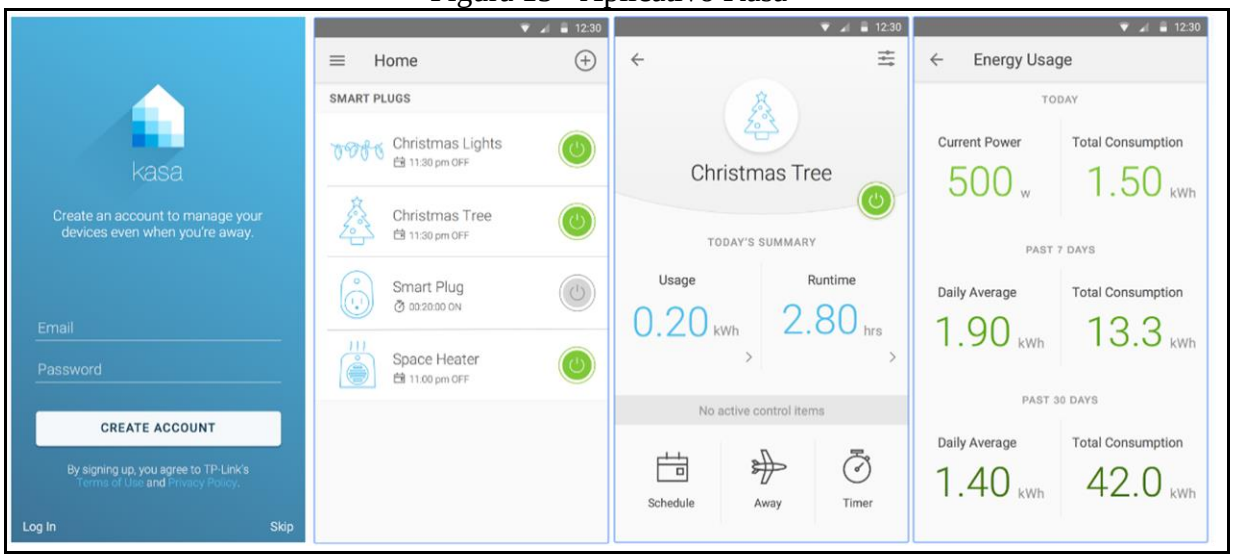
O plugue TP-Link HS110 permite monitorar o consumo de energia elétrica e controlar remotamente o fornecimento de energia elétrica (TP-LINK, 2015). A conectividade ocorre através de Wi-Fi, o que implica que o dispositivo permanece conectado na rede doméstica, acessando o componente Web nos servidores da TP-Link, e permitindo que sejam controlados através da Internet. O plugue possui um diodo emissor de luz (em inglês light-emitting diode – LED) indicador de status e um botão para auxiliar no processo de pareamento com o smartphone do usuário. A Figura 14 apresenta um plugue TP-Link HS110, no padrão estadunidense.

Figura 14 - Plugue TP-Link HS110

Fonte: TP-Link (2015).

O aplicativo Kasa permite configurar e gerenciar os plugues. Para configurar o TP-Link HS110, o usuário deve conectar seu smartphone na rede Wi-Fi do plugue para dar-lhe acesso à rede doméstica e à Internet. O usuário também precisa criar uma conta no serviço Kasa, através do aplicativo. A partir daí o usuário pode ativar e desativar o fornecimento de energia pelo plugue através do aplicativo, agendar horários de funcionamento e definir temporizadores. O aplicativo permite também a criação de cenas. Cada cena agrupa um conjunto de plugues, e outros dispositivos, tais como lâmpadas e interruptores inteligentes. As cenas podem ser ativadas ou desativadas pelo usuário. Quando uma cena está ativa, cada dispositivo assume a configuração definida para o mesmo naquela cena. A Figura 15 apresenta a tela de criação de conta de usuário no serviço Kasa, a tela de controle dos plugues configurados, a tela de acompanhamento e configuração dos plugues e a tela de acompanhamento de consumo de energia.

Figura 15 - Aplicativo Kasa

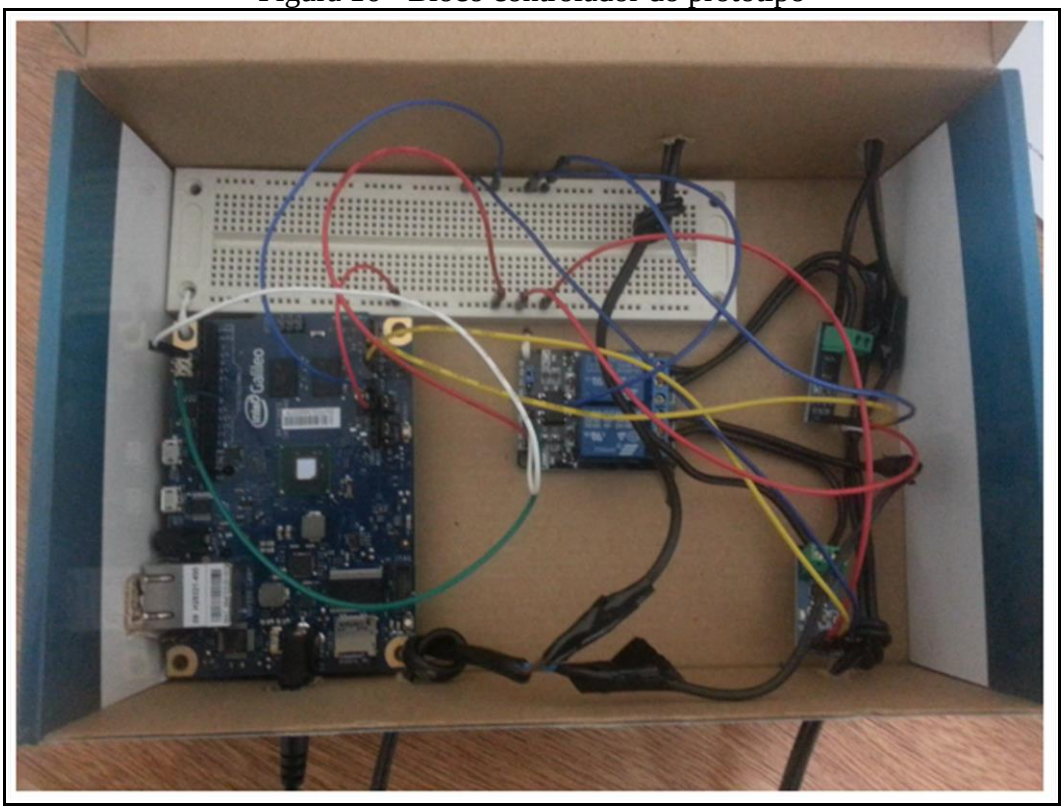


Fonte: Google Play (2017).

2.5.3 Protótipo de controle remoto de tomadas elétricas

O protótipo do sistema de controle remoto de tomadas implementado por Waka (2015) empregou a plataforma de prototipagem Intel Galileo (INTEL GALILEO BOARD, 2015), usando também algumas tecnologias de software livre, como o *framework* de automação residencial openHAB e o *middleware* de comunicação Mosquitto. A Figura 16 apresenta o bloco controlador do protótipo.

Figura 16 - Bloco controlador do protótipo



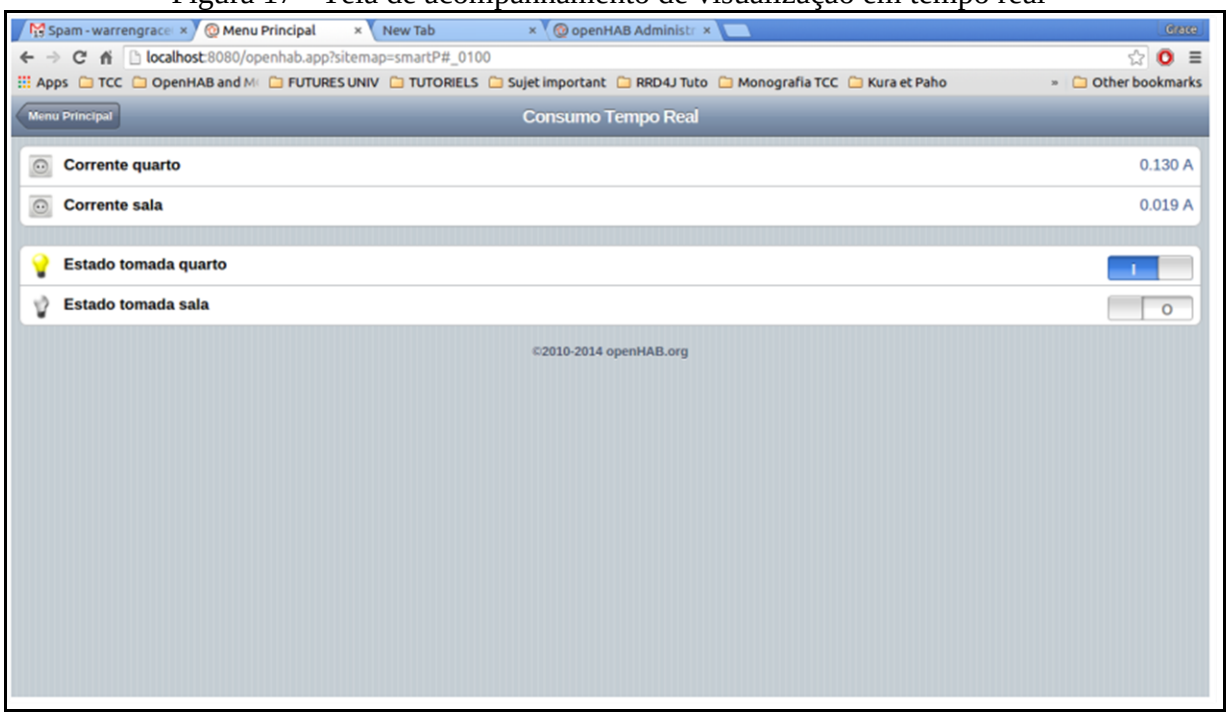
Fonte: Waka (2015).

O protótipo utiliza conectividade Wi-Fi e permite o controle remoto do fornecimento de energia elétrica através de um portal Web da plataforma openHAB. É acessível por um navegador de Internet. Waka (2015) afirma que o protótipo foi testado apenas dentro de uma mesma rede Wi-Fi.

O sistema realiza o monitoramento do consumo de energia elétrica pelos aparelhos conectados à tomada. Uma particularidade do protótipo é que o monitoramento do consumo se restringe à medição da corrente, apresentado ao usuário final na unidade de medida Ampère, com oscilação variando do positivo ao negativo por conta da natureza alternada da corrente.

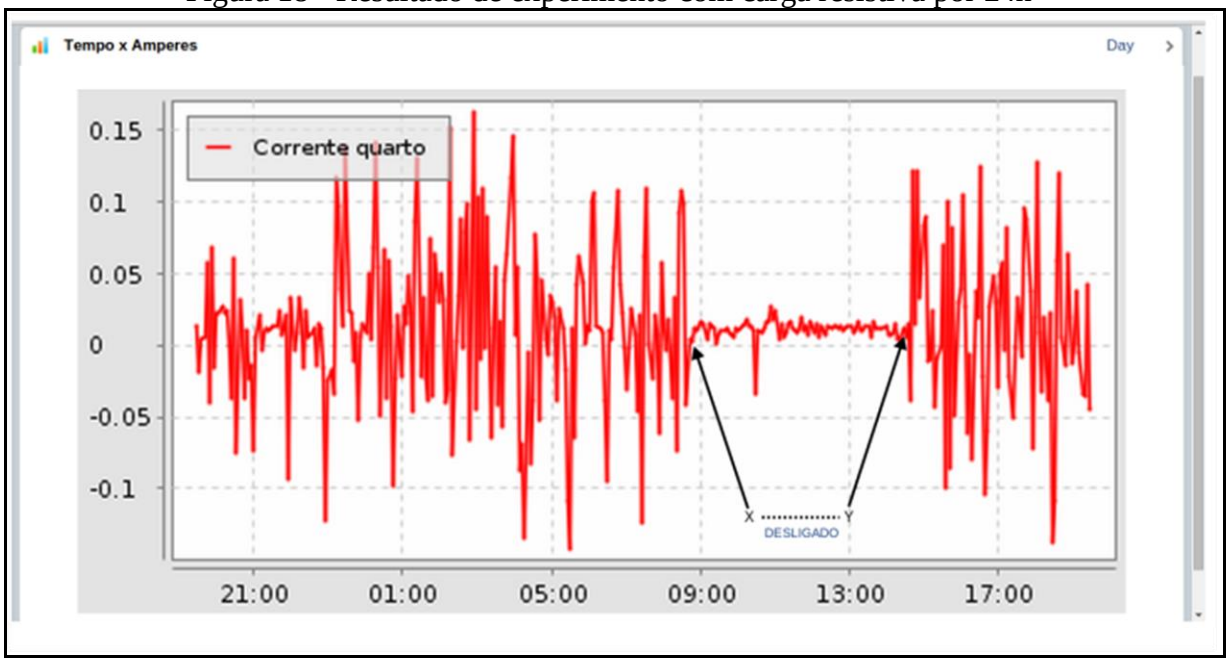
No portal Web do openHAB, é possível também controlar as tomadas, ativando e desativando o fornecimento de energia elétrica para os equipamentos conectados. As Figuras 17 e 18 apresentam, respectivamente, a tela de visualização de consumo em tempo real e um gráfico detalhando o resultado de um experimento da tomada ligada com uma lâmpada incandescente com potência de 60W. O gráfico foi gerado através de recurso gráfico da plataforma openHAB.

Figura 17 - Tela de acompanhamento de visualização em tempo real



Fonte: Waka (2015).

Figura 18 - Resultado de experimento com carga resistiva por 24h



Fonte: Waka (2015).

