UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Bruno Moreira Barbosa

Dispositivo não invasivo de monitoramento de energia elétrica

São Carlos

2020

Bruno Moreira Barbosa

Dispositivo não invasivo de monitoramento de energia elétrica

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Sistemas de Energia e Automação, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Maximiliam Luppe

São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Barbosa, Bruno Moreira
B238d Dispositivo não invasivo de monitoramento de energia elétrica / Bruno Moreira Barbosa; orientador Maximiliam Luppe. São Carlos, 2020.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2020.

1. Monitor de Energia. 2. ESP32. 3. Internet das Coisas. 4. AWS. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Bruno Moreira Barbosa

Título: "Dispositivo não invasivo de monitoramento de energia elétrica"

Trabalho de Conclusão de Curso defendido e aprovado em 23/06/2020,

com NOTA 9,0 (nove, zero), pela Comissão Julgadora:

Prof. Dr. Maximiliam Luppe - Orientador - SEL/EESC/USP

Prof. Associado Dennis Brandão - SEL/EESC/USP

Prof. Associado Evandro Luis Linhari Rodrigues - SEL/EESC/USP - Aposentado

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Este trabalho é dedicado a todos os pesquisadores do Brasil, que estão sendo e serão fundamentais na superação da crise que vivemos hoje devido à pandemia do COVID-19.

AGRADECIMENTOS

Agradeço, em primeiro lugar, aos meus pais, José e Delian, por serem meus alicerces e por todos os sacrifícios que me garantiram o acesso a uma educação de qualidade.

Às minhas irmãs, Camila e Larissa, pelo companherismo em todos os momentos da minha vida.

À minha família, por ter estado ao meu lado nas situações mais difíceis.

À minha namorada, Giovanna, por todo o amor, cuidado e apoio durante a minha graduação, inclusive na produção deste trabalho.

Ao professor Maximilian Luppe, pela orientação e mentoria durante o desenvolvimento desse projeto.

E aos meus amigos, Alceu, André, Bruno, Gabriel, Guilherme e Nelson, pela amizade que criamos na graduação e levaremos para toda a vida.

"A ciência é uma disposição de aceitar os fatos mesmo quando eles são opostos aos desejos."

Burrhus Frederic Skinner

RESUMO

BARBOSA, B. M. **Dispositivo não invasivo de monitoramento de energia elétrica**. 2020. 69p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

Este trabalho descreve o desenvolvimento de uma placa de sensoriamento microcontrolada para realizar as medições de corrente, potência e tensão de uma residência. A partir da utilização dos conceitos de IoT, os dados do medidor serão enviados a um servidor em nuvem. Essas informações devem ser acessadas por outro servidor, que os armazenará e os disponibilizará em uma interface gráfica e interativa para o consumidor final. Foi elaborado um *software* de sensoriamento aplicando a placa de desenvolvimento NodeMCU-ESP32S, que se conectou a um *broker* MQTT que foi desenvolvido utilizando o serviço AWS IoT Core. Além disso, foram desenvolvido dois *softwares* em Python responsáveis por acessar o *broker*, coletar os dados e apresentá-los em um dashboard em tempo real. Por fim, uma placa protótipo foi confeccionada, com a qual foram realizados testes para testar a funcionalidade do conceito quando instalado em uma residência.

Palavras-chave: Monitor de energia. ESP32. Internet das Coisas. AWS

ABSTRACT

BARBOSA, B. M. **Dispositivo não invasivo de monitoramento de energia elétrica**. 2020. 69p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2020.

This work describes the development of a microcontroller sensing board to perform current, power, and voltage measurements of a home. From using IoT concepts, the meter data will be sent to a server at a cloud. This information must be accessed by another server, which will store it and will be available in a graphical and interactive interface for the final consumer. It was formulated a sensing software using the NodeMCU-ESP32S development board, that was connected to an MQTT broker that was developed using AWS IoT Core service. Moreover, it was produced two software in Python responsible for accessing the broker, collect the data, and present them on a dashboard in real-time. Lastly, a prototype plate was made, with which tests were performed to test the concept functionality when installed at a residence.

Keywords: Energy monitor. ESP32. Internet of Things. AWS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	*Anologic to Digital Converter*
API	*Application Programming Interface*
CPU	*Central Processing Unit*
ESP-IDF	*Espressif IoT Development Framework*
GPIO	*General Purpose Input/Output*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IP	*Internet Protocol*
OS	*Operating System*
PWM	*Pulse Width Modulation*
QoS	*Quality of Service*
RTOS	*Real Time Operating System*
SQL	*Structured Query Language*
SoC	*System on a Chip*
TCP	*Transmission Control Protocol*

SUMÁRIO

1 INTRODUÇÃO

A energia elétrica, apesar de recente na história da humanidade, se apresenta hoje como um recurso fundamental para a dinâmica mundial. Dentre todos os aspectos da vida em sociedade, desde o bombeamento de águas para as cidades até as mais comuns formas de entretenimento, a eletricidade se mostra presente em praticamente todos os momentos. E isso não mais se restringe apenas aos países desenvolvidos: estima-se que 87% da população mundial tenha acesso a energia elétrica (RITCHIE; ROSER, 2020).

Nesse mesmo contexto, o advento da eletricidade proporcionou uma onda de desenvolvimento tecnológico que culminou em inúmeras inovações, dentre as quais se destaca a Internet das coisas (IoT, do inglês *Internet of Things*). Trata-se de um conceito muito amplo, mas é possível definir IoT como o que é obtido quando são conectados dispositivos (ou coisas), que não são operados por seres humanos, à Internet (WAHER, 2015). E uma quantidade crescente de dipositivos estão sendo lançados no mercado com a funcionalidade de serem conectados à internet, desde sensores até eletrodomésticos. Uma pesquisa recente projetou que, até 2025, haverá 41,6 bilhões de dispositivos IoT conectados à Internet, generando 79,4 zetabytes (ZB) de dados (FRAMINGHAM, 2019).

Apesar da grande difusão da energia elétrica, vale ressaltar que este não é um recurso infinito, e depende de fontes não-renováveis e renováveis para ser gerado. Sendo assim, para a maior parte do mundo, o acesso a este recurso está limitado à capacidade produtiva do país e ao custo para o consumidor final. Nesse contexto, dispositivos medidores de energia podem se mostrar úteis para que os consumidores controlem o quanto estão consumindo e, consequentemente, o quanto estão pagando.

Ainda que as residências necessariamente já possuam medidores eletromecânicos de energia para o controle das concessionárias, medidores eletrônicos com interfaces digitais são mais úteis aos consumidores, por permitirem não só um fácil acompanhamento em tempo real, como também a discriminação do consumo por circuitos dentro da casa. Conquanto se disponibilizarem recursos de aferição individual dessas máquinas e equipamentos, estar-se-á oportunizando melhorias nos códigos de comportamento do consumidor, enriquecendo uma matriz que antes primava somente por eficiência, agregando as preocupações com qualidade do consumo.

1.1 Motivação

A maioria dos medidores digitais de consumo de energia elétrica possuem um preço elevado e devem ser instalados em série com os circuitos da casa, o que torna mais complicado e oneroso para residências com a estrutura elétrica já concluída. Sendo assim,

faz-se conveniente o desenvolvimento de um medidor de baixo custo que possa ser instalado em qualquer caixa de distribuição de energia. Além disso, coadunando baixo custo e maiores vantagens para o consumidor, a utilização dos conceitos de IoT torna possível que o medidor conecte-se a Internet para disponibilizar tais medições em tempo real em uma interface interativa e amigável.

1.2 Objetivos

O presente trabalho se propõe a desenvolver um protótipo que consiste em uma placa de sensoriamento microcontrolada para realizar as medições de corrente, potência e tensão de uma residência, e enviar esses dados a um servidor em nuvem. Estes dados devem ser acessados por outro servidor, que os armazenará e os disponibilizará em uma interface gráfica e interativa para o consumidor final.

1.3 Organização do trabalho

O trabalho descrito nesse documento foi organizado da seguinte forma:

- **Embasamento teórico**: conceitos fundamentais para o entendimento do que foi desenvolvido e tecnologias utilizadas, bem como cálculos que serão aplicados
- **Materiais e métodos**: módulos e estruturas do *hardware* desenvolvido e detalhamento dos *softwares* disponíveis
- **Resultados e discussão**: resultado da implementação do protótipo e discussão dos testes
- **Conclusão**: comentários finais considerando os objetivos do trabalho e o que foi atingido

2 EMBASAMENTO TEÓRICO

Nesse capítulo, serão apresentados os conhecimentos fundamentais que permitiram a elaboração do projeto, elucidando os tipos de sensoriamento utilizados, os aspectos de *hardware* e *software* do sistema embarcado, e os conceitos de conectividade empregados.

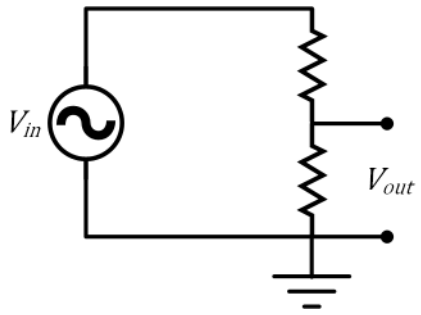
2.1 Sensoriamento

Um sensor pode ser definido como um dispositivo capaz de receber e responder a um estímulo ou sinal e, mais especificamente, um sensor eletrônico é um tipo de sensor que responde ao estímulo com uma corrente elétrica (FRADEN, 2003). Neste trabalho, a escolha e a calibragem dos sensores é de grande importância para a qualidade dos dados obtidos.

2.1.1 Medição de tensão alternada

Para se medir tensões alternadas, podem ser utilizados métodos diretos ou métodos indiretos (FERRERO, 2014), a depender principalmente dos níveis de tensão a serem medidos. Um método simples para realizar as medições é o uso de um divisor de tensão, em que o sinal de entrada passa por dois resistores responsáveis por reduzir o nível de tensão a um valor adequado para a leitura, conforme Figura 1. No entanto, esse método torna-se inadequado quando a tensão de entrada apresenta uma magnitude muito elevada, como é o caso da tensão de rede.

Figura 1 – Divisor de tensão para medir tensão alternada

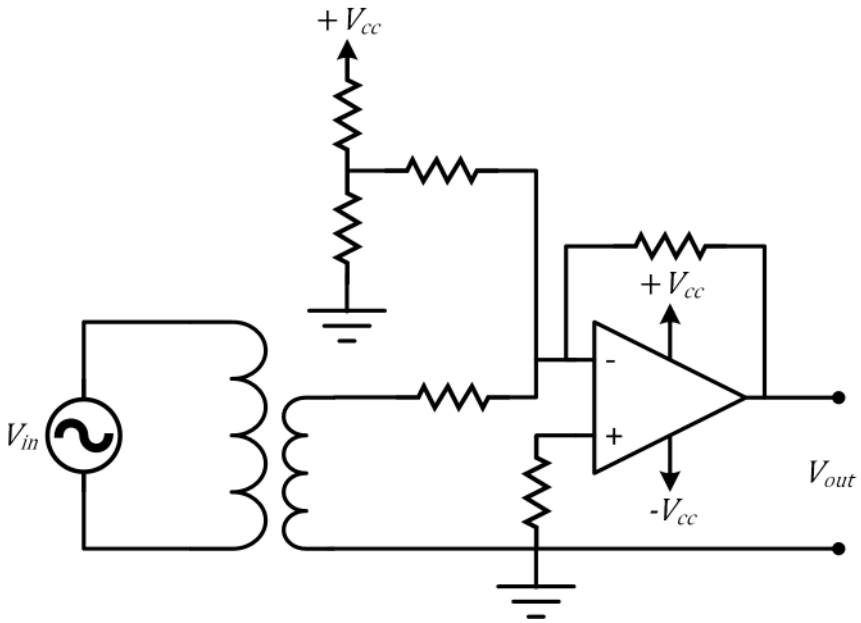


Fonte: o próprio autor

Para estes casos, são utilizados os métodos indiretos como, por exemplo, atenuar o sinal com o uso de um transformador de tensão e depois ajustar sua magnitude com o uso de um amplificador operacional, como na Figura 2. Além de garantir o nível de tensão desejado, esse método garante o isolamento galvânico entre a rede elétrica e a placa

de sensoriamento, tornando-a mais segura e com danos menos severos em caso de curto circuito.

Figura 2 – Transformador e amplificador operacional para medir tensão alternada

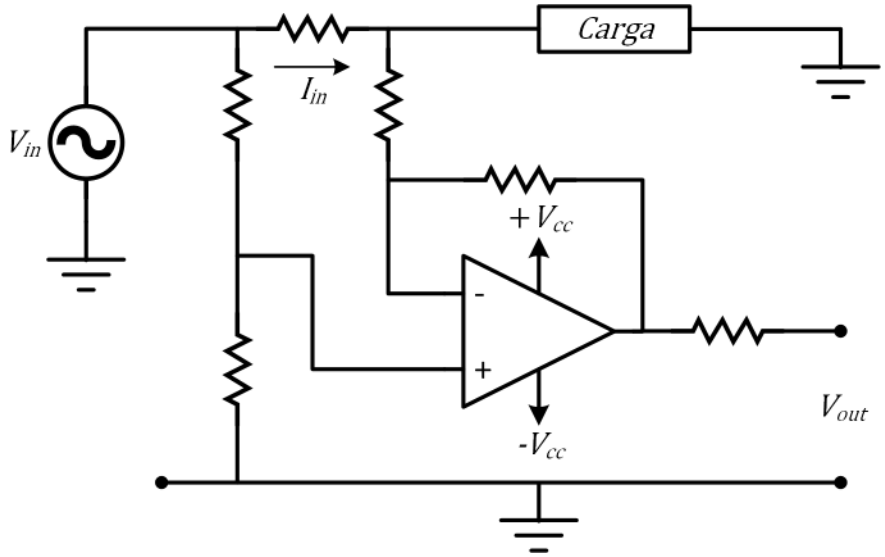


Fonte: o próprio autor

2.1.2 Medição de corrente alternada

Assim como as tensões, as correntes alternadas também podem ser mensuradas através de métodos diretos ou indiretos (MCNUTT, 2014). De maneira similar, é possível medir a corrente através do esquema da Figura 3, em que a corrente é obrigada a atravessar um resistor *shunt* e a queda de tensão resultante é amplificada através de um amplificador operacional, resultando em uma tensão proporcional à corrente. Esse método, no entanto, impõe que o trecho do circuito onde se deseja medir a corrente seja interrompido para instalar o sensor em série, o que nem sempre é viável. Além disso, são necessários resistores de potência para medir correntes elevadas.

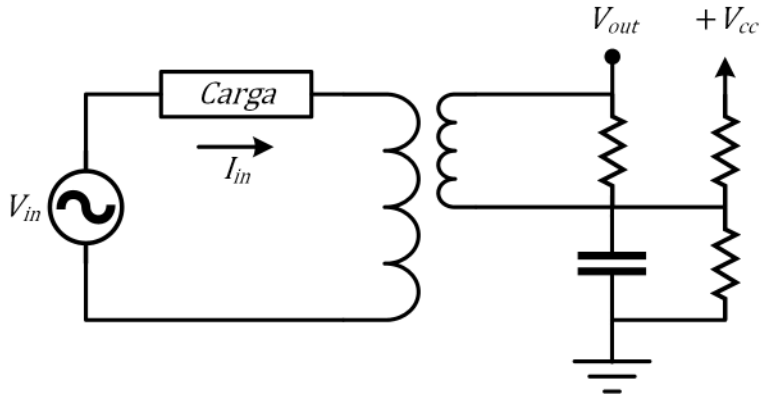
Figura 3 – Resistor *shunt* e amplificador operacional para medir corrente alternada



Fonte: o próprio autor

Assim, podem ser utilizados métodos indiretos, em que a corrente é atenuada através de um transformador de corrente, com o secundário conectado a um resistor para associar a corrente a um nível de tensão proporcional que pode ser lido por um microcontrolador, conforme Figura 4. Novamente, esse método permite o isolamento galvânico entre a corrente da rede medida e a tensão lida pela placa, e também garante que a corrente possa ser medida através de um dispositivo não-invasivo.

Figura 4 – Transformador de corrente para medir corrente alternada



Fonte: o próprio autor

2.1.3 Cálculo de potência

Os valores de corrente e tensão permitem o cálculo da potência instantânea P_i e a potência média P_{av} em um período T, conforme equações abaixo:

$$p(t) = V(t)I(t) \tag{2.1}$$

$$P_{av} = \frac{1}{T}\int_0^T p(t)dt \tag{2.2}$$

No entanto, a amostragem de um microcontrolador é discreta, logo a potência média referente a N pontos medidos pode ser através de uma integração númerica:

$$P_N = \frac{1}{N-1}\sum_{i=1}^{N}\frac{V[i]I[i] + V[i-1]I[i-1]}{2} \tag{2.3}$$

Já o valor eficaz, ou valor médio quadrático, de um sinal contínuo $i(t)$ pode ser calculado consoante à fórmula:

$$I_{RMS} = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt} \tag{2.4}$$

Tratando-se de uma amostragem discreta, o valor eficaz de um sinal de N pontos pode ser calculado como:

$$I_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} i^2[i]} \tag{2.5}$$

2.2 Sistemas embarcados

Um sistema embarcado pode ser definido como um sistema microprocessado construído especificamente para executar um conjunto de funções, mas que não é projetado para ser programado pelo usuário final (HEATH, 2002). O trabalho do projetista, portanto, é o de entender os requisitos de projeto, escolher um microcontrolador que permita essas funcionalidades e escrever um *software* sob medida que aproveite todos os recursos necessários.

2.2.1 Microcontroladores

Dentro da teoria de circuitos digitais, a unidade de processamento central (CPU, do inglês *central processing unit*) é um circuito capaz de realizar operações aritméticas básicas (como soma e subtração), operações lógicas e operações de controle, ou seja, é capaz de processar dados. Uma CPU construída em um único chip de silício pode ser definida como microprocessador (MALVINO; BROWN, 1993).

Quando um microprocessador é associado a outros elementos digitais, como memórias, *timers* e periféricos de entrada e saída, o dispositivo passa a ser denominado microcontrolador (TOCCI; WIDMER; MOSS, 2017). As memórias podem ser classificadas como não-voláteis, as quais não são perdidas quando o microcontrolador é desligado e

contém o conjunto de instruções que devem ser realizadas pelo microcontrolador; e voláteis, as quais são apagadas quando não há fonte de energia e armazenam os dados que são gerados durante a execução do programa. A capacidade das memórias deve ser levado em consideração durante o projeto, pois limita, por exemplo, o tamanho do programa a ser gravado no microcontrolador e a quantidade e o tamanho das variáveis que serão criadas durante a execução do programa.

Uma característica importante de um microcontrolador é a frequência de *clock* f_c, a unidade mínima de tempo que o microcontrolador levaria para realizar uma instrução e que serve para garantir o sincronismo entre as operações de seus componentes (WHITE, 2011). Esses pulsos de *clock* são utilizados pelos *timers* que, a cada d (divisor de frequência) ciclos de *clock*, incrementam um registrador, até atingir o valor máximo programado N. Quando esse valor é atingido, o programa realiza alguma instrução como, por exemplo, entrar em uma rotina de interrupção e reiniciar a contagem do *timer*. Os *timers* permitem, portanto, criar bases temporais de muita precisão para controlar eventos no programa, que são dadas pela fórmula:

$$t = \frac{d\ N}{f_c} \tag{2.6}$$

Cada microcontrolador possui um determinado número de *timers*, cada um deles com uma certa resolução n_t, que representa qual o número máximo de ciclos que ele pode contar. Por exemplo, um *timer* de 10bits pode contar um total 2^{10} ciclos. Da mesma forma, a resolução n_p do divisor de frequência do timer representa qual o maior número de ciclos de *clock* que podem ocorrer para que o *timer* incremente seu registrador em uma unidade.

Outros dois periféricos de grande importância para este trabalho são as entradas e saídas de propósito geral (GPIO, do inglês *General Purpose Input/Output*) e os conversores analógico-digitais (ADC, do inglês *Anologic to Digital Converter*). As GPIOs são portas do microcontrolador que podem ser utilizadas como entradas ou saídas para, respectivamente, ler ou escrever um estado digital. Elas podem também ser utilizadas para gerar sinais específicos, como a modulação por largura de pulso (PWM, do inglês *Pulse Width Modulation*), ou para servirem de interface para protocolos de comunicação, como CAN, I2C, serial, etc.

Já o ADC é um conversor capaz de tomar como entrada uma tensão analógica e transformar esse valor em uma grandeza digital proporcional à medida de entrada. A resolução de um ADC é de grande importância, porque indica qual é o menor valor de tensão incremental que pode ser reconhecido pelo conversor e, portanto, alterar a leitura digital (FREESCALE SEMICONDUCTOR, 2016). Desse modo, um ADC com uma tensão de referência de $V_{ref} = 5V$ e uma resolução de $n = 12bits$ consegue ler valores da ordem de:

$$v_i = \frac{V_{ref}}{2^n} \rightarrow v_i = \frac{5}{2^{12}} = 1,22\ mV \tag{2.7}$$

2.2.2 ESP32

ESP32 se refere a uma família de microcontroladores embarcados em chips (SoC, do inglês *System on a Chip*) integrados com Wi-Fi e Bluetooth desenvolvidos pela fabricante chinesa Espressif Systems, que tem como grande vantagem o seu baixo custo e o seu baixo consumo de energia (ESPRESSIF, 2019c). Nestes chips, são empregados microprorcessadores Tensilica Xtensa LX6 com um ou dois núcleos de processamento. Tais chips também são comercializados na forma de módulos, nos quais o chip, juntamente com uma memória flash e componentes diversos, são soldados em uma placa com roteamento para as portas do microcontrolador e com uma antena impressa. As Figuras 5 e 6 a seguir apresentam, respectivamente, um chip ESP32 e um módulo que o utiliza.

Figura 5 – SoC ESP32-D0WD Q6

Disponível em: https://www.espressif.com/en/products/hardware/socs. Acesso em fev 2020

Figura 6 – Módulo ESP32-WROOM-32

Disponível em: https://www.espressif.com/en/products/hardware/modules. Acesso em fev 2020

Esses módulos também podem ser instalados em placas de desenvolvimento, onde são associadas a reguladores de tensão, pinagem para as portas, conversores serial para gravação do programa via USB, LEDs de indicação, etc. Tais placas democratizaram o desenvolvimento de projetos com a funcionalidade de acesso ao Wi-Fi, devido ao seu baixo custo, às plataformas de desenvolvimento disponíveis, e à grande comunidade que utiliza os módulos ESP32.

2.2.3 *Espressif IoT Development Framework*

Assim como boa parte dos microcontroladores difundidos na atualidade, o ESP32 pode ser programado através da plataforma Arduino, que possui as conveniências de incluir uma camada de abstração que facilita a programação e um ambiente de desenvolvimento integrado (IDE, do inglês *Integrated Development Environment*) que permite a compilação e a gravação do programa em diversas placas de desenvolvimento.