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo é apresentado o desenvolvimento do protótipo de um plugue para tomada elétrica controlado remotamente. A seção 3.1 descreve os requisitos funcionais e não funcionais levantados para o desenvolvimento do protótipo, seguida da especificação do trabalho na seção 3.2, na qual se apresenta a arquitetura do protótipo desenvolvido, além dos diagramas de casos de uso. A seção 3.3 aborda detalhes da implementação e lista as ferramentas utilizadas no desenvolvimento, além de explicar o funcionamento do protótipo. Na seção 3.4 é apresentado o cenário experimental no qual o protótipo foi testado, e a seção 3.5 explana a metodologia empregada nos testes conduzidos. Ao final, na seção 3.6, os resultados obtidos a partir do presente projeto serão analisados.

3.1 REQUISITOS

O projeto proposto deve atender aos Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) apresentados no Quadro 1 e no Quadro 2, respectivamente. Todos os requisitos funcionais estão relacionados aos casos de uso apresentados na Figura 20.

Quadro 1 - Requisitos funcionais

Requisitos funcionais (RF)	**Casos de uso (UC)**
RF1: permitir ao usuário ativar e desativar o fornecimento de energia elétrica aos equipamentos conectados a um plugue	UC02, UC03, UC06, UC07, UC10
RF2: permitir ao usuário agendar horários de funcionamento de um plugue, indicando quando o plugue ele deve ativar ou desativar o fornecimento de energia elétrica	UC04, UC05, UC08, UC10
RF3: disponibilizar um relatório do consumo histórico dos equipamentos conectados a um plugue, na unidade de medida kWh	UC09, UC10, UC11
RF4: permitir que o usuário gerencie múltiplos plugues	UC01, UC10

Fonte: elaborado pelo autor.

Quadro 2 - Requisitos não funcionais

Requisitos não funcionais (RNF)
RFN1: consumir menos de 1W, ficando assim abaixo do consumo dos aparelhos eletroeletrônicos em modo *standby*
RNF2: apresentar compatibilidade com tomadas no padrão brasileiro, conforme a norma ABNT NBR 14136 (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002)
RNF3: utilizar conectividade Wi-Fi e permitir o controle remoto do plugue através de aplicativo móvel
RNF5: utilizar o módulo ESP8266
RNF6: o gerenciamento e comunicação com os plugues deve ser feito através do protocolo MQTT
RNF7: utilizar a IDE Visual Studio 2017 Community Edition com o *plugin* Visual Micro para o desenvolvimento do *firmware*.
RNF8: utilizar o padrão arquitetural Event Sourcing para salvar os eventos relacionados ao plugue

Fonte: elaborado pelo autor.

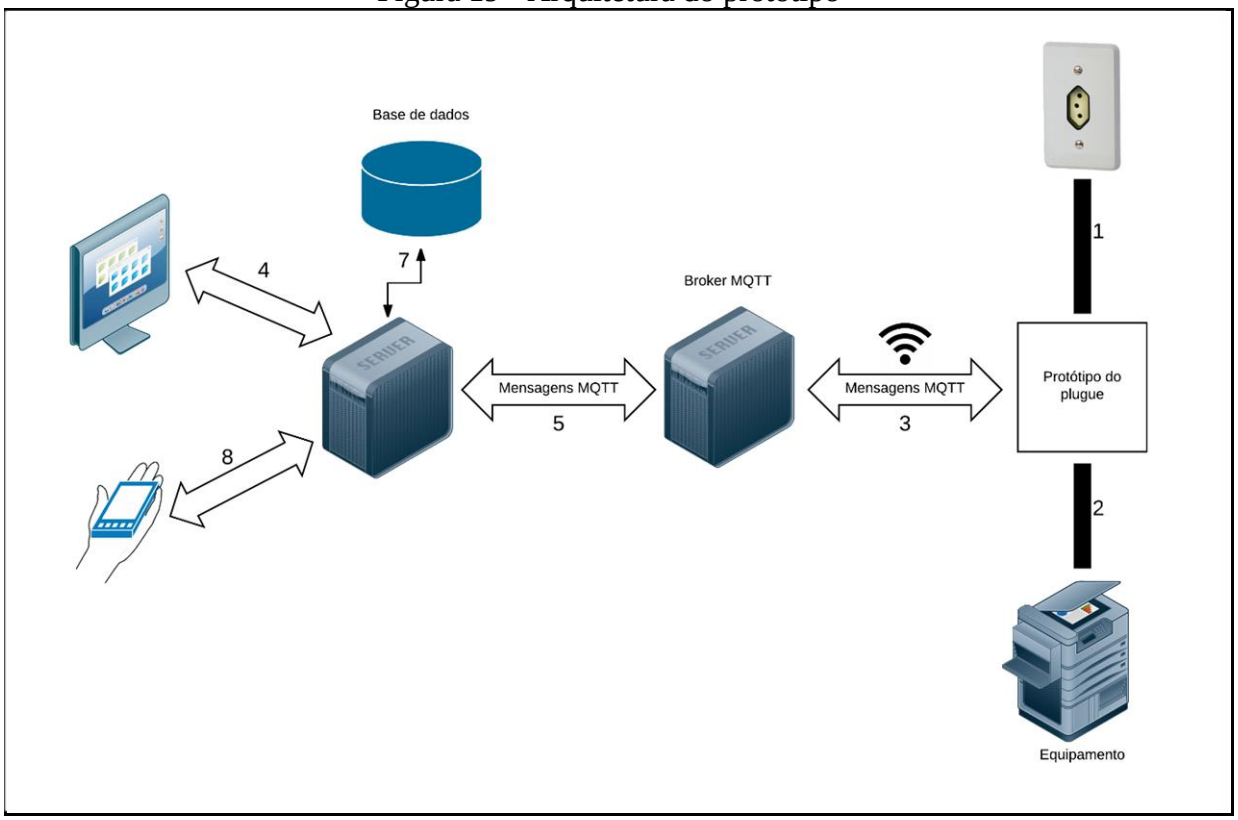
3.2 ESPECIFICAÇÃO

Nesta seção é realizada a especificação do protótipo de plugue para tomada elétrica controlado remotamente. Primeiramente, a arquitetura do protótipo é apresentada, sendo seguida por uma explicação de como o plugue pode ser configurado e utilizado para controlar o ligamento e desligamento de outros dispositivos eletroeletrônicos. Adicionalmente, são apresentados os diagramas de casos de uso e de sequência, desenvolvidos com a ferramenta `StarUML`.

3.2.1 Arquitetura

A Figura 19 apresenta a arquitetura do protótipo desenvolvido. Primeiramente, o usuário conecta o plugue à uma tomada elétrica ou a uma extensão (1), e na outra extremidade, um equipamento eletroeletrônico que deseje controlar (2). A partir de um navegador de Internet o usuário pode configurar o plugue através de um portal Web servido por uma aplicação servidora (4 e 8). Através deste portal, o usuário pode ligar ou desligar o equipamento conectado a ele, além agendar horários específicos para essas operações. Uma vez configurado, plugue manterá conexão constante com o *broker* MQTT (3), medindo o consumo de energia elétrica do equipamento (6) num intervalo de 5 segundos. A aplicação servidora, por sua vez, inscrita no *broker* no tópico das medições, armazena as leituras em uma base de dados (7). Os comandos para ligar ou desligar o equipamento, assim como agendamentos para essas operações, são enviados para o plugue na forma de mensagens MQTT através do *broker* (5).

Figura 19 - Arquitetura do protótipo



Fonte: elaborado pelo autor.

3.2.2 Casos de uso

Este projeto contém onze casos de uso (em inglês Use Case – UC), apresentados na Figura 20, e que apresentam as principais funcionalidades do protótipo desenvolvido, representados através da Unified Modeling Language (UML).

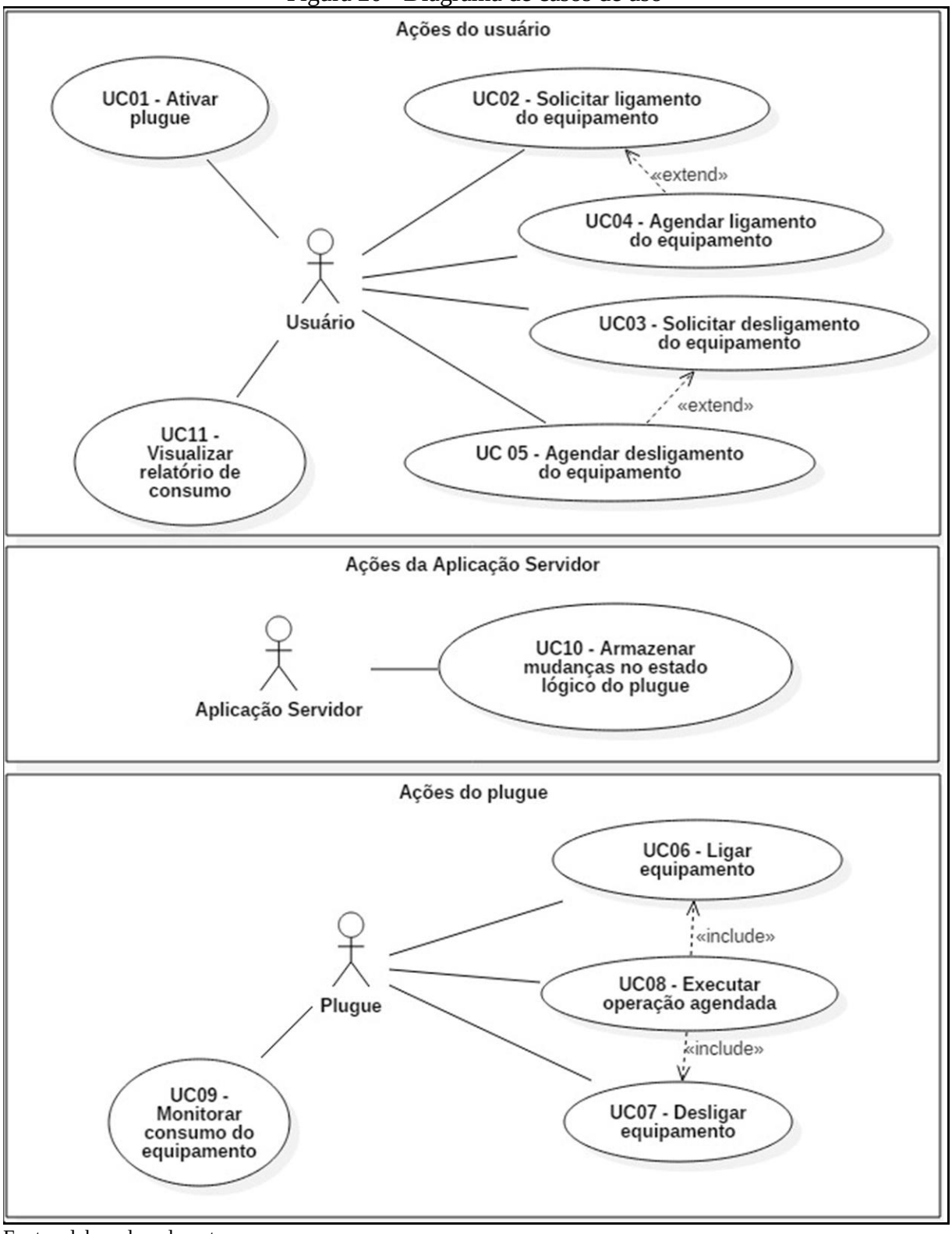
O caso de uso `Ativar plugue` descreve o processo de ativação do dispositivo, no qual o usuário nomeia o dispositivo e a aplicação confere um identificador ao mesmo. Uma vez ativado o plugue, o usuário pode solicitar o ligamento (UC02) e o desligamento (UC03) do equipamento conectado ao plugue. O usuário também pode agendar o ligamento (UC04 ou desligamento (UC05) do equipamento para um momento futuro.

Por sua vez, o caso de uso `Armazenar mudanças no estado lógico do plugue` (UC10) refere-se ao salvamento para consulta posterior de todos os eventos relacionados ao plugue, como ativação, mudanças de estado e medições realizadas.

Os casos de uso `Ligar Equipamento` (UC06) e `Desligar equipamento` (UC07) representam as ações por parte do plugue em resposta às solicitações do usuário, nos casos de uso UC02 e UC03. Além destes, o caso de uso `Executar operações agendadas` (UC08) descreve a execução destas duas operações de forma agendada.

Já o caso de uso `Monitorar consumo do equipamento` (UC09) representa o monitoramento do consumo de energia por parte do equipamento, realizado pelo plugue com frequência determinada. Além disso, o caso de uso `Visualizar relatório de consumo` (UC11) detalha o relatório do consumo histórico de energia pelo dispositivo conectado ao plugue.