No entanto, essa camada de abstração também dificulta a programação de muitas funcionalidades mais complexas, como interrupções e *timers*. Sendo assim, a Espressif fornece a possibilidade de se utilizar o ambiente de desenvolvimento *Espressif IoT Development Framework* (ESP-IDF), que é o *framework* oficial de desenvolvimento para ESP32. Esse ambiente exige a configuração manual de muitos recursos internos e periféricos e também o conhecimento mais aprofundado da linguagem de programação C embarcada e da arquitetura do microcontrolador. No entanto, justamente essa possibilidade de programação com menos níveis de abstração permite que o projetista tenha um controle muito maior dos recursos do *hardware* e possa aproveitar ao máximo o que a placa pode oferecer. Ele também contém uma série de interfaces de programação de aplicações (API, do inglês *Application Programming Interface*) nativas que facilitam operações mais complicadas, como a conexão com uma rede Wi-Fi.

O ESP-IDF também pode ser instalado no PlataformIO, um ecossistema para o desenvolvimento embarcado compatível com uma série de plataformas e de placas, com o foco em IoT. O PlatformIO possui uma IDE que torna mais fácil instalar o ambiente de desenvolvimento ESP-IDF no computador, e configurar, compilar e gravar o programa no microcontrolador.

2.2.4 FreeRTOS

Um sistema operacional (OS, do inglês *Operating System*) é um programa, ou conjunto de programas, responsável por gerenciar os recursos de *hardware* de um computador, além de controlar quais processos serão executados em um determinado instante (SILBERSCHATZ; GALVIN; GAGNE, 2012). Para microcontroladores mais avançados, como é o caso do ESP32, não somente se faz necessária a utilização de um sistema operacional, como também esse deve se orientar a aplicações em tempo real, responder a eventos

externos rapidamente e completar todos os serviços requisitados em um intervalo máximo de tempo. Isso define um sistema operacional em tempo real (RTOS, do inglês *Real Time Operating System*).

Existe uma série de RTOSs disponíveis comercialmente, dentre os quais se destaca o FreeRTOS, por ser simples, leve, robusto e distribuído sob a licença MIT. Além disso, a maioria das APIs nativas do ESP-IDF são construídas sobre o FreeRTOS, sendo necessária a sua utilização para o desenvolvimento de qualquer programa.

2.3 Conectividade

2.3.1 Protocolo TCP/IP

A pilha de protocolos TCP/IP refere-se a um conjunto de protocolos que permite a comunicação entre qualquer tipo de dispositivo (SOCOLOFSKY; KALE, 1991), cujo nome se origina de dois protocolos: Protocolo de Controle de Transmissão (TCP, do inglês *Transmission Control Protocol*) e Protocolo de Internet (IP, do inglês *Internet Protocol*). A pilha pode ser dividida em cinco camadas, conforme Figura 7, sendo cada camada responsável por um aspecto da comunicação (GORALSKI, 2017):

Figura 7 – Camadas da pilha TCP/IP

Fonte: o próprio autor

- **Camada física**: Apresenta as funcionalidades necessárias para transportar o fluxo de dados através do meio físico até outro sistema.
- **Camada de enlace**: Interpreta logicamente os dados transmitidos pela camada física, realizando o enquadramento, o endereçamento físico e a detecção de erros.
- **Camada de rede**: Entrega os dados, da fonte até o destino, na forma de pacotes, através de quantos enlaces forem necessários.
- **Camada de transporte**: Garante que os pacotes da camada de rede cheguem de forma consistente ao destinatário para formar uma mensagem completa.

- **Camada de aplicação**: Contém os protocolos para um serviço específico de comunicação, servindo como uma ponte entre a camada de transporte e o usuário.

A pilha TCP/IP firmou-se como a principal solução para transmissão de dados por possuir algumas características como (SOCOLOFSKY; KALE, 1991):

- **Padronização**: É um protocolo roteável completo e o mais aceito atualmente, padronizando a transmissão de dados pela Internet.

- **Interconectividade**: Permite a conexão de sistemas distintos.

- **Roteamento**: É compatível com tecnologias antigas e recentes.

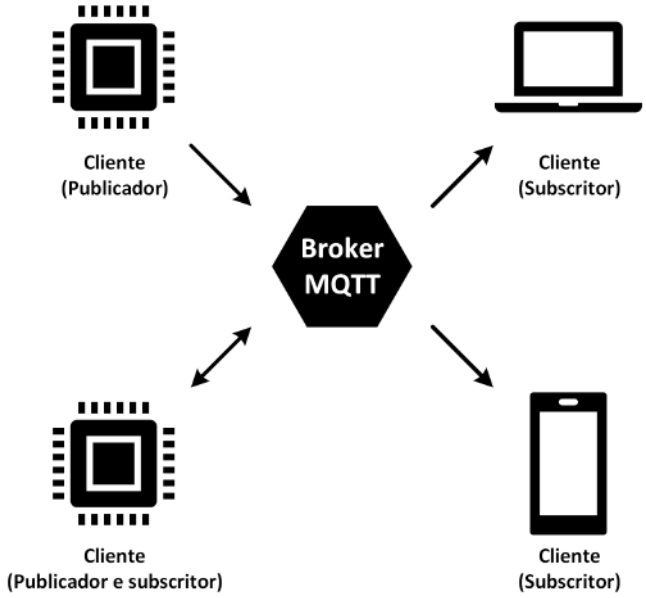
- **Robustez**: Garante a confiabilidade da transmissão de dados, mesmo que haja uma grande distância entre a fonte e o destino.

2.3.2 Protocolo MQTT

O protocolo MQTT (do inglês, *Message Queuing Telemetry Transport*) é um protocolo de comunicação máquina-máquina (M2M, do inglês *machine-to-machine*) e focado em IoT, projetado para rodar sobre o protocolo TCP/IP de maneira leve e confiável (HILLAR, 2017). É recomendado para situações em que vários dispositivos precisam trocar dados entre si em tempo real, mas consumindo o mínimo possível de banda de Internet.

O protocolo é arquitetado considerando três elementos principais: um dispositivo publicador (em inglês, *publisher*) envia dados a um *broker*, que recebe as publicações, filtra as mensagens e encaminha para um subscritor (em inglês, *subscriber*), como ilustra a Figura a 8. Pode-se dizer, portanto, que trata-se de um protocolo cliente-servidor, em que o cliente pode ser um publicador, um subscritor, ou ambos. O controle dos fluxos de mensagens é feito através de tópicos: um cliente publica suas mensagens em um tópico e, sempre que isso ocorre, os outros clientes que estão assinando esse tópico recebem a mensagem.

Figura 8 – Diagrama do protocolo MQTT



Fonte: o próprio autor

Um outro conceito importante é o de Qualidade de Serviço (QoS, do inglês, *Quality of Service*), que pode variar de 0 a 2 (IBM, 2020):

- **QoS 0**: Modo de transferência mais rápido em que a mensagem não é armazenada e o emissor só tenta entregá-la uma única vez; caso o emissor seja desconectado nesse momento, a mensagem se perde.

- **QoS 1**: Modo de transferência padrão, a mensagem é armazenada no emissor e, se esse não receber uma confirmação de que a mensagem foi entregue, ele tenta novamente até concluir a entrega, podendo enviar várias vezes a mesma mensagem.

- **QoS 2**: Modo de transferência mais lento, a mensagem é armazenada no emissor e no receptor até ser processada e a mensagem é entregue apenas uma vez.

Considerando uma operação com QoS 1, os seguintes comandos são utilizados durante a comunicação:

Tabela 1 – Comandos do protocolo MQTT

Nome	Direção	Descrição
CONNECT	Cliente a servidor	Requisição de conexão
CONNACK	Servidor a cliente	Requisição de conexão reconhecida
PUBLISH	Bidirecional	Publicação de uma mensagem
PUBACK	Bidirecional	Publicação reconhecida
SUBSCRIBE	Cliente a servidor	Requisição de subscrição
SUBACK	Servidor a cliente	Requisição de subscrição reconhecida
UNSUBSCRIBE	Cliente a servidor	Requisição de cancelamento de subscrição
UNSUBACK	Servidor a cliente	Requisição de cancelamento reconhecida
PINGREQ	Cliente a servidor	Requisição de PING
PINGRESP	Servidor a cliente	Resposta de PING
DISCONNECT	Bidirecional	Notificação de desconexão

Fonte: Banks et al. (2019)

2.3.3 AWS IoT Core

A AWS IoT Core é um serviço de nuvem oferecido pela Amazon Web Service que permite a conexão segura de dispositivos a aplicativos de nuvem. Esse núcleo aceita alguns protocolos de comunicação como o MQTT, em que o serviço funciona como *broker* (AMAZON WEB SERVICES, 2020a). Além disso, existe a funcionalidade do gerenciamento de dispositivos conectados, que torna mais fácil a tarefa de registrar, organizar, monitorar e gerenciar remotamente os dispositivos que estão conectados à nuvem.

Este sistema também possui grande integração com várias outras plataformas, como o sistema operacional FreeRTOS, no qual é possível instalar uma biblioteca que permite a conexão do microcontrolador a esta nuvem. Também existe a compatibilidade com muitas linguagens de programação como, por exemplo, Python, em que a própria AWS disponibiliza e mantém bibliotecas com APIs para facilitar a conexão com a nuvem.

No que se diz respeito à segurança, o sistema utiliza o protocolo de segurança TLS (do inglês, *Transport Layer Security*) com certificados X.509 para a autenticação das conexões (CORBETT, 2017). Nessa autenticação, é necessário que o cliente apresente o certificado comum da AWS (certificado CA), para garantir que ele está se comunicando com o servidor geral da AWS. Em seguida, o servidor solicita um certificado X.509 e o valida frente à conta da AWS em relação ao registro de certificados. Por fim, o servidor requisita o cliente a provar a propriedade de sua chave privada que corresponde à chave pública contida no certificado (AMAZON WEB SERVICES, 2020b).

2.3.4 Python

O Python é uma linguagem de programação que ganhou espaço nos últimos anos, sendo considerada uma das mais utilizadas em diversas áreas de desenvolvimento. Sua

universalidade é resultado de uma combinação de elementos que facilitam a montagem de projetos, dentre eles sua legibilidade, versatilidade e bibliotecas externas que possibilitam mais trabalho com menos códigos. É uma linguagem de alto nível, multiparadigma, modular e funcional (PYTHON SOFTWARE FOUNDATION, 2020). Essas capacidades proveram sua aplicação disseminada em sistemas de IoT, principalmente porque o Python permite implementar a lógica de negócios diretamente a nível do dispositivo, diminuindo o volume de dados a serem tratados.

2.3.5 SQLite

Um banco de dados pode ser definido como um conjunto de dados relacionados entre si sendo que um dado é um fato que pode ser gravado e que tem um significado implícito (ELMASRI; NAVATHE, 2010). Uma das formas de se organizar um banco de dados é através de uma tabela, isto é, em linhas e colunas, estrutura denominada Banco de Dados Relacional. Esse tipo de banco de dados pode ser operado através de comandos na linguagem SQL (do inglês, *Structured Query Language*). A interação do usuário com um banco de dados é facilitada através de um conjunto de *software* denominado Sistema de Gerenciamento de Banco de Dados (DBMS, do inglês *Database Management System*).

O SQLite é uma biblioteca escrita em linguagem C para gerenciar banco de dados nas mais diversas plataformas. Executando comandos SQL, essa biblioteca se propõe como uma implementação leve, rápida, independente e completa (SQLITE, 2020a). Ao contrário de grande parte dos gerenciadores, o SQLite não necessita de implementação de um servidor dedicado para o gerenciamento, já que o processo que necessita acessar o banco de dados pode ler e escrever diretamente no arquivo do banco no disco (SQLITE, 2020b).