Figura 20 - Diagrama de casos de uso



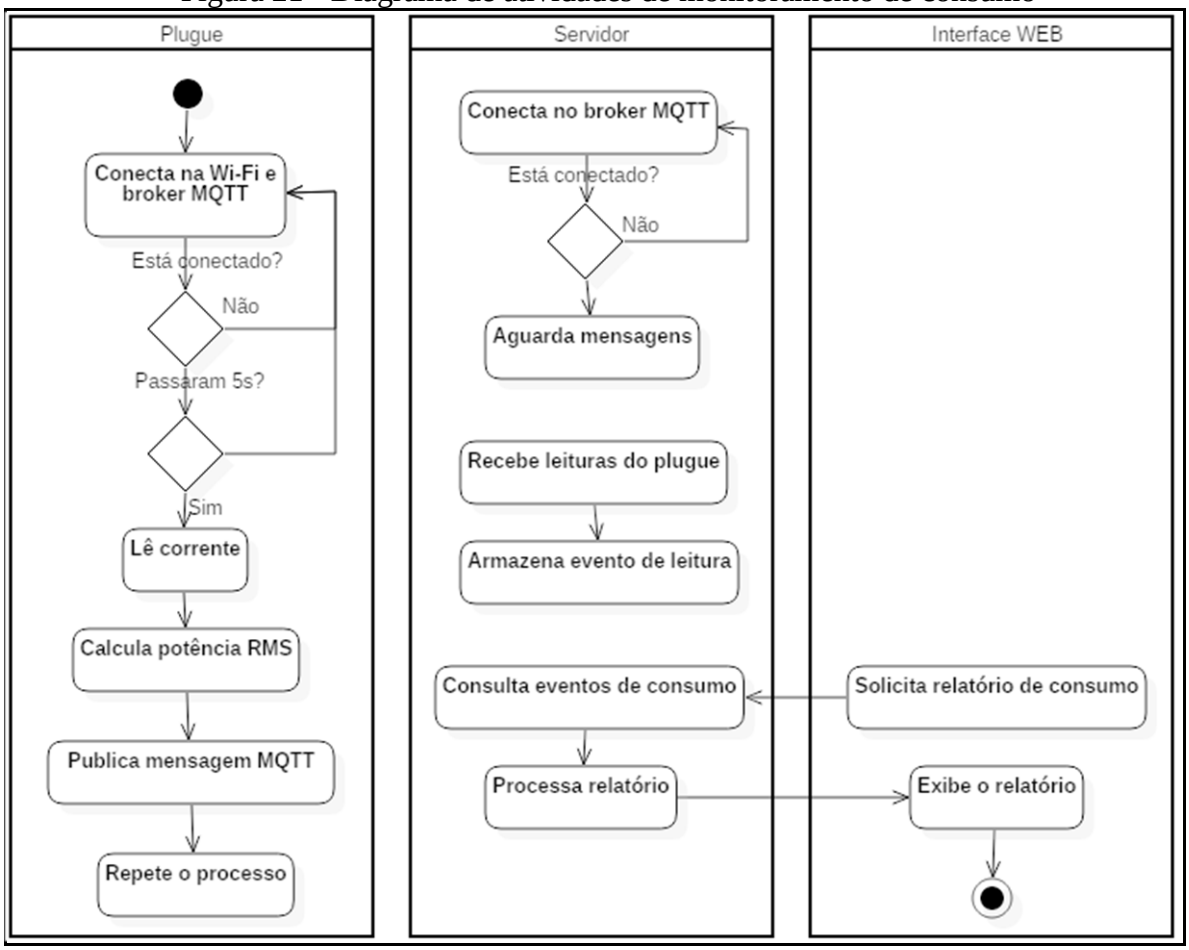
Fonte: elaborado pelo autor.

Na Figura 21 pode-se observar que o processo de monitoramento de consumo de energia é dividido em três partes, da seguinte forma:

a) a primeira etapa corresponde ao monitoramento por parte do plugue instalado, conectado ao *broker* MQTT através de conexão Wi-Fi;

b) a segunda etapa corresponde ao armazenamento das leituras por parte do servidor;
c) a terceira corresponde à solicitação do relatório de consumo por parte do usuário através da aplicação Web.

Figura 21 - Diagrama de atividades de monitoramento de consumo

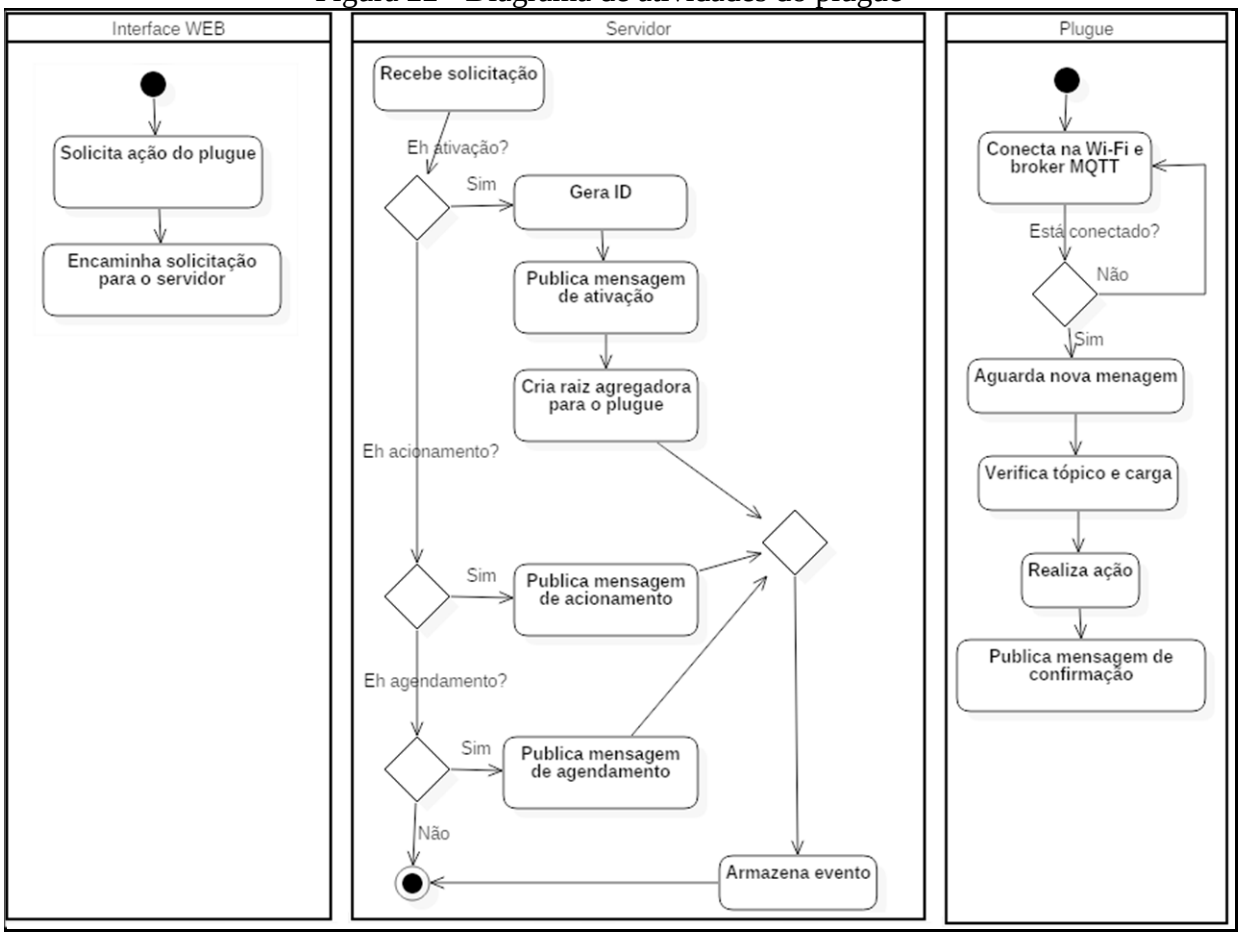


Fonte: elaborado pelo autor.

Na Figura 22 pode-se ver o fluxo da aplicação quando o usuário solicita alguma ação por parte do plugue. Este processo se dá em três etapas:

a) na primeira etapa, o usuário solicita uma ação por parte do plugue, que é encaminhada para o servidor. Esta ação pode ser uma ativação, acionamento ou agendamento;
b) a segunda etapa detalha o processamento realizado pelo servidor para cada tipo de solicitação;
c) a terceira etapa descreve o processamento destas solicitações por parte do plugue.

Figura 22 - Diagrama de atividades do plugue



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para implementar o trabalho proposto foram utilizados dois conjuntos distintos de ferramentas. O primeiro conjunto é referente à aplicação denominada Servidor, e o segundo consiste nas ferramentas usadas para o desenvolvimento do protótipo em hardware.

A aplicação Servidor foi implementada na linguagem de programação C#, no ambiente de desenvolvimento Jetbrains Rider e .NET Framework 4.6.1. Para a implementação do servidor HTTP foi utilizado o *microframework* `NancyFX` e as bibliotecas `Topshelf` e `Newtonsoft.Json`. Todos os dados de eventos relacionados ao plugue são armazenados em uma base de dados PostgreSQL, utilizando-se o recurso Document Storage para armazenamento não relacional dos dados. Os eventos são salvos utilizando o padrão arquitetural Event Sourcing através da biblioteca `Marten`. Para a comunicação com o plugue, utilizando o

protocolo MQTT, foi utilizada a biblioteca `M2Mqtt.Net`, e o *broker* público utilizado foi o mantido pela Eclipse Foundation em `iot.eclipse.org`. Por último, também foi utilizada a biblioteca `Polly` para implementar uma estratégia de reconexão com o *broker* MQTT em caso de queda de conexão. Para implementar os testes de unidade, foram utilizadas as bibliotecas `NUnit` e `FluentAssertions`.

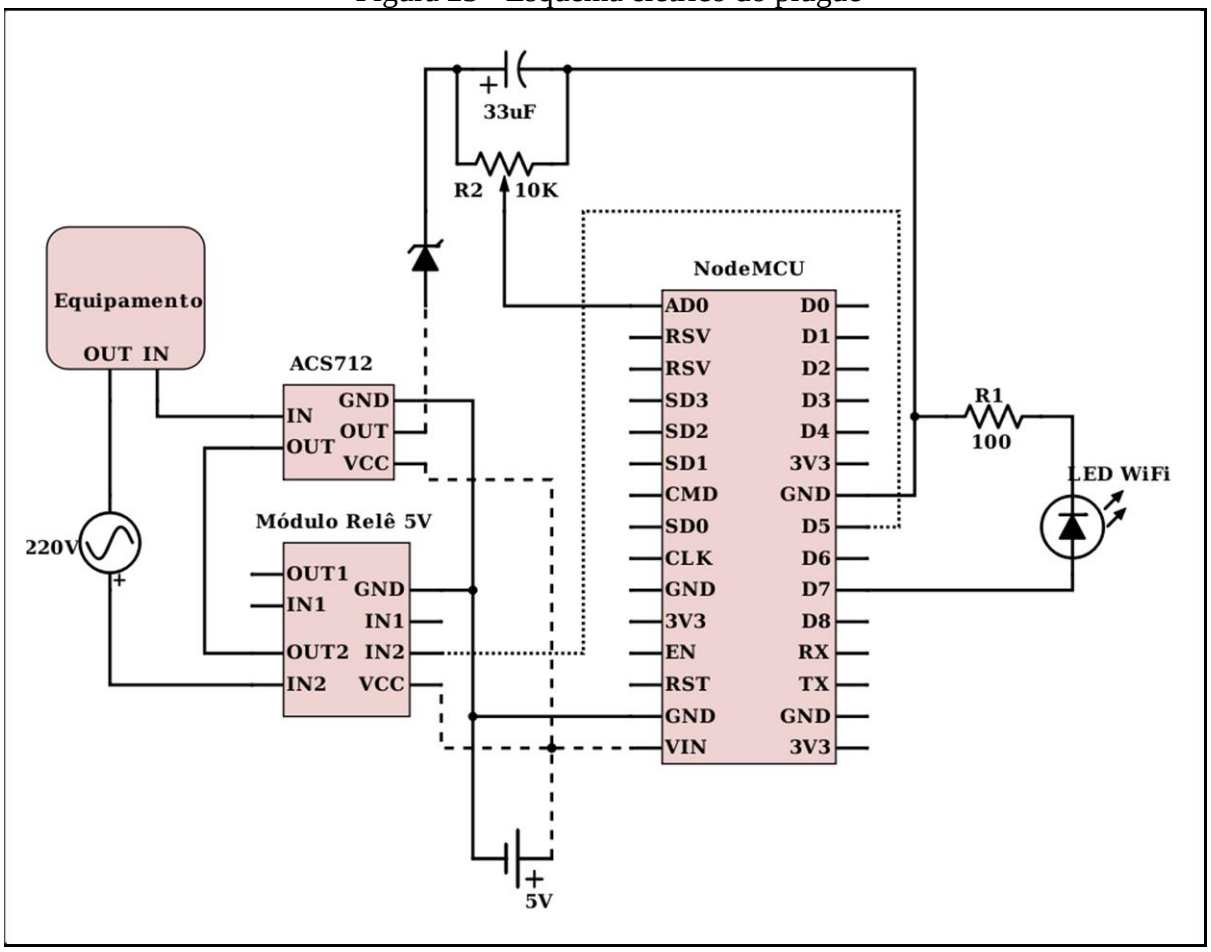
A interface de usuário foi desenvolvida como uma aplicação Web, utilizando HTML 5, Cascade Style Sheets (CSS) e Javascript. Entre as bibliotecas utilizadas, destacam-se o `jQuery`, `Bootstrap` e `Mustache`.

Para o desenvolvimento do plugue foi utilizada a placa de desenvolvimento NodeMCU, com o módulo ESP8266. O firmware foi implementado na linguagem de programação C++ nos ambientes de desenvolvimento Arduino IDE e Visual Studio com o *plugin* Visual Micro. Para a comunicação através do protocolo MQTT, foram utilizadas as bibliotecas `ESP8266WiFi` e `PubSubClient`, além da biblioteca `TickerScheduler` para executar os eventos agendados.

3.3.2 Construção do hardware

Para construir o hardware do protótipo, foram utilizados um NodeMCU, baseado no módulo ESP8266, um módulo relê para Arduino com dois canais, um sensor de corrente elétrica ACS712 e uma fonte chaveada de 5V. Além disso, foi necessária a utilização de outros componentes, como fios, resistores, potenciômetros, LEDs, capacitores eletrolíticos, entre outros e diodos Zener. No Apêndice A pode ser encontrada a relação de todos os componentes que compõem o protótipo, assim como o preço de cada um dos componentes. Na Figura 23 pode ser visto o esquema elétrico do plugue, feito na ferramenta `Scheme.it`.

Figura 23 - Esquema elétrico do plugue



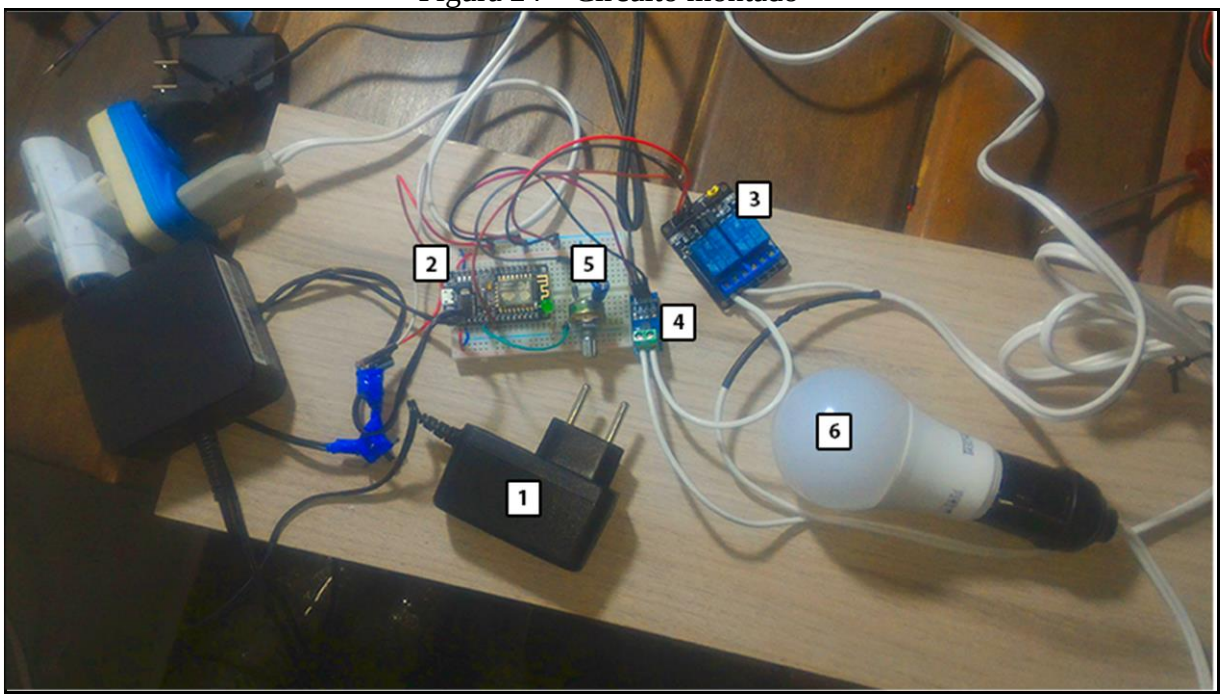
Fonte: elaborado pelo autor.

Como pode ser visto na Figura 23, a fonte chaveada é ligada diretamente na entrada AC. A fonte *bivolt* suporta tensões de entrada na faixa de 100V a 240V AC e apresenta saída de 5V DC com corrente de até 2A. A saída em 5V é utilizada para alimentar o NodeMCU através do pino VIN, que necessita de tensões entre 3,3V e 5V, além do módulo e relê e do sensor de corrente.

Apenas um dos canais de 10A do módulo relê é utilizado. Este é controlado através da porta D5 do NodeMCU, ligando ou desligando qualquer equipamento conectado ao plugue. Como o equipamento eletroeletrônico deve estar ligando somente após a ativação pelo usuário, o equipamento fica ligado à saída Normally Closed (NC) do relê. Um LED verde é utilizado para indicar o status da conexão WiFi. O LED é ligado na porta D7, com a ajuda de um resistor de 100 Ohm.

A leitura do sensor de corrente ACS712, alimentado com 5V, é realizada através da porta analógica (AD0) do módulo. A tensão de saída do sensor é, no entanto, reduzida para 3,3V. Pode-se observar o circuito montado na Figura 24.

Figura 24 – Circuito montado



Fonte: fotografado pelo autor.

A seguir, pode-se conferir o detalhamento do hardware, conforme a Figura 24:

a) o item 1 é a fonte de alimentação responsável por alimentar o NodeMCU, o sensor de corrente e o relê. Sua tensão de entrada é de 100V a 240V, e a tensão de saída é de 5V, com capacidade de carga de 2A. O NodeMCU é alimentado através da porta VIN;

b) o item 2 consiste na placa de desenvolvimento NodeMCU, com um módulo ESP8266;

c) o item 3 é o módulo relê de dois canais, responsável pela ativação dos equipamentos. É alimentado com a saída do item 1, com tensão de 5V, e é acionado através da porta D5 do item 2;

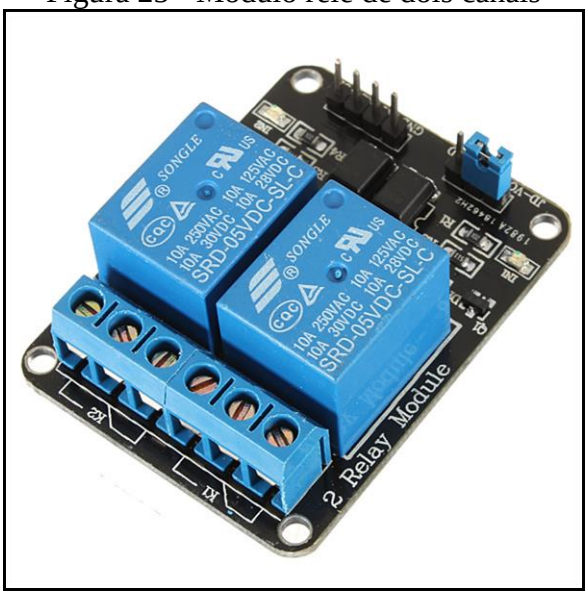
d) o item 4 é o sensor de corrente ACS712. Ele é responsável por monitorar o consumo de quaisquer cargas conectadas ao plugue. Este sensor é alimentado com tensão de 5V pela saída do item 1. A saída analógica é lida pelo item 2 através da porta AD0;

e) o item 5 se trata de um circuito para reduzir a tensão de saída do item 4 para 3,3V, a tensão requisitada pelo NodeMCU nessa porta. Este circuito é composto por um potenciômetro de 10K Ohm, um capacitor eletrolítico de 30uF e um diodo Zener. Este circuito pode ser visto também na Figura 23;

f) o item 6 representa uma carga resistiva, ou qualquer equipamento ligado ao plugue. Para fins de exemplo, foi utilizada uma lâmpada de LED.

O módulo relê, que pode ser visto na Figura 25, possui dois canais de 10 amperes e é alimentado com 5V. Apenas o segundo canal é utilizado no protótipo. O módulo apresenta duas entradas digitais, cada qual para controlar o acionamento de um dos canais. A segunda entrada é utilizada para ativar ou desativar o relê, ligando ou desligando o equipamento conectado ao plugue. Estas entradas requerem uma tensão de 2V a 5V para serem ativados.

Figura 25 - Módulo relê de dois canais

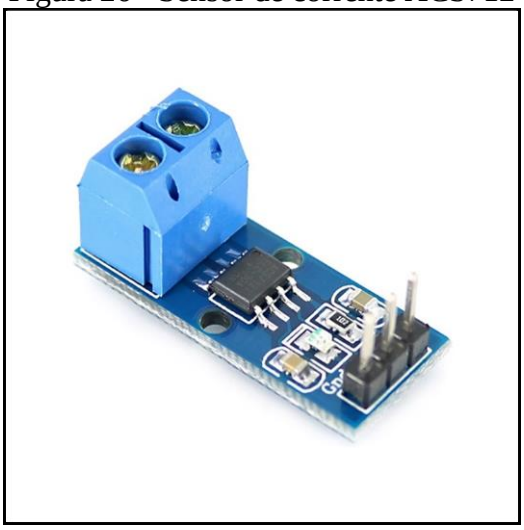


Fonte: FilipeFlop (2017).

Para calcular o consumo de energia elétrica pelo equipamento ligado ao plugue, é necessário primeiro conhecer a potência do equipamento. Neste trabalho, considera-se que a tensão da rede elétrica seja de 220V constantes. A corrente é obtida através do uso do ACS712, um sensor de corrente bidirecional que pode ser utilizado para ler tanto corrente contínua quanto alternada. O modelo empregado, que pode ser visto Figura 26, realiza leituras na escala de -30A a 30A, apresentando sensibilidade de 66mV/A e erro na saída de até 1,5% quando estiver operando em temperatura ambiente de 25ºC.

O sensor mapeia a corrente lida numa escala de 0 a 5V, correspondendo a um mapeamento direto da escala de -30A a 30A. Como o sensor é utilizado para medir corrente alternada, as leituras dos valores na porta analógica correspondem a 0V para -30A, e 5V para 30A. A ausência de corrente, desta forma, corresponde a 2,5V. Como a tensão de saída está sendo reduzida para 3,3V, a ausência de valor passa a corresponder a 1,66V. A porta lógica AD0 no NodeMCU possui um conversor analógico-digital, que por sua vez, mapeia a escala 0 a 5V com uma precisão de 10 *bits*, para a faixa de 0 a 1023.

Figura 26 - Sensor de corrente ACS712



Fonte: FilipeFlop (2017).

3.3.3 A aplicação Servidor

A aplicação Servidor fornece uma Interface de Programação de Aplicativos (em inglês Application Programming Interface – API) através de *endpoints* HTTP no padrão REST.

O Quadro 3 lista os *endpoints* através dos quais é possível controlar um plugue. Todos os *endpoints* apresentam seus resultados no formato JSON.

Quadro 3 - Endpoints para controle do plugue

Método HTTP	**Endpoint**	**Função**
POST	/plugs/activate	Ativa um plugue. Esta operação retorna o `id` do plugue, no formato Globally Unique Identifier (GUID).
POST	/plugs/deactivate	Desativa um plugue, desassociando-o do seu histórico de eventos.
POST	/plugs/reactivate	Reativa um plugue, resassociando-o ao seu histórico de eventos.
GET	/plugs/	Obtém uma lista com dados de todos os plugues.
GET	/plugs/{id}	Obtém os dados do plugue com identificador `id`.
POST	/plugs/{id}/try-turn-on	Publica mensagem instruindo ligar o equipamento conectado ao plugue com identificador `id`.
POST	/plugs/{id}/try-turn-off	Publica mensagem instruindo desligar o equipamento conectado ao plugue com identificador `id`.
POST	/plugs/{id}/scheduling/turn-on	Publica mensagem instruindo o agendamento para ligar o equipamento conectado ao plugue com identificador `id` após um determinado intervalo.
POST	/plugs/{id}/scheduling/turn-off	Publica mensagem instruindo o agendamento para desligar o equipamento conectado ao plugue com identificador `id` após um determinado intervalo.
POST	/plugs/{id}/rename	Substitui o nome do plugue com identificador `id`.
GET	/plugs/{id}/reports/consumption	Obtém um relatório de consumo do equipamento ligado ao plugue com identificador `id`.
GET	/plugs/{id}/reports/timeline	Obtém os dados do relatório Linha do Tempo do plugue com identificador `id`.

Fonte: elaborado pelo autor.

Durante o processo de inicialização da aplicação Servidor, ocorre a configuração da biblioteca `M2Mqtt.Net`. Como pode ser visto no Quadro 4, a implementação atual não implementa nenhuma questão de segurança, visto que não estava previsto na proposta. Além disso, a aplicação está utilizando o *broker* público mantido pela Eclipse Foundation para prototipação de soluções em IoT.

Quadro 4 - Inicialização do *broker MQTT*

```
const string brokerHostName = "iot.eclipse.org";
const int brokerPort = 1883;
_mqttClient = new MqttClient(
    brokerHostName:brokerHostName,
    brokerPort: brokerPort,
    secure: false,
    sslProtocol: MqttSslProtocols.None,
    userCertificateSelectionCallback: null,
    userCertificateValidationCallback: null
);
```

Fonte: elaborado pelo autor.

Ao receber uma requisição para um dos *endpoints* POST que precisam de uma ação por parte do plugue, a aplicação publica uma mensagem para o *broker* MQTT, conforme pode ser visto no Quadro 5.

Quadro 5 - Mensagem MQTT publicadas pela aplicação Servidor

Endpoint	**Tópico MQTT**	**Carga**
/plugs/{id}/try-turn-on	/smart-plug/state	`Id` e `turn-on`
/plugs/{id}/try-turn-off	/smart-plug/state	`Id` e `turn-off`
/plugs/{id}/scheduling/turn-on	/smart-plug/schedule-on	`Id` e Tempo em milisegundos
/plugs/{id}/scheduling/turn-off	/smart-plug/schedule-off	`Id` e Tempo em milisegundos

Fonte: elaborado pelo autor.

3.3.4 Processamento das mensagens recebidas pelo plugue

O *firmware* do plugue, por sua vez, faz uso da biblioteca `PubSubClient` para se inscrever nos tópicos descritos no Quadro 5, publicados pela aplicação Servidor, e para publicar mensagens em tópicos que serão apresentados mais à frente. Além de inscrever-se nos tópicos publicados pela aplicação Servidor, o *firmware* configura uma função *callback* para manipular quaisquer mensagens sejam encaminhadas pelo *broker*. Isso é feito através da função `setCallBack` da classe `PubSubClient`, que é chamada passando como parâmetro a função `mqtt_callback`, que pode ser vista na Figura 27.

Figura 27 - Função responsável por processar as mensagens do broker MQTT

```
void mqtt_callback(char* topic, byte* payload, unsigned int length) {
  String message;
  for (int i = 0; i < length; i++) {
    char c = (char)payload[i];
    message += c;
  }

  auto topicString = String(topic);

  if (topicString == "/smart-things/plug/clean-identity") {
    Serial.println("identity cleared");
    eepromManager.clear();
    initCredentials();
    return;
  } else if (topicString == "/smart-things/plug/activate") {
    if (!hasCredentials) {
      auto newId = message;
      Serial.println(newId);
      eepromManager.saveId(newId);
      initCredentials();
      return;
    }
    else return;
  }

  if (!hasCredentials) {
    return;
  }

  auto delimiterIndex = message.indexOf('|');
  if (delimiterIndex == -1) {
    return;
  }

  auto targetId = message.substring(0, delimiterIndex);
  auto data = message.substring(delimiterIndex + 1);

  if (id != targetId) {
    return;
  }

  if (topicString == "/smart-things/plug/state") {
    if (data == "turn-on") {
      turnOn();
    }
    else if (data == "turn-off") {
      turnOff();
    }
  } else if (topicString == "/smart-things/plug/schedule-on") {

    auto intervaloMilisegundos = data.toInt();
    scheduler.schedule(turnOn, intervaloMilisegundos);
  } else if (topicString == "/smart-things/plug/schedule-off") {

    auto intervaloMilisegundos = data.toInt();
    scheduler.schedule(turnOff, intervaloMilisegundos);
  }

  Serial.flush();
}
```

Fonte: elaborado pelo autor.