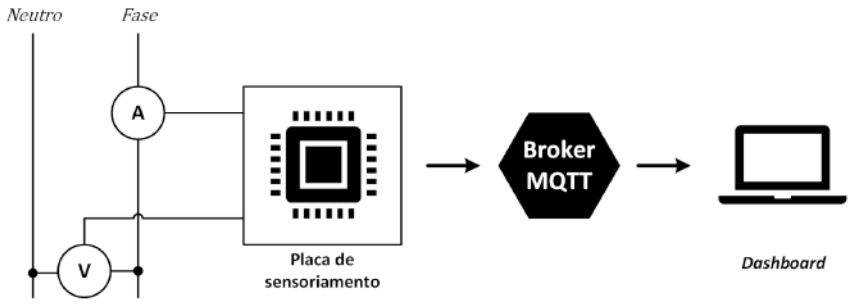
3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados as etapas da implementação do projeto, abordando os sensores e o microcontrolador escolhidos, o *software* desenvolvido para o sensoriamento, a estrutura em nuvem criada e o *software* para receber, armazenar e exibir os dados enviados pelos sensores.

3.1 Descrição do projeto

Um microcontrolador conectado a sensores de tensão e corrente é responsável por tomar as medições da rede, realizar todos os cálculos pertinentes e enviar a um broker. Um servidor, então, faz a requisição dos dados, salva em um banco de dados e os exibe em um *dashboard*. A Figura 9 ilustra o caso considerando apenas a medição da fase principal, tendo em vista que o sistema completo possui mais sensores de correntes.

Figura 9 – Arquitetura do sistema



Fonte: o próprio autor

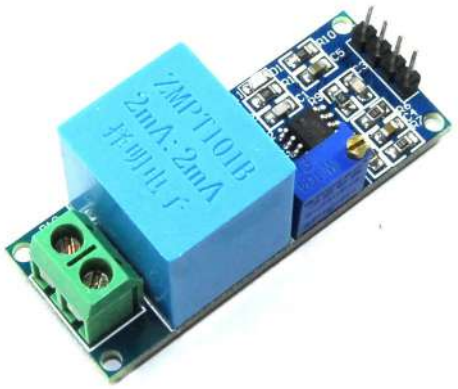
3.2 *Hardware* de sensoriamento

A placa de sensoriamento foi construída utilizando-se o sensor de tensão ZMTP101B, o sensor de corrente SCT-013 e a placa de desenvolvimento NodeMCU-32S. Todos esses dispositivos foram conectados em uma placa de circuito impresso e a outros componentes, como condicionadores de sinais, LEDs de indicação e capacitores de filtragem.

3.2.1 Sensor de tensão

O sensor de tensão ZMTP101B (Figura 10) é um módulo que acopla um transformador de tensão a um amplificador operacional. Assim, a tensão é atenuada a uma taxa constante no transformador e, no estágio do amplificador, é inserido um nível DC de aproximadamente $2,5V$ e é possível regular o ganho através de um potenciômetro. Sendo assim, o sensor isola galvanicamente a tensão medida da tensão analógica entregue que será lida pelo microcontrolador.

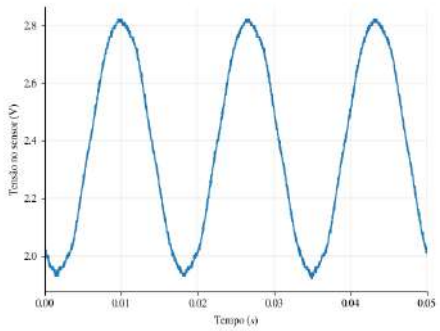
Figura 10 – Sensor de tensão ZMPT101B



Disponível em: http://newkarachielectronics.com/index.php/product/ac-voltage-sensor-module-zmpt101b-250v-voltage-transformer-module-arduino/

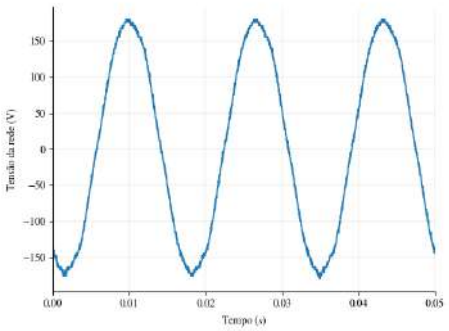
A Figura 11 apresenta os valores lidos pelo sensor quando conectado a uma rede de $V_{RMS} = 127V$, ou seja, a uma tensão de pico de $V_P = 179.6$. Nessas condições, a conversão resultou em uma onda que oscilava entre os valores máximos e mínimos de, respectivamente, $v_{max} = 2,79V$ e $v_{min} = 1,86V$. Essa onda obtida no sensor equivale a uma tensão de pico de $v_p = 0,45V$, indicando uma taxa de conversão de $\alpha = 399.12$. Removendo o valor médio $v_{med} = 2,31V$ da série de dados que foi inserido pelo amplificador operacional, e aplicando a taxa de conversão encontrada, chega-se à Figura 12.

Figura 11 – Tensão convertida pelo sensor



Fonte: o próprio autor

Figura 12 – Tensão da rede



Fonte: o próprio autor

3.2.2 Sensor de corrente

O SCT-013-X é uma família de sensores de corrente, cujo funcionamento se dá através de um transformador de corrente com núcleo de ferrite. Cada sensor possui uma relação de transformação da forma $XA : 1V$, em que o valor X é indicado no modelo do

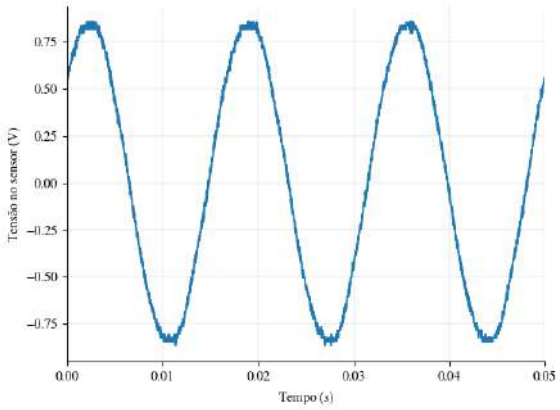
sensor. De tal modo, o sensor SCT-013-050 possui uma taxa de transformação de $50A : 1V$ e consegue medir correntes de até $50A$ de pico.

Figura 13 – Sensor de corrente SCT013

Disponível em: https://www.filipeflop.com/produto/sensor-de-corrente-nao-invasivo-20a-sct-013/

O sensor SCT-013-015 foi acoplado a uma carga resistiva que consumia uma corrente eficaz de $9.24A$, equivalente a uma corrente de pico de $13.07A$, e a Figura 14 apresenta os valores retornados pelo sensor. Nessas condições, foi observada uma tensão de pico resultante de $v_p = 0.86V$, o que equivale a uma taxa de conversão de 15.19. Isso equivale a um erro de apenas 1.27% no valor nominal de conversão.

Figura 14 – Corrente medida na carga resistiva



Fonte: o próprio autor

3.2.3 Placa de desenvolvimento

O NodeMCU-32S 15 é uma placa de desenvolvimento que embarca a módulo ESP-WROOM-32 que, por sua vez, utiliza o SoC ESP32-D0WDQ6. Esse é um dispositivo

conta com um microprocessador dual core de baixo consumo Tensilica Xtensa 32-bit LX6 com suporte embutido a redes Wi-Fi e Bluetooth v4.2 e memória flash integrada.

Figura 15 – Placa de desenvolvimento NodeMCU-32S

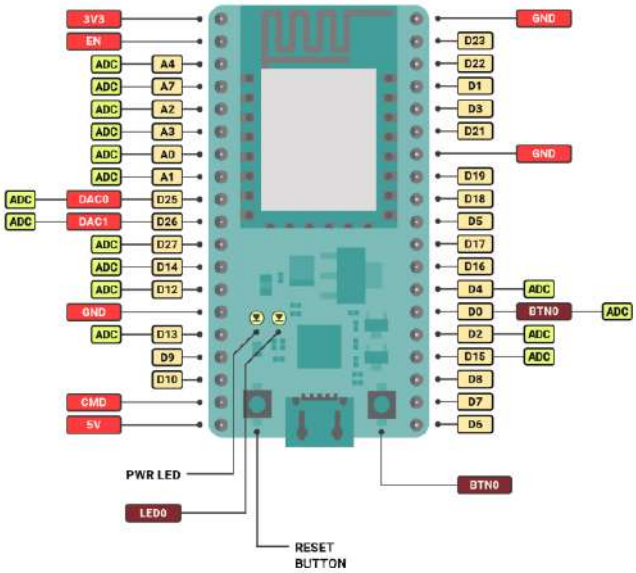
Disponível em: https://www.shenzhen2u.com/NodeMCU-32S

A pinagem da placa encontra-se na Figura 16 e, dentre suas principais características, citam-se:

- *clock* ajustável de 80MHz até 240MHz
- Memória ROM de 448KB
- Memória SRAM de 8KB
- Memória flash externa de 4MB
- Faixa de operação de 2.2VDC - 3.6VDC
- Alimentação de 5VDC através do conector micro USB
- Nível lógico de 3.3V
- Corrente típica de 80mA
- Corrente máxima por porta de 12mA
- 36 GPIOs
- 2 ADCs distribuídos em 18 canais com resolução de 12 bits
- Possui antena integrada para Wi-fi e Bluetooth

- 2 grupos de *timers* com 2 *timers* cada, com 64bits e 16bits de divisor de frequência

Figura 16 – Pinagem da placa de desenvolvimento NodeMCU-32S



Adaptado de: https://docs.zerynth.com/latest/official/board.zerynth.nodemcu_esp32/docs

O primeiro conversor suporta 8 canais, enquanto que o segundo conversor suporta 10 canais. No entanto, não é possível utilizar o segundo enquanto o Wi-Fi estiver conectado, uma vez que este conversor é utilizado no driver do Wi-Fi. Cada canal permite a configuração de uma atenuação para medir diferentes faixas de tensão, conforme Tabela 2:

Tabela 2 – Relação entre a atenuação e o fundo de escala do ADC

Atenuação (dB)	Fundo de escala (V)
0	1.1
2.5	1.5
6	2.2
11	3.9

Fonte: ESPRESSIF (2019a)

3.2.4 Estrutura da placa

A placa de sensoriamento foi alimentada por uma fonte de tensão de 5V conectada através de um borne e filtrada por um capacitor. Essa tensão alimenta a placa NodeMCU-32S que, por sua vez, regula internamente para o nível de 3.3V.

Para que fosse possível mensurar todas as variáveis de interesse, foi necessário utilizar um sensor de tensão para medir uma fase da rede, um sensor de corrente para cada fase da rede, e três sensores de corrente para serem distribuídos entre os circuitos restantes da residência.

Os sensores de corrente aplicam uma transformação praticamente linear ao sinal original e, portanto, resultam em tensões negativas quando a corrente também é negativa. O ADC do microcontrolador, entretanto, não é capaz de medir tensões negativas. Sendo assim, foi necessário utilizar um potênciometro para criar um divisor de tensão entre a alimentação da placa (+3V3) e o terra (GND), e consequentemente, criar um terra virtual (GNDA), conforme Figura 17.

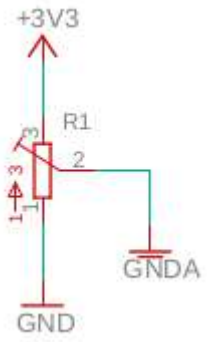


Figura 17 – Divisor de tensão para criar um terra virtual (GNDA)

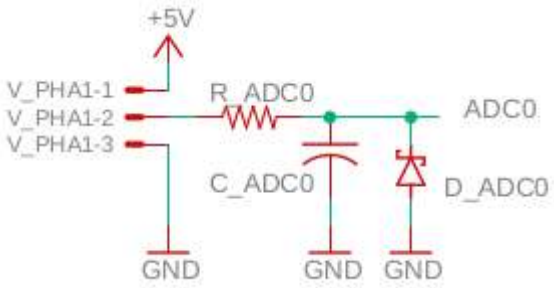
Fonte: o próprio autor

Tanto para o sinal do sensor de tensão quanto para os dos sensores de corrente, foram utilizados filtros passa-baixa para atenuar os ruídos de alta frequência, com resistores de $10k\Omega$ e capacitores de $100nF$, resultando em uma frequência de corte de $f_c = 1.59kHz$ de acordo com a Equação 3.1. Esse valor é mais de vinte vezes maior do que a frequência fundamental da rede de $60Hz$, o que não ocasiona em perda de harmônicos de baixa ordem.

$$f_c = \frac{1}{2\pi RC} \tag{3.1}$$

O sensor é conectado à placa através de um conector triplo, através do qual ele também é alimentado pelos 5V do circuito. Após o sinal do sensor de tensão ser filtrado, ele é conecato ao canal ADC0 do microcontrolador. Para proteger a porta em caso de uma sobretensão, foi associado um diodo tipo Zener de $5.1V$. A interface completa encontra-se na Figura 18.

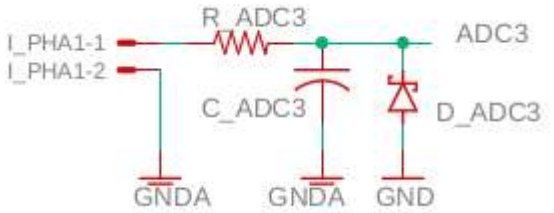
Figura 18 – Interface entre o sensor de tensão e a porta da placa



Fonte: o próprio autor

De maneira similar ao sensor de tensão, os sensores de corrente são conectados à placa através de um conector duplo, são filtrados e o sinal é transmitido ao microcontrolador com a proteção de um zener. No entanto, não é necessário alimentar o sensor, uma vez que se trata de um transformador. Além disso, um dos terminais do sensor é conectado ao terra virtual para adicionar o nível DC ao sinal. A interface é apresentada na Figura 19.

Figura 19 – Interface entre os sensores de corrente e a porta da placa



Fonte: o próprio autor

Também foram utilizados LEDs para indicar a conexão com o Wi-Fi e a conexão com o servidor, conectados a GPIOs do microcontrolador. A Tabela 3 resume o que foi conectado a cada porta.

Tabela 3 – Relação entre os periféricos e as portas aos quais são conectados

Periférico	Porta
Sensor de tensão (Fase 1)	ADC0
Sensor de corrente (Fase 1)	ADC3
Sensor de corrente (Fase 2)	ADC6
Sensor de corrente (Circuito 1)	ADC7
Sensor de corrente (Circuito 2)	ADC4
Sensor de corrente (Circuito 3)	ADC5
LED Wi-Fi	IO17
LED Servidor	IO16

Fonte: o próprio autor

O esquemático completo da placa pode ser encontrado no Apêndice A.

3.3 *Software* de sensoriamento

A arquitetura do *software* responsável por tomar as medidas do sensores, realizar os devidos cálculos e enviar os dados ao servidor pode ser conferida na Figura 20.

3.3.1 Tasks

A função **app_main** é a primeira a ser inicializada no programa e, portanto, é responsável por chamar outras funções que configuram os periféricos.

Nela são chamadas as funções **NVS_config**, dedicada a inicializar a Memória Não Volátil (NVS, do inglês, *Non-Volatile Storage*), **initialize_wifi**, que configura e inicializa a API da conexão Wi-FI, **ADC_config**, responsável por configurar os ADCs que serão utilizados para medir os sensores, e **GPIO_config**, dedicada a definir como saídas as portas que acionaram os LEDs de indicação.

Além disso, nessa função são criadas as *tasks* através da API **xTaskCreate** do FreeRTOS, que serão executadas paralelamente durante todo o programa. As *tasks* **Handler_TIMER0** e **Handler_TIMER1** referem-se às funções que lidarão com as interrupçoes dos *timers*, e a *task* **AWS_IoT** aloca a API da AWS responsável pela conexão com os servidores AWS para o protocolo MQTT.

3.3.2 Conexão Wi-Fi

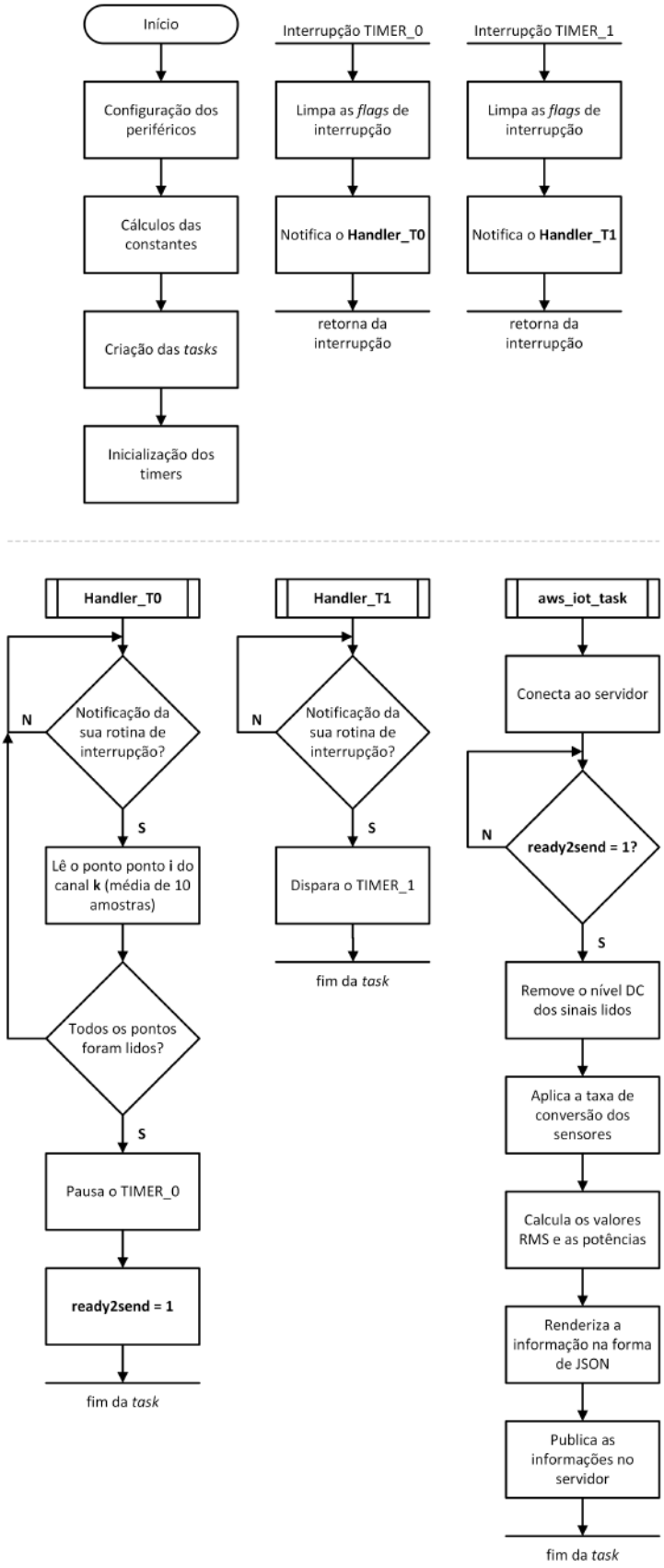
Para utilizar a conexão Wi-Fi do ESP32, é utilizada a biblioteca **esp_wifi.h** da ESPRESSIF. Para utilizar essa biblioteca, é necessário configurar a função **initialise_wifi**, onde são informadas o SSID e a senha da rede que se deseja conectar, e a função **event_handler**, responsável por lidar com as respostas da API quando se tenta iniciar uma conexão Wi-Fi. Uma conexão bem sucedida é indicada pelo acionamento do LED2. As duas funções supracitadas foram baseadas nos códigos fornecidos disponibilizados em (ESPRESSIF, 2019d).

3.3.3 ADC

Em virtude da impossibilidade de se utilizar o segundo ADC ao mesmo tempo que a conexão Wi-Fi, utilizou-se apenas o primeiro ADC. Os canais utilizados foram armazenados em um vetor no cabeçalho do programa, assim como a atenuação e a indicação de qual ADC será utilizado.

Essas variáveis são utilizadas para configurar o ADC na função **ADC_config**. Nesta mesma função, o ADC é configurado para operar com resolução de 12bits e para

Figura 20 – Arquitetura do *software* de sensoriamento



Fonte: o próprio autor

utilizar a tensão de referência que foi gravada na memória *eFuse* durante a calibragem do dispositivo durante sua produção.

3.3.4 *Timers* e rotinas de interrupção

Neste projeto, são utilizados dois timers, cada um sendo responsável por gerar uma base temporal com funções diferentes. Os *timers* são inicializados de acordo com a função **Timer_Init** abaixo, para a qual são passados os parâmetros de interesse, como divisor de frequência e limite de contagem.

O programa funciona em um ciclo principal de 10 segundos, no qual todos os sensores são lidos, os cálculos são executados e os dados são enviados para o servidor. Para isto, foi utilizado o **TIMER1**, configurado com um divisor de frequência $d = 40000$ e contando até $N = 20000$, garantindo intervalos de 10 segundos.

As amostragens dos sensores são configurados para ler 3 ciclos de onda da rede elétrica, dos quais são amostrados 100 pontos. Isso demanda uma taxa de $f = 2kHz$, o que é garantido pelo **TIMER0**, configurado com um divisor de frequência $d = 400$ e contando até $N = 100$.

Quando a contagem dos *timers* é concluída, o programa entra na respectiva rotina de interrupção, que limpa as *flags* e disponibiliza o *timer* para um novo disparo. Além disso, a função **xTaskNotifyFromISR** funciona como um semáfaro para permitir a execução das *tasks* **Handler_TIMER0** e **Handler_TIMER1**.

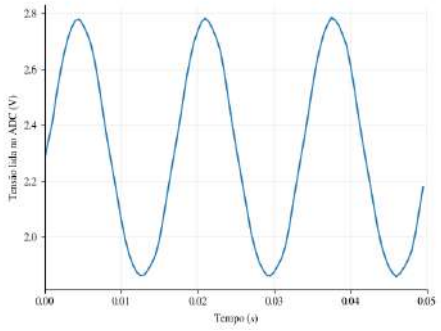
3.3.5 Aquisição dos dados

Conforme exposto anteriormente, o **TIMER0** garante uma taxa amostral para os sensores. Cada interrupção gerada, portanto, habilita o semáforo que permite a execução da *task* **Handler_TIMER0**. Nessa *task*, são executadas 10 leituras do ADC, que são salvas em um *buffer* e, ao final da leitura do sensor, a média das leituras é calculada. Essa operação se repete por 100 vezes, de modo que são amostrados 3 ciclos da rede e, quando as leituras de um sensor são concluídas, iniciam-se as leituras do próximo, até que todos os seis sensores sejam lidos e suas informações armazenadas. Quando isso ocorre, o *timer* é desligado, uma flag indica que os ciclos de leituras foram concluídos e todas os dados são enviados para as etapas de cálculo.

Já o **Handler_TIMER1** é habilitado por sua respectiva interrupção. Como a função desse *timer* é simplesmente garantir intervalos de 10 segundos entre os ciclos de leituras, o **Handler_TIMER1** redispara o **TIMER0** para reiniciar todo o processo.