Ao receber uma mensagem do *broker*, o método `mqtt_callback` verifica a qual dos tópicos inscritos a mensagem se refere. Ao identificar que se trata de uma mensagem com o tópico `/smart-plug/state`, ele verifica o conteúdo da carga. Caso o conteúdo seja o texto `turn-on`, ele executa o método `turnOn`, responsável por ativar o relê, liberando a passagem de corrente elétrica para o equipamento ligado ao plugue. Ao identificar que o conteúdo da carga é o texto `turn-off`, a função `mqtt_callback` executa o método `turnOff`, que desativa o relê, desligando o equipamento conectado ao plugue.

Quando o tópico da mensagem for `/smart-plug/schedule-on`, o método `mqtt_callback` primeiramente lê a carga, que contém o intervalo em milissegundos que deve aguardar até executar a ação de ligar o relê. Em seguida, utiliza a biblioteca `TickerScheduler` para agendar a execução de uma função anônima após o intervalo definido na carga. A função anônima, por sua vez, ativa o relê, da mesma forma que ocorre com mensagens com tópico `/smart-plug/state`.

Já quando a mensagem recebida for para o tópico `/smart-plug/schedule-off`, o procedimento será o mesmo, com a diferença de que o relê será desativado.

Após ativar ou desativar o relê, o que pode acontecer em cada um dos quatro casos supracitados, o *firmware* publica uma mensagem MQTT confirmando a operação. Essa mensagem está atrelada ao tópico `/smart-plug/new-state`, e a carga é o texto `on`, caso o relê tenha sido ativado, e `off` quando tiver sido desativado.

3.3.5 Monitoramento do consumo de energia elétrica

Dado que a medição trata de corrente alternada, cada medição é composta da seguinte forma. Primeiro, são coletadas mil leituras do pino analógico durante o intervalo de um segundo. De cada valor lido, é decrescido o valor denominado *offset* do sensor, equivalente a 2,5V. Para mitigar o efeito de picos em leituras específicas do sensor, são acumulados os valores de 1500 leituras consecutivas, segundo a fórmula $\sqrt[2]{\sum_{0}^{1500} l^2}$, onde `l` é um valor entre 0 e 1023. O valor resultante é então multiplicado o valor da proporção de tensão para corrente e dividido pela sensibilidade do sensor. O resultado desta operação é a corrente elétrica. Por fim, o valor da corrente é então multiplicado pela tensão de 220V, obtendo assim a potência.

Uma vez calculada a potência, o *firmware* publica uma mensagem informando a aplicação Servidor da nova leitura. Essa mensagem é publicada a cada segundo, com o tópico `/smart-plug/consumption` e a carga sendo a potência calculada.

3.3.6 Armazenamento dos eventos do plugue

Quando uma ação muda o estado do plugue, a aplicação Servidor acrescenta o evento correspondente à cadeia de eventos do plugue em uma base de dados PostgreSQL, em modelo não relacional, usando a biblioteca `Marten`.

Para salvar um evento usando o padrão arquitetural Event Sourcing, a biblioteca requer dois parâmetros: o identificador do grupo de eventos, e um objeto representando o evento. Além disso, é necessário informar o tipo da classe que representa a raiz agregadora. Essas informações permitem que a biblioteca crie duas tabelas. A primeira delas representa a raiz agregadora e tem sempre um registro, representando o estado atual da cadeia de eventos. Na implementação deste trabalho, a classe utilizada `Plug` foi utilizada como raiz agregadora, e a tabela gerada leva o nome `mt_doc_plug`. A segunda tabela é a `mt_events`, na qual ficam são armazenados todos os eventos.

Ao utilizar o identificador único do plugue como identificador do grupo de eventos, o padrão Event Sourcing permite reconstruir o estado do plugue, com todos as suas propriedades, em qualquer ponto do tempo desde a ativação do plugue. Isto é possível porque a biblioteca `Marten` inclui automaticamente nos metadados do evento uma marca de horário a novo cada evento da cadeia, permitindo que sejam ordenados e aplicados em sequência na raiz agregadora.

A classe `PlugEventSequencer` disponibiliza uma interface para acrescentar cada um dos possíveis eventos à cadeia cujo identificador é o GUID do plugue. Quando um novo plugue é ativado, o evento `PlugActivated` é salvo com o identificador e o nome do plugue. Ao renomear um plugue, o evento `PlugRenamed` é salvo, contendo apenas o novo nome do plugue. Depois de confirmada a mudança de estado do relê, um dos seguintes é salvo: `PlugTurnedOn` ou `PlugTurnedOff`. Os dois eventos são meramente informativos e não contêm informações adicionais. Por fim, ao ser notificada de uma nova leitura, a aplicação Servidor acrescenta o evento `ConsumptionReadingReceived`, contendo a potência calculada pelo plugue.

Por convenção da biblioteca `Marten`, a raiz agregadora deve implementar um método `Apply` para cada evento suportado. Esses métodos são, dessa forma, responsáveis por alterar o estado atual do plugue de acordo com o evento em questão. No Quadro 6 pode ser vista a classe `Plug`, juntamente com a aplicação de cada um dos eventos. Após aplicada a alteração correspondente, a instância da classe `Plug` reflete a alteração no estado observada no momento em que o evento foi originalmente acrescentado à cadeia.

Quadro 6 - Raiz agregadora que representa um plugue

```
public class Plug
{
    public Guid Id { get; set; } = PlugIds.PlugOneId;
    public PlugState CurrentState { get; set; } = PlugState.Off;
    public string Name { get; private set; } = "PlugOne";
    public double LastConsumptionInWatts { get; private set; }

    public bool IsOn() => CurrentState == PlugState.On;

    public void Apply(PlugActivated activation)
    {
        Id = activation.PlugId;
        Name = activation.PlugName;
    }

    public void Apply(PlugTurnedOn plugTurnedOn)
    {
        CurrentState = PlugState.On;
    }

    public void Apply(PlugTurnedOff plugTurnedOff)
    {
        CurrentState = PlugState.Off;
    }

    public void Apply(ConsumptionReadingReceived consumptionReading)
    {
        LastConsumptionInWatts = consumptionReading.ConsumptionInWatts;
    }

    public void Apply(PlugRenamed plugRenamed)
    {
        Name = plugRenamed.NewName;
    }
}
```

Fonte: elaborado pelo autor.

Para gerar o relatório de consumo de energia para um determinado plugue, o procedimento realizado consiste em filtrar, dentre todos os eventos, apenas aqueles associados ao identificador do plugue em questão e que sejam do tipo `ConsumptionReadingReceived`. A partir da lista de eventos obtida, e mais a marca de horário do evento, é gerado o relatório de consumo do equipamento conectado ao plugue.

3.3.7 Operacionalidade da implementação

A aplicação Servidor disponibiliza um portal Web na porta HTTP 8001, através do qual o usuário pode controlar o plugue. Para fins de exemplo, o usuário pode acessar o portal no endereço `http://localhost:8001/admin`.

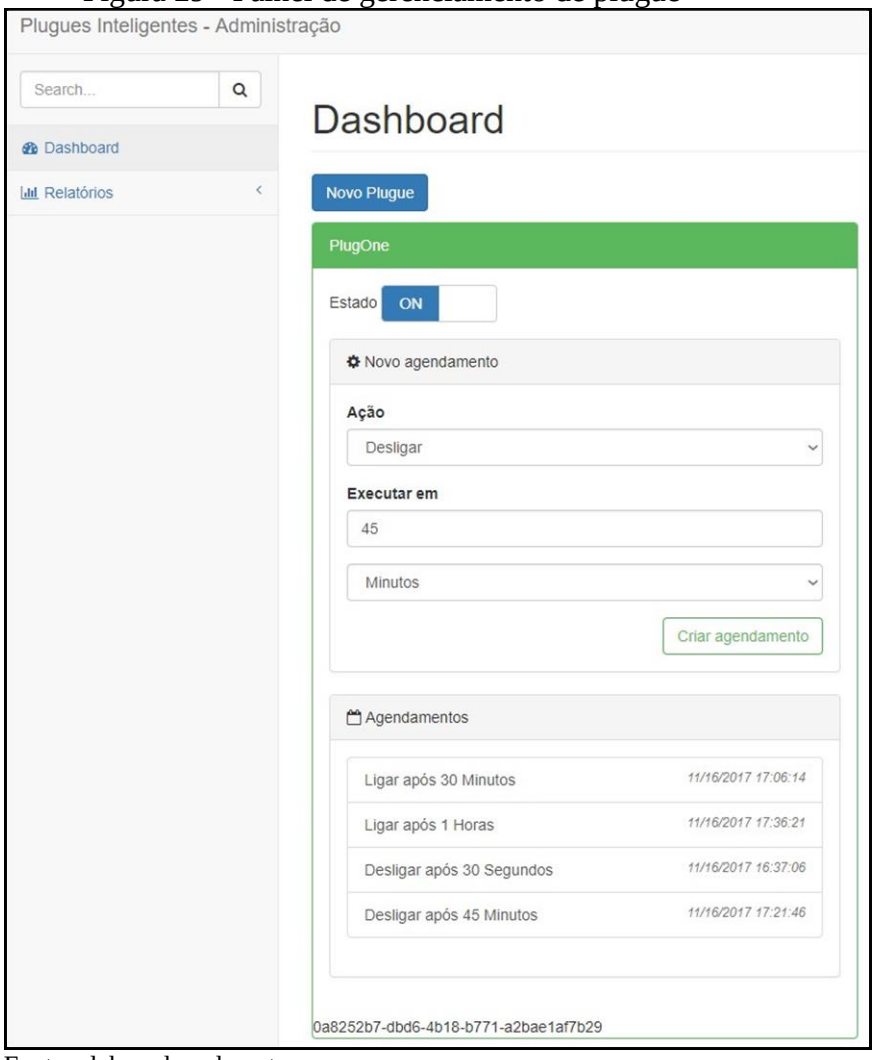
Uma vez que o usuário acessar o portal, será apresentada a interface de ativação do plugue, como pode ser visto na Figura 28. Após o usuário informar o nome do plugue, o botão `Ativar` é habilitado.

Figura 28 – Tela de ativação de plugue

Fonte: elaborado pelo autor.

Em seguida, depois de o usuário ativar o plugue, a aplicação gera um painel de gerenciamento, exclusivamente para o plugue ativado, e que pode ser visto na Figura 29. O usuário tem a opção de cadastrar quantos plugues forem necessários. Esta tela oferece as seguintes opções ao usuário: ativar ou desativar o plugue, manter agendamentos e visualizar o relatório de consumo.

Figura 29 - Painel de gerenciamento de plugue



Fonte: elaborado pelo autor.

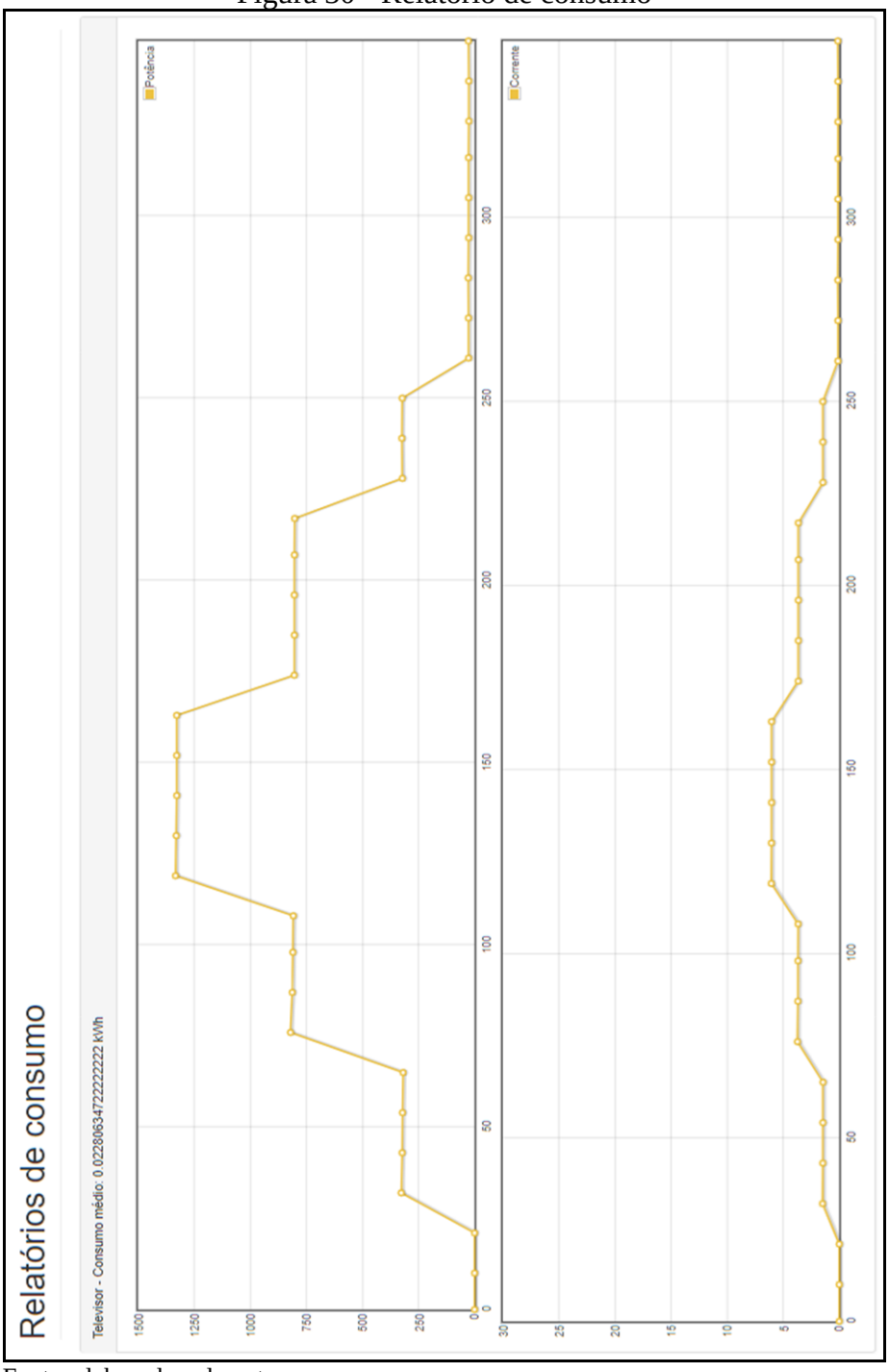
Caso o usuário acione o botão para alternar o estado do plugue, o equipamento ligado ao plugue será ligado ou desligado em instantes. Logo abaixo do botão para acionamento do plugue, há uma área que permite ao usuário agendar ações para serem executadas posteriormente, num momento determinado pelo usuário. Como pode ser visto na Figura 29, o usuário pode informar o tipo da ação e a quantidade de tempo no futuro após o qual a ação será executada, além de selecionar a tipo de intervalo, dentre as seguintes opções: `Dias`, `Horas`, `Minutos` e `Segundos`.

Quando o usuário informar o intervalo desejado e optar por `Criar agendamento`, a aplicação comunica o plugue do agendamento. Abaixo do painel `Novo agendamento` pode ser encontrado a listagem de agendamentos. Cada agendamento criado pelo usuário é adicionado a esta listagem, informando a ação agendada, e o horário previsto para execução.

O usuário pode também escolher visualizar o relatório de consumo ao selecionar a opção `Relatório`. Quando ele o fizer, será redirecionado para o relatório de consumo, que pode ser

visto na Figura 30. Esse relatório consiste num gráfico de linha, que apresenta o consumo em Watts ao longo do tempo, com resolução de 5 segundos. Abaixo do gráfico de consumo, é apresentado também o relatório de corrente elétrica, em Amperes. A implementação atual não apresenta a opção de filtrar períodos, buscando todas as medições desde a ativação do plugue.

Figura 30 - Relatório de consumo



Fonte: elaborado pelo autor.

No relatório `Linha do Tempo`, visível na Figura 31, o usuário pode acompanhar todo o histórico de eventos de um plugue, desde a sua ativação. Cada evento apresenta uma descrição

curta e o momento em que ocorreu. No caso de agendamentos, os eventos contam também com uma descrição longa.

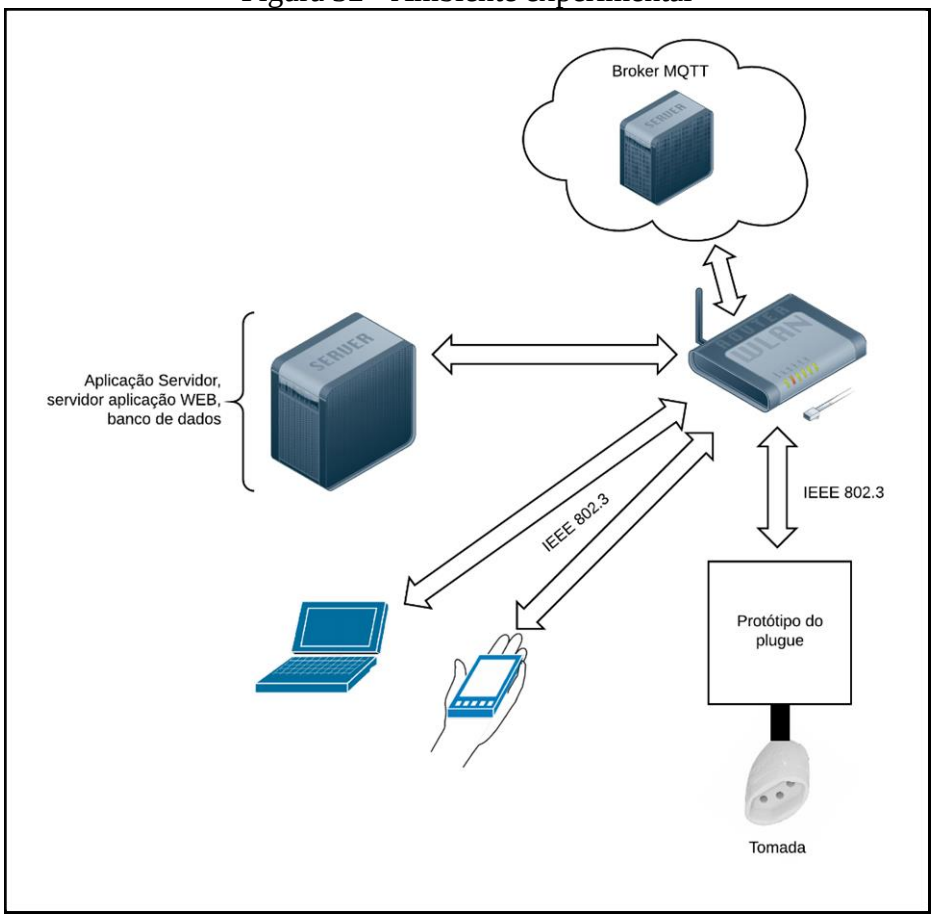
Figura 31 - Linha do Tempo

Fonte: elaborado pelo autor.

3.4 AMBIENTE EXPERIMENTAL

O ambiente no qual foram realizados os experimentos foi em uma rede local residencial, conforme pode ser visto na Figura 32. Pode-se notar que o *broker* MQTT utilizado para teste está rodando fora da rede.

Figura 32 - Ambiente experimental



Fonte: elaborado pelo autor.

3.5 METODOLOGIA

Existem três tipos de cargas elétricas que são representativas da grande maioria dos equipamentos eletroeletrônicos usados em meio residencial: cargas resistivas, cargas capacitivas e cargas indutivas. Para testar o monitoramento do consumo de energia, foram criados cenários de teste para cada um dos três tipos de carga, conforme pode ser visto no Quadro 7.

Quadro 7 - Cenários de teste com diferentes tipos de carga elétrica

Tipo de carga elétrica	**Equipamento utilizado**
Resistiva	Aquecedor elétrico, 1200W, 220V / 60hz
Capacitiva	2 lâmpadas fluorescentes, 59W, 220V / 60hz
Indutiva	Aspirador de pó, 1400W, 220V / 60hz

Fonte: elaborado pelo autor.

O teste de cada um dos cenários consistiu na observação do sistema por um período determinado, durante o qual a corrente elétrica foi aferida com o auxílio de alicate amperímetro do modelo M266, apresentado na Figura 33, e registrado para verificação.

Figura 33 - Alicate amperímetro

Fonte: fotografado pelo autor.

A fim de validar o agendamento de ações do plugue, foi realizada uma sequência de agendamentos consecutivos, e verificado se esses eventos ocorreram, conforme esperado, e se ocorreram no momento certo.

3.6 ANÁLISE DOS RESULTADOS

O protótipo implementado atingiu seu principal objetivo, pois permitiu o controle remoto de equipamentos conectados ao plugue, tanto em caráter remoto quanto agendado, assim como o monitoramento do consumo de energia desses equipamentos, disponibilizando um relatório de consumo em um portal Web acessível de um navegador.

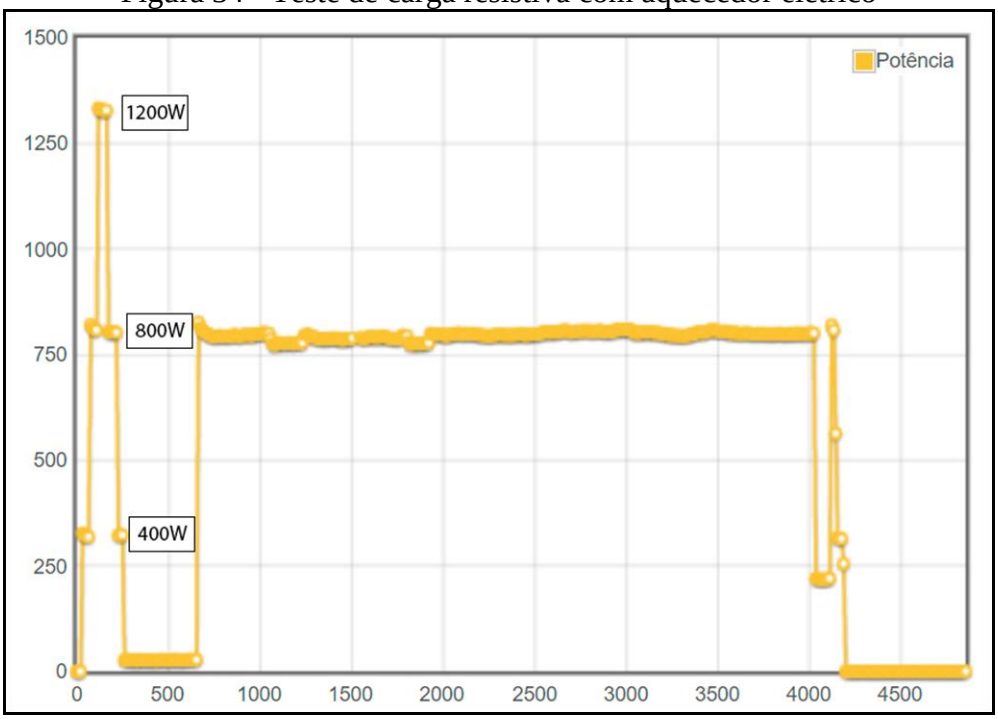
O aplicativo móvel, previsto em um dos objetivos específicos, não foi implementado. Como alternativa, a interface Web foi implementada de tal forma que é compatível com navegadores móveis, e o leiaute garante que os elementos são ajustados de acordo com as dimensões da tela. Esta interface foi testada nas versões móveis dos navegadores Chrome e Opera.

O requisito não-funcional RNF1 previa que o circuito do plugue não consumisse mais do que 1 Watt. Esse número não foi atingido, por conta do uso de uma fonte chaveada, ligada diretamente na alimentação da tomada. Uma alternativa poderia ter sido o uso de baterias combinado aos recursos de hibernação do ESP8266. Em especial, o modo *light sleep* permite desligar o modem Wi-Fi ao mesmo tempo em que a conexão é mantida aberta. Este modo não

incrementaria excessivamente o tempo de resposta do plugue para comandos imediatos, ao mesmo tempo que ofereceria um consumo reduzido por parte do plugue.

Em relação aos testes realizados, o resultado do teste com carga resistiva pode ser conferido na Figura 34. Pode-se notar no gráfico os momentos em que o equipamento foi ligado, desligado, e os períodos em que permaneceu operante. O aquecedor elétrico possui três estágios, de 400W, 800W e 1200W, respectivamente, que podem ser notados no primeiro segmento do gráfico. Este teste teve a duração aproximada de 75 minutos. Vale notar que a potência e corrente apresentadas nos gráficos a seguir não se tratam da potência efetiva, ou Root Mean Square (RMS), e a isso se deve a divergência em relação às leituras do alicate amperímetro, conforme pode ser visto na Tabela 1.

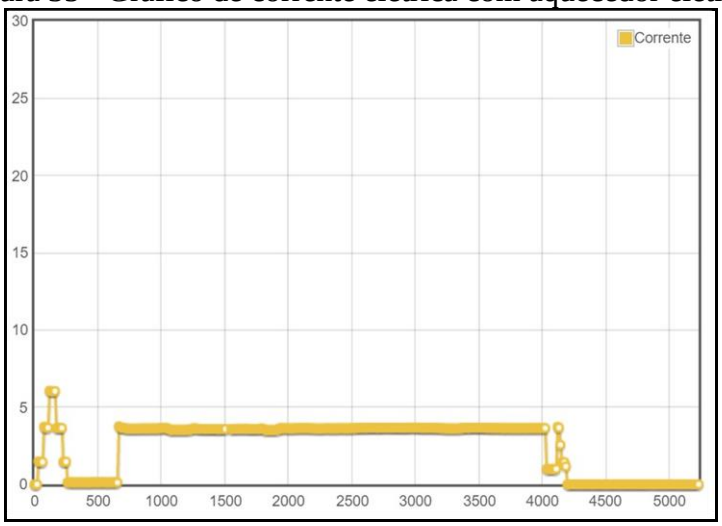
Figura 34 - Teste de carga resistiva com aquecedor elétrico



Fonte: elaborado pelo autor.

A Figura 35 apresenta a corrente elétrica medida durante o mesmo período. Durante a maior parte do teste, em que o aquecedor estava ligado no modo 800W, o alicate amperímetro estava marcando exatos 3 amperes RMS, o que, com uma tensão da rede de 220V resultaria em uma potência de 660W RMS .