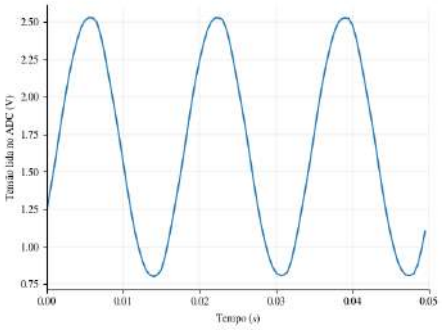
As Figuras 21 e 22 apresentam, respectivamente, as medições do sensor de tensão e do sensor de corrente pela placa de sensoriamento:

Figura 21 – Tensão lida pela placa de sensoriamento



Fonte: o próprio autor

Figura 22 – Corrente lida pela placa de sensoriamento



Fonte: o próprio autor

3.3.6 Cálculos

Caso a residência seja bifásica, para medir a potência consumida pela segunda fase, é necessário conhecer o formato de onda da tensão e da corrente. Entretanto, devido às limitações do número de canais do ADC disponíveis, só está sendo utilizado um sensor de tensão na primeira fase. Para solucionar esse problema, foi criada uma rotina que rotaciona circularmente o vetor que armazena os valores de tensão amostrados da fase medida, com base no número de pontos amostrados durante a medição:

```
void constant_calculations(void)
{
    // Calcula quantos pontos devem ser deslocados para extrapolar a segunda
        ↪ fase
        sample_time = (float)(TO_DIVIDER*TO_OVERFLOW)/CLOCK;
        sample_angle = (float)360*sample_time*f_e;
        points2shift = round(30/sample_angle);
}
```

Todos os valores amostrados são armazenados no vetor **msgBuffer_raw[k][j]**, em que o índice [k] indica qual canal foi lido e o índice [j] indica o ponto da medição. Além disso, para cada canal, a soma de todos os valores amostrados é armazenada no vetor **msgBuffer_sum[k]**, o que é utilizado para calcular o valor médio da série de dados, que é utilizado para eliminar o nível DC da onda medida. Posteriormente, as leituras digitais são convertidas em leituras analógicas considerando o valor do fundo de escala (**VREF**) e a resolução do ADC (12bits) e, por fim, esse valor é multiplicado pela taxa de conversão do sensor, que se encontra armazenada no vetor **ratio[k]**.

Quando todos os sensores são lidos e os valores são devidamente convertidos, eles permitem os cálculos da corrente eficaz conforme Equação 2.5 através de uma função que recebe o vetor como parâmetro e retorna a média quadrática da série de dados. As

medidas de corrente são associadas às suas respectivas medidas de tensão para calcular a potência ativa conforme equação 2.3:

3.3.7 Conexão com o servidor e envio de dados

A AWS fornece e mantem as bibliotecas **aws_iot_config.h**, **aws_iot_log.h**, **aws_iot_version.h** e **aws_iot_mqtt_client_interface.h** para abstrair uma API que conecta com o servidor da AWS IoT Core e comunica via protocolo MQTT. Para que essa conexão seja autenticada, é necessário incluir os certificados de segurança da AWS referentes ao dispositivo autorizado.

A função **aws_iot_task** foi baseada em um código disponibilizado como exemplo (ESPRESSIF, 2019b) e configura o *host* do servidor broker, a identificação do dispositivo, o tópico em que esse será subscrito, o *QoS*, enquanto o **LED3** sinaliza uma conexão bem sucedida com o broker.

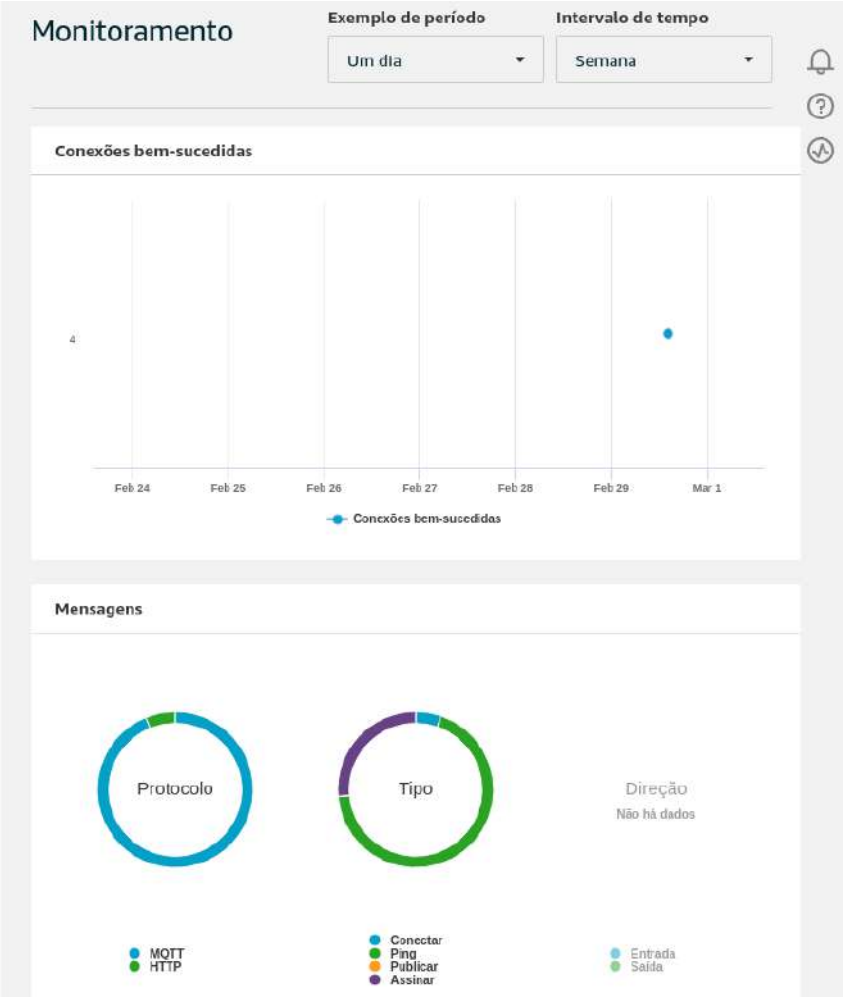
Quando a flag **ready2send** indicar que todas as medições foram feitas, uma string renderizada na forma de JSON tem seus campos preenchidos com os resultados dos cálculos e é enviada ao servidor. Caso a operação não seja bem sucedida, uma mensagem de erro é retornada. A mensagem é renderizada e enviada na seguinte forma:

```
{ "ID": 0,
    "Phase 1": {
        "CURRENT": 0,
        "POWER": 0
    },
    "Phase 2": {
        "CURRENT": 0,
        "POWER": 0
    },
    "Circuit 1": {
        "CURRENT": 0,
        "POWER": 0
    },
    "Circuit 2": {
        "CURRENT": 0,
        "POWER": 0
    },
    "Circuit 3": {
        "CURRENT": 0,
        "POWER": 0
    }}
```

3.4 AWS IoT

Ao criar uma conta na AWS e acessar o serviço AWS IoT, automaticamente um servidor *broker* é configurado, cuja página inicial é exibida na Figura 23.

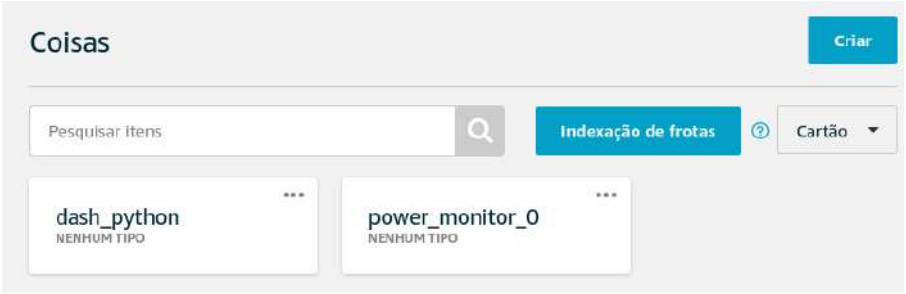
Figura 23 – Página inicial do serviço AWS IoT



Fonte: o próprio autor

Em primeiro lugar, foi necessário configurar as “coisas” que seriam conectadas a esse servidor e, para cada uma delas, gerar os certificados de segurança que autenticam a conexão com o servidor. Esse certificados são baixados durante a criação e podem ser baixados para serem incorporados aos programas das respectivas “coisas”. A Figura 24 apresenta a aba onde é possível conferir todas as “coisas” que foram criadas.

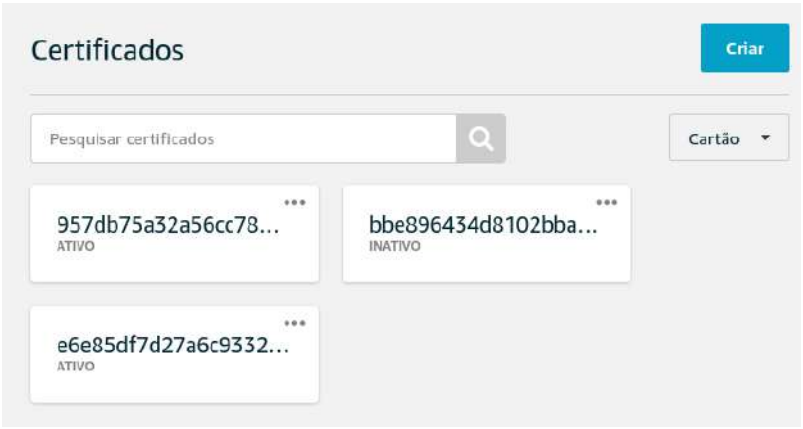
Figura 24 – Lista das “coisas” criadas



Fonte: o próprio autor

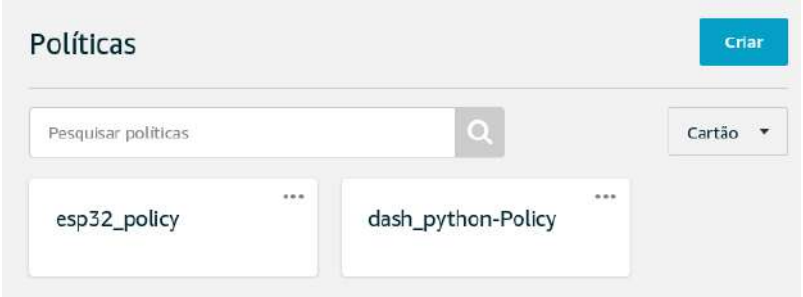
Para cada coisa, é atribuida uma política, que define quais são as permissões de publicação e para quais tópicos essa permissão se aplica. Como será utilizado um único tópico, ambas as “coisas” possuem permissão de publicação e subscrição para todos os tópicos, que no caso se resume a apenas um. A Figura 25 apresenta o painel com todos os certificados ativos ou inativos que foram gerados, e a Figura 26 apresenta as políticas criadas.