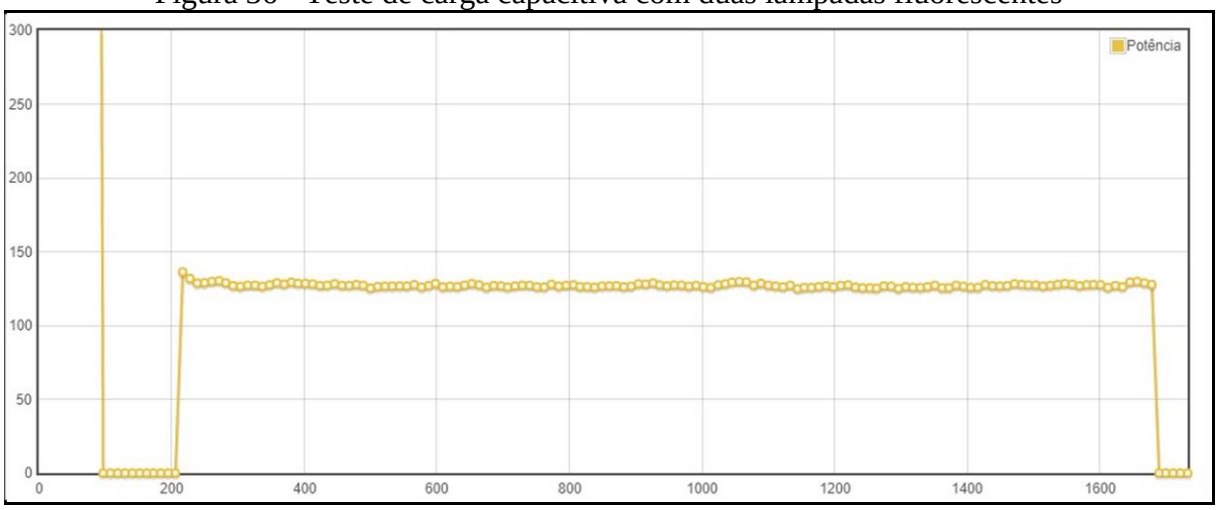
Figura 35 - Gráfico de corrente elétrica com aquecedor elétrico



Fonte: elaborado pelo autor.

Na Figura 36 é apresentado o resultado do teste com carga capacitiva, no qual foram empregadas duas lâmpadas fluorescentes de 59W cada uma. Observa-se um pico no início da leitura, quando o plugue foi ligado, seguido por um momento de estabilidade onde a leitura indica corretamente a ausência de consumo. As lâmpadas permaneceram ligadas por um período de 25 minutos. Ao longo desse período, pode-se notar que a leitura de consumo se manteve em aproximados 129W.

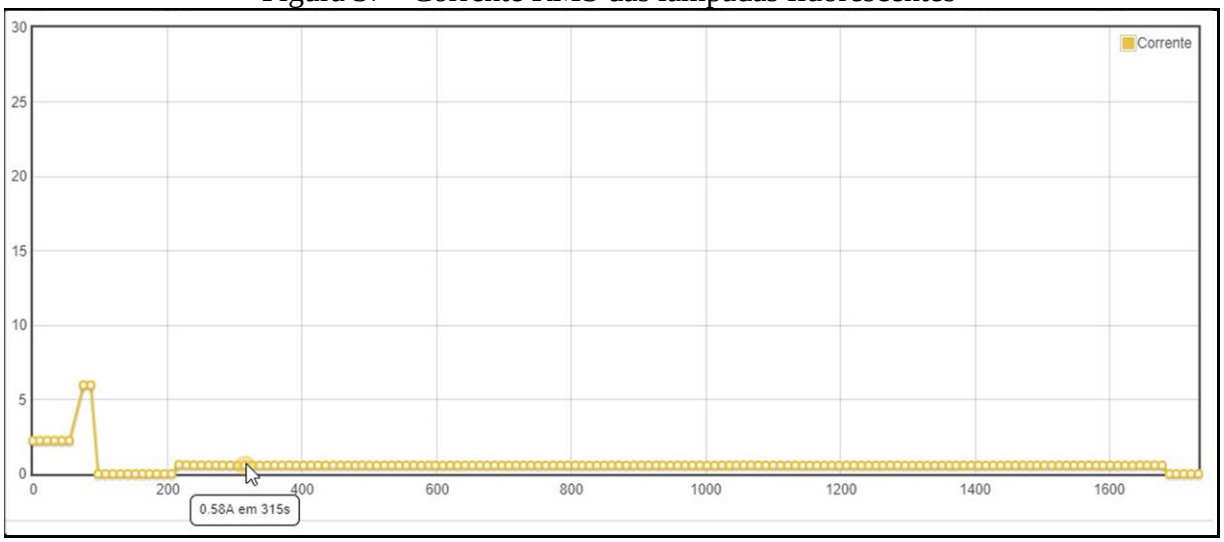
Figura 36 - Teste de carga capacitiva com duas lâmpadas fluorescentes



Fonte: elaborado pelo autor.

Na Figura 37 pode-se ver que a corrente indicada pela aplicação é de aproximadamente 0,58 amperes durante o período em que as lâmpadas permaneceram ligadas, enquanto o alicate amperímetro indica a corrente de 0,4 amperes RMS.

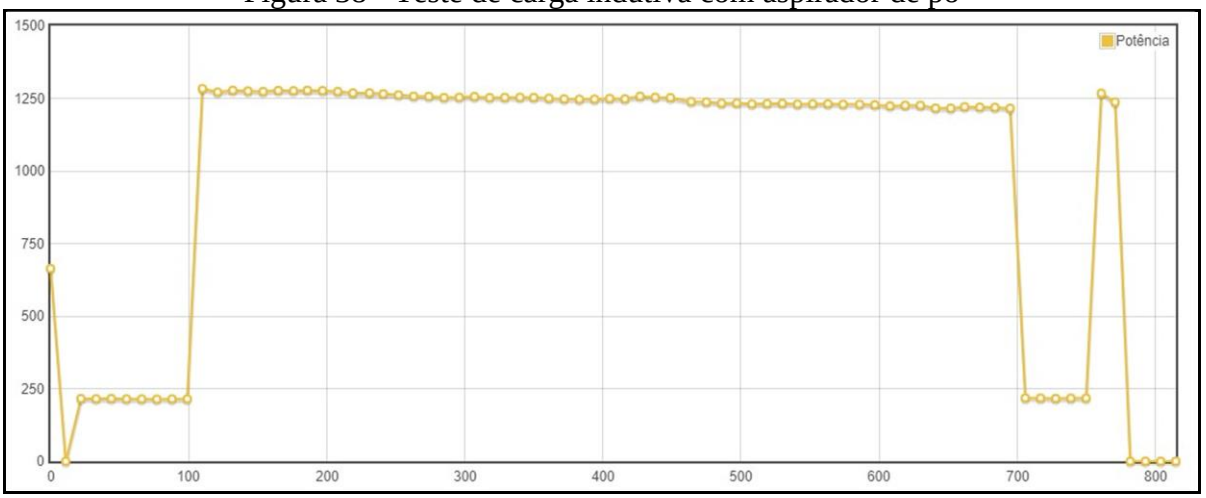
Figura 37 - Corrente RMS das lâmpadas fluorescentes



Fonte: elaborado pelo autor.

No gráfico da Figura 38 é apresentado o resultado do teste com carga indutiva, para o qual foi utilizado um aspirador de pó de 1400W, ligado pelo período de 10 minutos. Pode-se notar que enquanto o aparelho permaneceu ligado, a aplicação apontou consumo aproximado de 1250W. Além disso, foram observadas instabilidades na leitura durante os períodos de inatividade do aparelho, ao início e ao final do teste, nos quais a leitura foi diferente do zero esperado para estes casos, e que podem ser devidos a problemas de software.

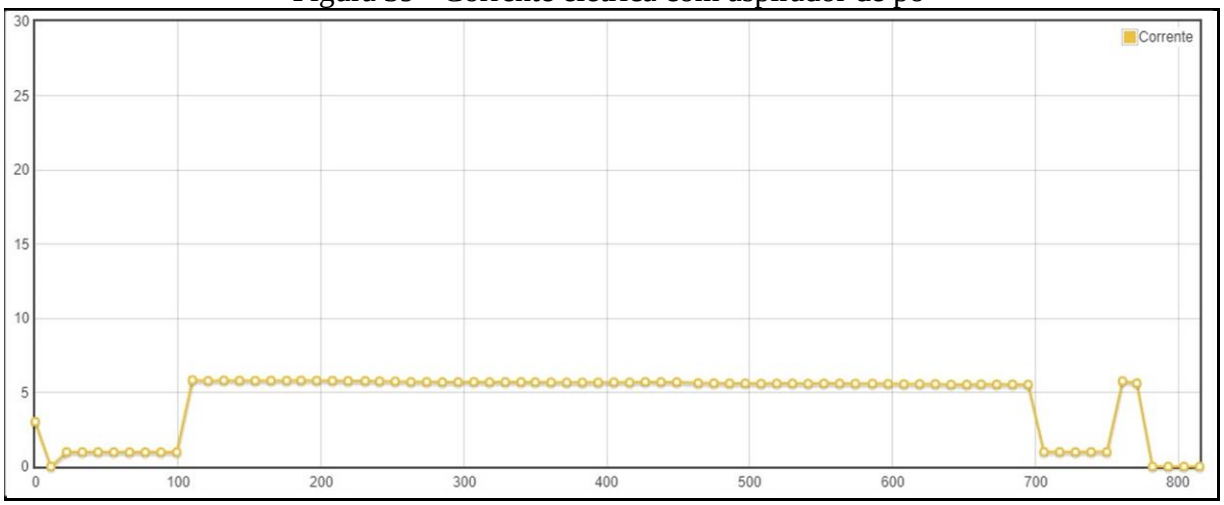
Figura 38 - Teste de carga indutiva com aspirador de pó



Fonte: elaborado pelo autor.

Conforme pode-se observar na Figura 39, a aplicação apresentou a corrente de aproximadamente 6 amperes, enquanto o alicate amperímetro acusou 4.1 amperes.

Figura 39 - Corrente elétrica com aspirador de pó



Fonte: elaborado pelo autor.

Dado que a potência e corrente mensuradas não estão em RMS, e que, segundo Braga (2017), a corrente RMS pode ser aproximada ao multiplicar-se a corrente instantânea por 0.707, a Tabela 1 foi composta a fim de facilitar a comparação dos resultados.

Tabela 1 - Comparação das leituras de corrente

Equipamento	Corrente lida	Corrente RMS aproximada	Corrente RMS lida pelo M266
Aquecedor elétrico	4,1A	2,89A	3A
Aspirador de pó	6A	4,24A	5,1A
Lâmpadas	0,58A	0,41A	0,4A

Fonte: elaborado pelo autor.

Conforme pode-se ver na Tabela 1, os resultados obtidos estão bastante próximos do esperado. Ainda assim, o uso de um sensor mais adequado ao uso residencial, como a variação do ACS712 que mede a faixa de -20A a 20A, pode melhorar a acuracidade das leituras. A qualidade da alimentação do sensor pode ser melhorada, pois notou-se que a fonte de empregada nem sempre é capaz de suprir a tensão exata de 5V que o ACS712 necessita para realizar uma leitura de qualidade livre de ruídos.

Por fim, no Quadro 8 são apresentados os resultados do teste de agendamentos consecutivos, contemplando a ação esperada, o horário esperado e o que foi observado. Como pode ser visto, todas as ações agendadas foram executadas conforme o esperado. Foram observados atrasados de aproximadamente um segundo em alguns dos agendamentos. Esse atraso se deve à latência de comunicação entre a aplicação Web, a aplicação Servidor, o *broker* público MQTT e o plugue.

Quadro 8 - Resultados dos testes com agendamento

Horário do agendamento	Ação	Intervalo	Horário estimado	Ação foi executada?	Horário
10:00:00	Ligar	30 segundos	10:00:30	Sim	10:00:30
10:00:05	Desligar	1 minuto	10:01:05	Sim	10:01:06
10:00:15	Ligar	90 segundos	10:01:45	Sim	10:01:45
10:00:30	Desligar	5 minutos	10:05:30	Sim	10:05:30
10:00:45	Ligar	1 hora	11:00:45	Sim	11:00:46

Fonte: elaborado pelo autor.

No Quadro 9 pode-se ver um comparativo das características mais relevantes entre o protótipo desenvolvido e os trabalhos correlatos.

Quadro 9 - Comparativo entre o protótipo e os trabalhos correlatos

Características mais relevantes	**Trabalhos correlatos**			**Maas (2017)**
	Sonoff Pow	**TP-Link HS110**	**Waka (2015)**	**Plugue Inteligente**
Portal cativo para informar credenciais de rede	Sim	Sim	Não	Não
Interface de usuário	Android/iOS	Android/iOS	Navegador Web	Navegador Web
Histórico de consumo	Sim	Não	Sim	Sim
Conexão Wi-Fi	Sim	Sim	Sim	Sim
Permite agendamento	Sim	Sim	Não	Sim
Histórico de atividades do plugue/tomada	Sim	Sim	Não	Sim
Sensor intrusivo	Sim	Não	Não	Não
Aplicação controla múltiplos dispositivos	Sim	Sim	Não	Sim
Custo	US$ 10,50 (R$35,00)	US$ 39,99 (R$ 133,00)	R$ 443,90 (protótipo)	R$ 148,39 (protótipo)

Fonte: elaborado pelo autor.

Considerando as informações disponíveis no Quadro 9 pode-se verificar que o Sonoff Pow é o mais completo em recursos, tendo como único contraponto o fato de necessitar cortar o fio elétrico do equipamento ou de uma extensão para realizar a instalação. Além disso, é a opção mais barata dentre os trabalhos analisados. O TP-Link HS110 é uma opção não invasiva que apresenta um preço relativamente alto, especialmente em relação ao Sonoff Pow. Sua maior desvantagem no mercado nacional é que ele não está disponível no padrão de tomadas brasileiro, e requer a utilização de adaptadores. Ambos permitem agendar ações execução posterior e disponibilizam aplicativos móveis. O protótipo de Waka (2015), por sua vez, não permitia o agendamento de ações, mas disponibiliza um relatório de consumo e é controlável através do navegador. O protótipo desenvolvido também apresentou custo menor do que o de Waka (2015), totalizando o valor de R$ 148,39, aproximando-se do valor do modelo comercial TP-Link HS110. Considerando-se que este valor pode ser reduzido através da importação de componentes da China, o protótipo pode vir a ser uma alternativa viável até mesmo em relação a alguns produtos como o TP-Link HS110.