Figura 25 – Lista de certificados



Fonte: o próprio autor

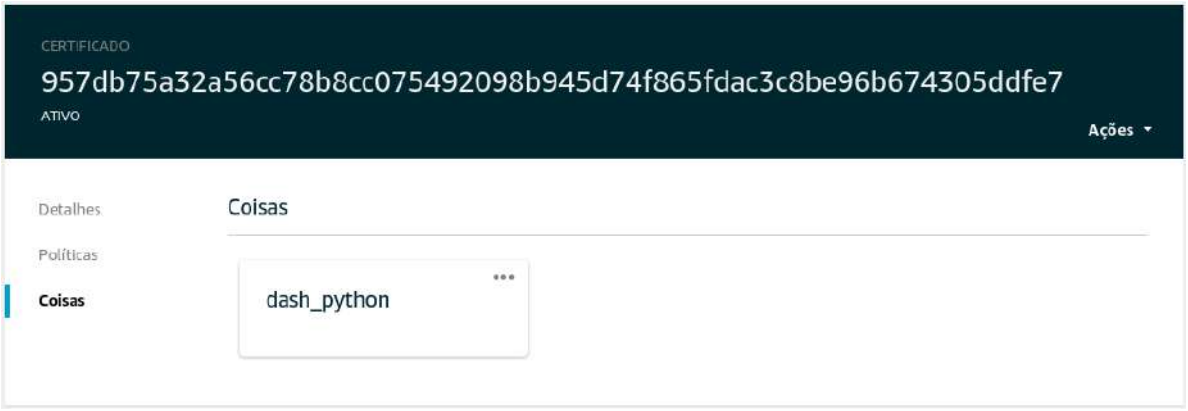
Figura 26 – Lista de políticas



Fonte: o próprio autor

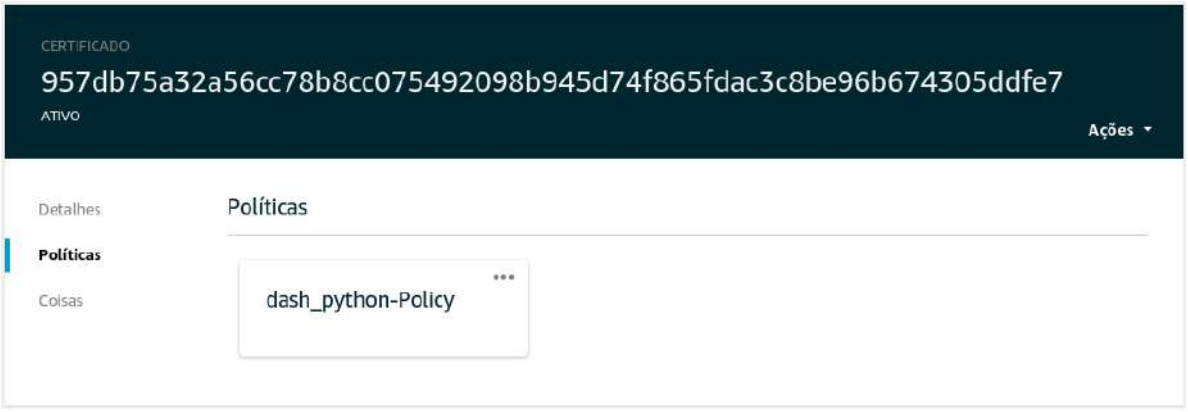
No painel de controles da Figura 25, é possível clicar em cada um dos certificados e conferir a qual “coisa” este está atrelado e quais políticas eles possuem, conforme Figuras 27 e 28.

Figura 27 – “Coisas” atribuídas ao certificado



Fonte: o próprio autor

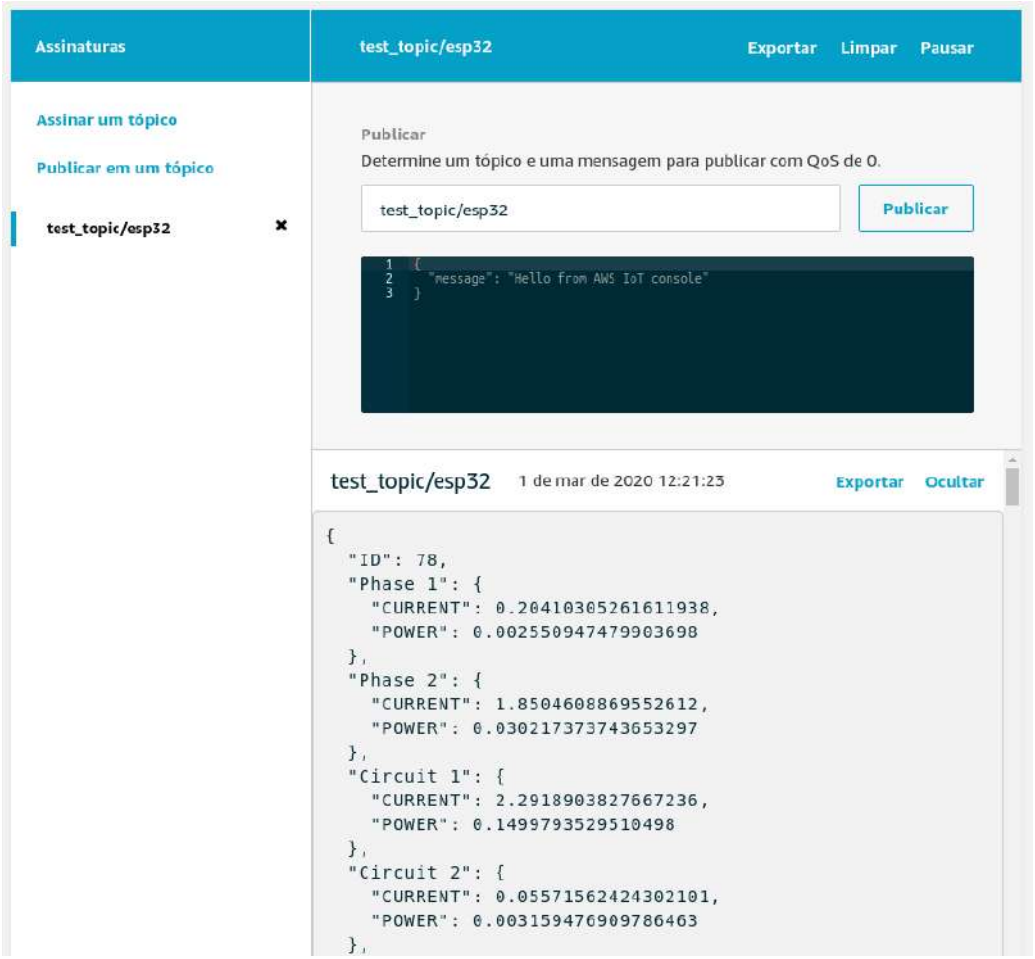
Figura 28 – Políticas atribuídas ao certificado



Fonte: o próprio autor

Por fim, também se pode acessar uma área de testes, em que é possível assinar um tópico para verificar quais mensagens estão chegando, bem como pode-se publicar em um tópico para verificar se os subscritos estão recebendo as mensagens. A Figura 29 apresenta a situação em que foi assinado o tópico no qual a placa de sensoriamento faz suas publicações:

Figura 29 – Página de testes



Fonte: o próprio autor

3.5 *Software* aquisição e exibição dos dados

O servidor para a aquisição e exibição dos dados foi dividido em dois programas escritos na linguagem Python. O primeiro (**back.py**) é responsável por receber os dados do servidor e salvá-los no arquivo do banco de dados. Já o segundo, (**front.py**), acessa o banco de dados para exibi-los em um *dashboard*.

3.5.1 Recebimento dos dados

No programa responsável por receber os dados, é utilizada a biblioteca **AWSIoTPythonSDK**, que cumpre a função de disponibilizar uma API com a qual seja possível se comunicar com o *broker* MQTT da AWS. Sendo assim, o primeiro passo é passar as informações como servidor, certificados de autenticação, identificação do cliente, tópico e o modo (subscritor ou publicador). Em seguida, é possível se conectar ao servidor da AWS e se inscrever no tópico devido.

Ao se estabelecer uma conexão bem sucedida com o servidor, chega-se à etapa de

configurar o banco de dados com a ajuda da biblioteca **sqlite3**, que permite gerenciar um banco de dados SQLite. Inicialmente, caso o arquivo já exista, a tabela de dados é limpa, e caso não exista, o arquivo é criado. Em seguida, é inicializada uma tabela que armazena a corrente e a potência de cada medição, a energia total acumulada (considerando ambas as fases), e a data que a mensagem foi recebida. Posteriormente, a função **mqttCallback** é responsável por receber e decodificar as mensagens sempre que elas são publicadas no servidor.

Em **mqttCallback** anterior, é chamada a função **saveJSON**. Nela, verifica-se se o mês do recebimento da mensagem coincide com o da última mensagem. Em caso positivo, calcula-se a energia consumida nesse período de tempo e soma-se ao último valor e, em caso negativo, desconsidera-se o último valor e inicia-se a medida do novo mês. Em seguida, os valores de cada um dos campos do JSON são inseridos na tabela do banco de dados, anulando os valores negativos.

3.5.2 Exibição dos dados

O programa que realiza a exibição dos dados também utiliza a bibliteca **sqlite3**. O arquivo é aberto no modo de leitura com os comandos da biblioteca e, para realizar a leitura das últimas dez linhas do banco de dados e salvar todos os dados em um vetor, é definida a função **loadBuffer**.

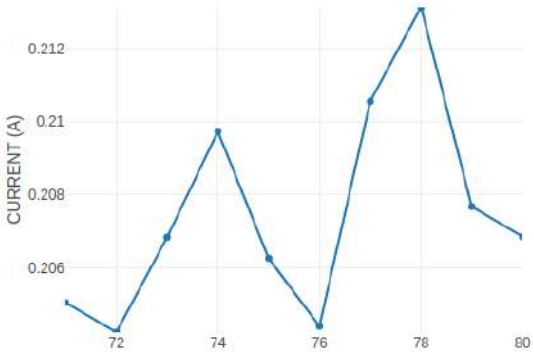
Para exibir os dados, foi utilizada a biblioteca **plotly**, a qual facilita a criação de um *front end* que pode ser embarcado em páginas da internet. Primeiramente, foi definida a estrutura da página no que diz respeito aos elementos que ficarão em cada uma das linhas e das colunas de uma estrutura HTML. Nesse mesmo código, foi inicializado um campo que será responsável pela seleção de qual fase ou circuito se deseja exibir, conforme Figura 30.

Figura 30 – Estrutura para a seleção do que será exibido

Fonte: o próprio autor

A biblioteca **plotly** permite a criação de gráficos interativos que, disparados por algum gatilho, atualizam os seus valores. Para isso, foi criada uma função para cada gráfico exibido, o qual, a cada 10 segundos, chamam a função **loadBuffer** para atualizar o vetor com as últimas leituras gravadas no banco de dados e exibe os dados atualizados. A Figura 31 apresenta um dos gráficos criados com essa bilioteca.

Figura 31 – Exemplo do gráfico de corrente (sensores desconectados)



Fonte: o próprio autor

Os três códigos anteriores podem ser conferidos em sua totalidade no Apêndice B.

4 RESULTADOS E DISCUSSÕES

4.1 Implementação da placa

A placa de sensoriamento foi implementada em um circuito impresso com o auxílio do *software* Eagle para o projeto e confeccionada através do método de transferência térmica, cujo resultado apresenta-se na Figura 32.

Figura 32 – Placa de sensoriamento implementada

Fonte: o próprio autor

Nessa placa, a interface com o sensor de tensão é feita por meio de um conector de três pinos, e com os sensores de corrente se dá por conectores de dois pinos. Além disso, a alimentação da placa é feito por uma fonte de 5 volts que chega à placa através de um borne. A Figura 33 mostra a instalação da placa em uma caixa, associada ao sensor de tensão, a uma fonte de alimentação e a um único sensor de corrente.

Figura 33 – Placa e interface com os periféricos

Fonte: o próprio autor

A Tabela 4 apresenta os custos relacionados à confecção do protótipo no que diz respeito ao *hardware*.