O protótipo desenvolvido se mostra interessante ao unir as funcionalidades de controle de equipamentos em caráter imediato ou agendado, além do monitoramento de consumo, em um desenho não intrusivo. Além disso, o protótipo possui como diferencial a possibilidade de consultar o estado lógico do plugue em qualquer momento, permitindo ao usuário consultar todo o histórico de ações executadas.

4 CONCLUSÕES

Este trabalho apresentou o desenvolvimento de um protótipo de plugue para tomada elétrica que pode ser controlado remotamente. Os objetivos incluíam o desenvolvimento de um aplicativo móvel a partir do qual o usuário poderia controlar o plugue e visualizar um relatório de consumo dos equipamentos conectados a ele.

Dentre os objetivos do trabalho, um deles foi cumprido parcialmente. No lugar do aplicativo móvel previsto originalmente, foi desenvolvido um portal Web acessível tanto em navegadores para dispositivos *desktop* quanto móveis. Os relatórios, no entanto, não foram otimizados para dispositivos móveis.

Em relação ao hardware, a escolha pelo NodeMCU se mostrou uma decisão acertada, suprindo todos os requisitos para o projeto sem grandes surpresas ao longo do desenvolvimento, e o sensor de corrente utilizado apresentou resultados satisfatórios.

A maior parte do desenvolvimento do *firmware* foi realizada na IDE Visual Studio com o *plugin* Visual Micro, o que facilitou muito o desenvolvimento na linguagem C++ em relação à IDE do Arduino. No entanto, houve problemas ao descarregar algumas versões do *firmware* que já faziam uso do Wi-Fi, nas quais o software descarregado para o módulo não conectava à rede Wi-Fi estabelecida. A partir de então, foi utilizada a IDE do Arduino para o desenvolvimento do *firmware*.

Quanto ao software, o *microframework* NancyFX mostrou-se robusto o suficiente para o projeto, permitindo a configuração facilitada de *endpoints* REST em uma aplicação auto hospedada. A biblioteca Marten também se mostrou muito versátil e atendeu as expectativas, possibilitando a aferição do histórico do plugue.

Por fim, este trabalho mostra-se relevante por implementar um plugue não invasivo capaz de controlar equipamentos remotamente e oferecer o monitoramento do consumo de energia elétrica desses equipamentos. Outra contribuição do trabalho foi o uso do ESP8266 e do padrão Event Sourcing para aferição do histórico lógico do plugue. Por último, vale ressaltar que o plugue é compatível com o padrão nacional de tomadas, excluindo a necessidade de adaptadores, como ocorre com plugues importados, e que apresentou custo próximo do TP-Link HS110.

4.1 EXTENSÕES

O protótipo desenvolvido ao longo deste trabalho atingiu seu objetivo principal. No entanto, existem vários pontos passíveis de melhoria e novas funcionalidades que podem acrescentar benefícios para os usuários do plugue. São eles:

a) adicionar um portal cativo que permita ao usuário informar as credenciais da rede WiFi em que o plugue será utilizado;
b) empregar técnicas de aprendizado de máquina para conferir comportamentos inteligentes ao plugue, permitindo que se adapte à rotina do usuário;
c) acrescentar novas opções de agendamento de ações, como a possibilidade de agendar uma ação para determinados dias da semana;
d) persistir os agendamentos na memória EPROM do ESP8266;
e) reimplementar a interface de usuário para dispositivos móveis, como um Progressive Web App, ou aplicativo nativo;
f) reduzir o consumo de energia do plugue, implementando estratégias de hibernação do ESP8266;
g) utilizar a variante do sensor de corrente que efetua leituras na faixa de -20A a 20A, visando aumentar a sensibilidade das leituras;
h) utilizar um sensor de tensão, para aumentar a precisão do cálculo de consumo;
i) implementar recursos de segurança na comunicação com o *broker* MQTT através do uso de certificados digitais;
j) criar um HUB para centralizar o controle de múltiplos plugues;
k) implementar uma interface de hardware para conectar módulos de sensores ao plugue, de modo que possam conferir novos comportamentos condicionados por dados sensoriais;
l) utilizar um plugue Sonoff como plataforma de hardware, modificando seu *firmware* para adicionar novas funcionalidades.

REFERÊNCIAS

ABESCO. Brasil desperdiça R$ 12 bi em 5 anos. **Abesco**, 2015. Disponivel em: <http://www.abesco.com.br/pt/novidade/brasil-desperdica-r-12-bi-em-5-anos>. Acesso em: 25 mar. 2017.

ABREU, H. Í. D. Eficiência Energética de Equipamentos Elétricos Residenciais. **Faip**, Marília, maio 2015. 1.

ADDICORE. ESP8266 ESP-12Q (ESP-12E) WiFi Module. **Addicore**, 2017. Disponivel em: <https://www.addicore.com/ESP8266-ESP-12-p/ad247.htm>. Acesso em: 29 ago. 2017.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 14136: informação e documentação: Plugues e tomadas para uso doméstico e análogo até 20 A/250 V em corrente alternada - Padronização**. Rio de Janeiro. 2002.

AVSYSTEM. OMA LwM2M - Brief description. **AVSystem**, 2017. Disponivel em: <https://avsystem.github.io/Anjay-doc/LwM2M.html>. Acesso em: 3 set. 2017.

BRAGA, N. C. O que é corrente alternada. **Instituto NCB**, 2017. Disponivel em: <http://www.newtoncbraga.com.br/index.php/cursos-on-line/93-cursos/curso-de-eletronica/2729-cbe007>. Acesso em: 29 nov. 2017.

CISCO. Cisco Visual Networking Index Predicts Near-Tripling of IP Traffic by 2020. **Cisco**, 2016. Disponivel em: <https://newsroom.cisco.com/press-release-content?type=press-release&articleId=1771211>. Acesso em: 23 nov. 2017.

COAP. **COAP**, 2014. Disponivel em: <http://coap.technology>. Acesso em: 23 maio 2017.

DANTAS, B. F. **Estimativa do impacto no consumo de energia causado pelo standby dos aparelhos eletroeletrônicos**. 92 f. Dissertação (Mestrado em Metrologia) - Programa de Pós-Graduação em Metrologia (Área de Concentração: Metrologia para Qualidade e Inovação) - Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro. 2014.

DAVIS, B. 10 examples of the Internet of Things in healthcare. **Econsultancy**, 2017. Disponivel em: <https://econsultancy.com/blog/68878-10-examples-of-the-internet-of-things-in-healthcare>. Acesso em: 23 nov. 2017.

ENERGIA SUSTENTÁVEL. **Energia Sustentável**, 2015. Disponivel em: <https://energiasustentavel.pt/saiba-como-economizar-energia-eletrica>. Acesso em: 23 maio 2017.

EPE. **Balanço Energético Nacional: Relatório Síntese, Ano Base 2015**. Brasília. 2016.

FILIPEFLOP. Módulo Relé 5V 2 Canais, 2017. Disponivel em: <https://www.filipeflop.com/produto/modulo-rele-5v-2-canais/>. Acesso em: 18 nov. 2017.

FILIPEFLOP. Sensor de Corrente ACS712 -30A a +30A, 2017. Disponivel em: <https://www.filipeflop.com/produto/sensor-de-corrente-acs712-30a-a-30a/>. Acesso em: 18 nov. 2017.

FOWLER, M. Event Sourcing. **Martin Fowler**, 2005. Disponivel em: <https://martinfowler.com/eaaDev/EventSourcing.html>. Acesso em: 19 nov. 2017.

FULLER, DANIEL. AH Tech Talk: IoT Fragmentation Is Reaching Critical Mass. **Android Headlines**, 2016. Disponivel em: <https://www.androidheadlines.com/2016/04/ah-tech-talk-iot-fragmentation-is-reaching-critical-mass.html>. Acesso em: 3 set. 2017.

GOOGLE PLAY. Kasa, 2017. Disponivel em: <https://play.google.com/store/apps/details?id=com.tplink.kasa_android>. Acesso em: 2017 mar. 2017.

GREENPEACE. **Relatório de Hidrelétricas na Amazônia: Um Mau Negócio Para o Brasil e Para o Mundo**. São Paulo, p. 10-12. 2016.

INTEL Galileo Board. **Intel**, 2015. Disponivel em: <http://www.intel.com/content/www/us/en/support/boards-and-kits/intel-galileo-boards/intel-galileo-board.html>. Acesso em: 29 mar. 2017.

I-SCOOP. Internet of Things – the complete online guide to the IoT. **I-Scoop**, 2016. Disponivel em: <https://www.i-scoop.eu/internet-of-things-guide/#What_is_the_Internet_Of_Things>. Acesso em: 4 set. 2017.

ITEAD. Sonoff Pow WiFi Switch With Power Consumption Measurement. **Itead**, 2015. Disponivel em: <https://www.itead.cc/sonoff-pow.html>. Acesso em: 2017 maio 2017.

JAFFEY, T. MQTT and CoAP, IoT Protocols. **Eclipse**, 2014. Disponivel em: <https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php>. Acesso em: 3 abr. 2017.

KOLBAN, N. **Kolban's Book on the EPS32 & ESP8266**. Victoria: LeanPub, 2016. Disponivel em: <https://leanpub.com/ESP8266_ESP32>. Acesso em: 25 mar. 2017.

LEVY, M.; WALLIS, M. The challenges of microcontrollers living on the edge (of the IoT). **Embedded Computing**, Fountain Hills, 24 out. 2014. Disponivel em: <http://embedded-computing.com/articles/the-challenges-of-microcontrollers-living-on-the-edge-of-the-iot/>. Acesso em: 3 abr. 2017.

MICROSOFT. Event Sourcing pattern. **Microsoft Azure**, 2017. Disponivel em: <https://docs.microsoft.com/en-us/azure/architecture/patterns/event-sourcing>. Acesso em: 19 nov. 2017.

MINDTEK. 6LoWPAN - Techinal Overview. **Mindtek**, 2009. Disponivel em: <http://ms11.voip.edu.tw/~jryan/ref/6LoWPAN%20Technical%20Overview.pdf>. Acesso em: 7 set. 2017.

MORENO, V. et al. How can We Tackle Energy Efficiency in IoT Based Smart Buildings? **Sensors**, Murcia, Espanha, 30 maio 2014. 9582-9614.

MQTT. **MQTT**, 2017. Disponivel em: <http://mqtt.org>. Acesso em: 23 maio 2017.

NODEMCU. NodeMCU DEVKIT V1.0. **NodeMCU**, 2017. Disponivel em: <https://github.com/nodemcu/nodemcu-devkit-v1.0>. Acesso em: 4 set. 2017.

OASIS. OASIS Standards. **OASIS**, 2017. Disponivel em: <https://www.oasis-open.org/standards>. Acesso em: 23 maio 2017.

OPEN MOBILE ALLIANCE, 2016. Disponivel em: <http://openmobilealliance.org/in-the-news/understanding-lightweight-m2m-protocol-and-its-benefits>. Acesso em: 3 set. 2017.

OPENHAB. **OpenHAB**, 2017. Disponivel em: <https://www.openhab.org>. Acesso em: 25 mar. 2017.

SHELBY, Z. Smart Cities are the Internet of Things. **SlideShare**, Helsinki, 20 jun. 2013. Disponivel em: <https://pt.slideshare.net/zdshelby/smart-cities-are-the-internet-of-things>. Acesso em: 15 set. 2017.

SHELBY, Z.; BORMANN, C. 6LoWPAN: The wireless embedded Internet - Part 1: Why 6LoWPAN? **EETimes**, 2011. Disponivel em: <http://www.eetimes.com/document.asp?doc_id=1278794>. Acesso em: 7 set. 2017.

TEXAS INSTRUMENTS. Overview for 6LoWPAN. **Texas Instruments**. Disponivel em: <http://www.ti.com/lsds/ti/wireless-connectivity/6lowpan/overview.page>. Acesso em: 7 set. 2017.

TP-LINK. Wi-Fi Smart Plug with Energy Monitoring. **TP-Link**, 2015. Disponivel em: <http://www.tp-link.com/us/products/details/cat-5516_HS110.html>. Acesso em: 26 mar. 2017.

TRACY, P. Understanding Lightweight M2M protocol and its benefits. **RCR Wireless News**, 2016. Disponivel em: <https://www.rcrwireless.com/20161212/internet-of-things/lwm2m-tag31-tag99>. Acesso em: 3 set. 2017.

WAKA, G. M. **Controle remoto de tomadas elétricas baseado nos conceitos da Internet das coisas**. Trabalho de Conclusão de Curso - Universidade Federal do Rio Grande do Sul. Porto Alegre. 2015.

ZHU, Q. et al. **IOT Gateway:** BridgingWireless Sensor Networks into Internet of Things. Embedded and Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on. Hong Kong, China: IEEE. 2010. p. 347-352.

APÊNDICE A – Relação de preço dos componentes utilizados

A Tabela 2 apresenta todos os componentes utilizados na elaboração do protótipo. Todos os componentes foram adquiridos em lojas nacionais, em especial na FilipeFlop.

Tabela 2 - Relação de componentes

Componente	Valor Unitário	Quantidade	Valor Total
Módulo WiFi ESP8266 NodeMcu ESP-12	R$ 49,90	1	R$ 49,90
Sensor de Corrente ACS712 -30A a +30A	R$ 22,90	1	R$ 22,90
Módulo Relé 5V 2 Canais	R$ 12,90	1	R$ 12,90
LED Difuso 5mm Verde	R$ 0,19	1	R$ 0,19
Potenciômetro 100K	R$ 1,90	1	R$ 1,90
Diodo 1N4004	R$ 0,14	1	R$ 0,14
Resistor 100 Ohm	R$ 0,09	1	R$ 0,09
Protoboard 400 Pontos	R$ 14,90	1	R$ 14,90
Fonte DC chaveada 5V 2A Plug P4	R$ 39,90	1	R$ 39,90
Plugue fêmea 2P+T 10A	R$ 3,00	1	R$ 3,00
Plugue macho 2P+T 10A	R$ 2,49	1	R$ 2,49
		TOTAL	**R$ 148,31**

Fonte: elaborado pelo autor.