Tabela 4 – Custos refentes à confecção do protótipo

Componente	Quantidade	Valor total (dólares)
Caixa de plástico (15x10x5cm)	1	$4,15
Capacitor cerâmico ($10nF$)	6	$0,05
Capacitor eletrolítico ($470\mu F$)	1	$0,06
Conector borne (2 vias)	1	$0,06
Conector alojamento (2 vias)	5	$0,24
Conector alojamento (3 vias)	1	$0,06
Diodo zener 1W ($3,3V$)	6	$0,18
Fonte de tensão ($5V/700mA$)	1	$0,67
LED 3mm	2	$0,03
NodeMCU ESP32S	1	$3,98
Placa de cobre (10x10cm)	1	$0,77
Resistor 1/4W ($1k\Omega$)	8	$0,04
SCT013	2	$6,42
Trimpot ($10k\Omega$)	1	$0,08
ZMPT101B	1	$1,71
		$18,50

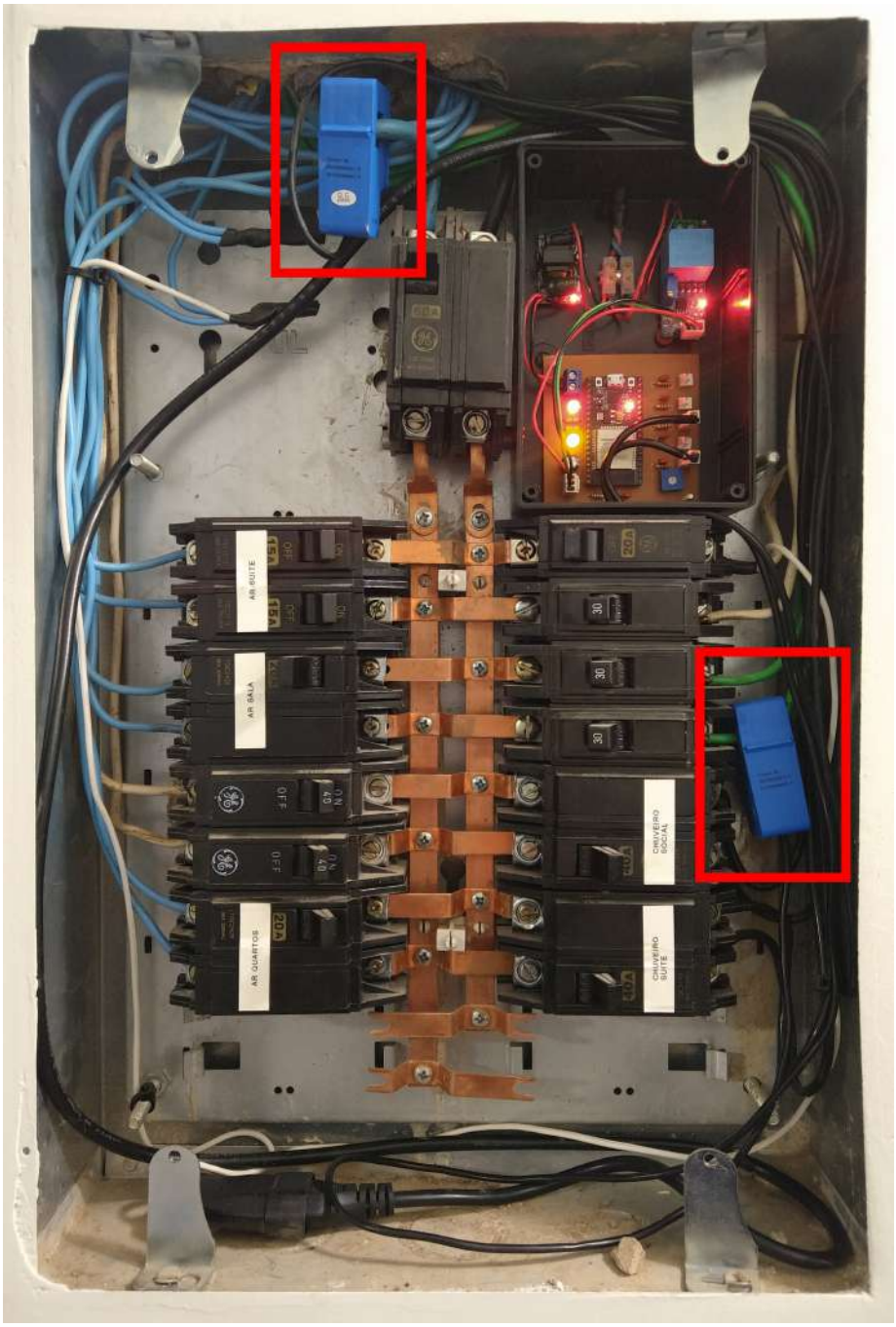
Fonte: o prórpio autor

Caso fossem utilizados sensores de corrente para todos os 5 canais desenvolvidos, haveria um acréscimo de 3,21 dólares para cada canal, totalizando um protótipo 28,13 dólares. No entanto, o uso de apenas dois sensores de corrente foi suficiente para testar a funcionalidade do protótipo.

4.2 Testes

A propósito de testes, o protótipo foi instalado dentro da caixa de distribuição de energia de uma residência. Como se dispunham de apenas dois sensores de corrente, de 50A e 15A, esses foram acoplados, respectivamente, à fase principal (*Phase 1*) e ao circuito da cozinha (*Circuit 1*), sendo que a residência possuia uma configuração bifásica. O arranjo pode ser verificado na figura 34, onde o retângulo vermelho superior indica o sensor de 50A e o inferior, o de 15A.

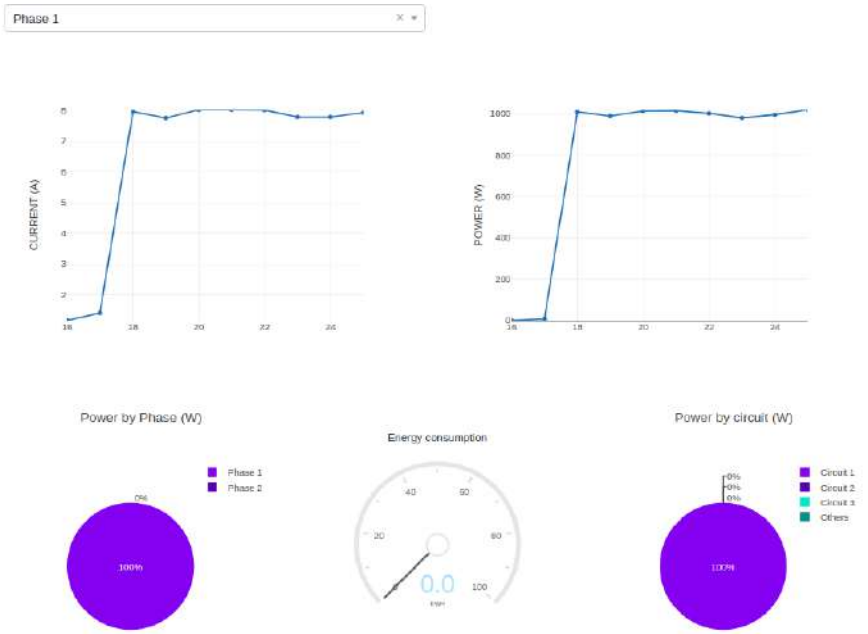
Figura 34 – Instalação na caixa de distribuição de energia



Fonte: o próprio autor

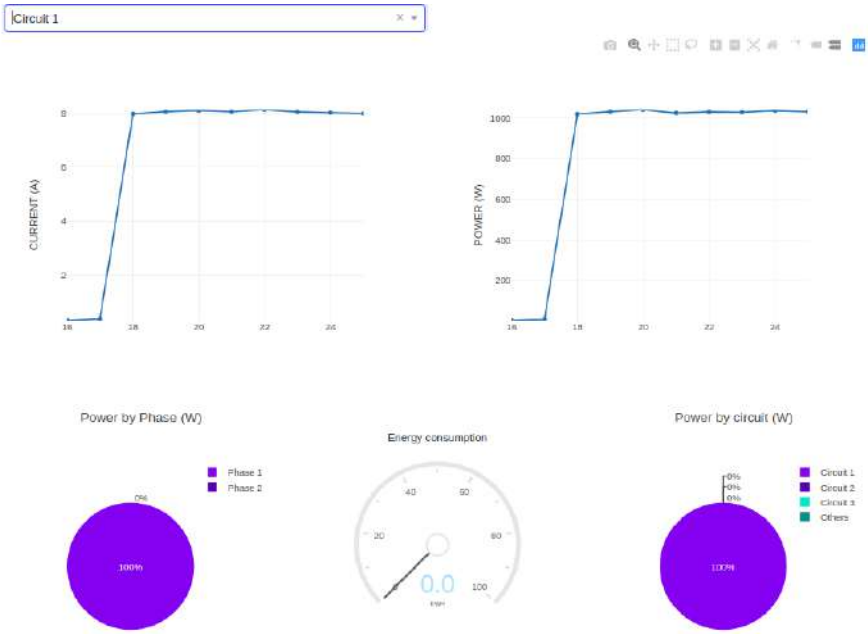
No primeiro teste realizado, todos os equipamentos da casa foram desligados com exceção de uma carga resistiva de 1050W. A intenção era verificar se os dois sensores retornariam valores próximos, tendo em vista que estavam medindo o mesmo equipamento em canais diferentes e com sensores de ganhos diferentes. Os dashboards resultantes das medições se encontram nas figuras 35 e 36.

Figura 35 – Dashboard apresentando as medições da fase 1



Fonte: o próprio autor

Figura 36 – Dashboard apresentando as medições da circuito 1

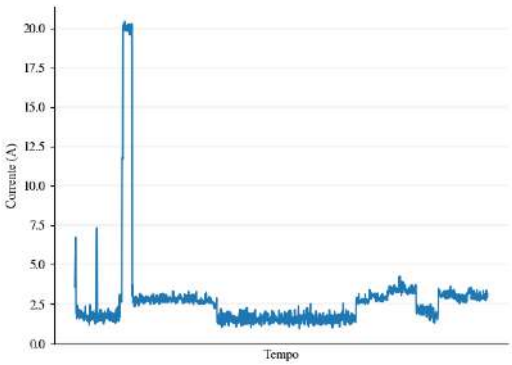


Fonte: o próprio autor

Como era esperado, as duas medições se mostraram próximas, tanto para corrente quanto para potência, além de estarem consistentes com as especificações do aparelho. Ademais, observa-se no dashboard que toda a potência foi atribuída à Fase 1 e ao Circuito 1, já que os outros sensores estavam desconectados.

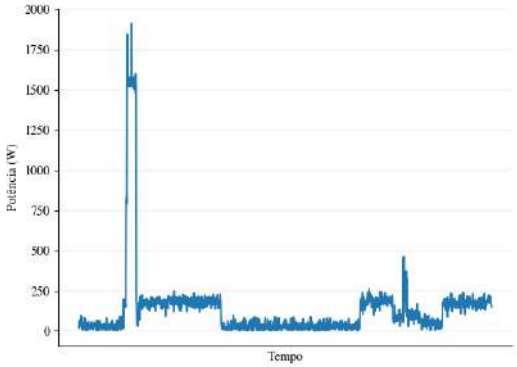
Em seguida, o protótipo monitorou o consumo do domicílio durante o período de 4 horas. A corrente e a potência medidas pelos dois sensores encontram-se nas figuras abaixo.

Figura 37 – Corrente na Fase 1



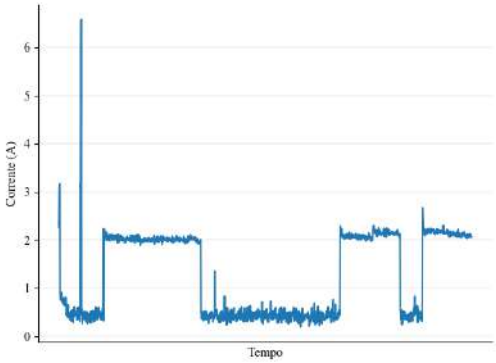
Fonte: o próprio autor

Figura 38 – Potência na Fase 1



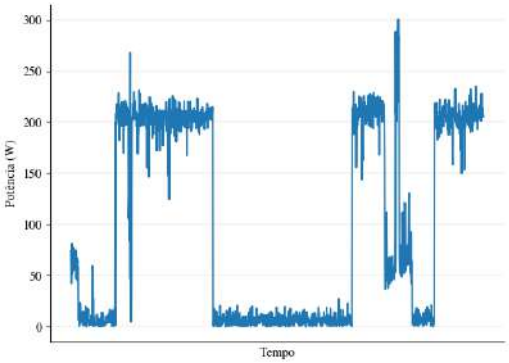
Fonte: o próprio autor

Figura 39 – Corrente no Circuito 1



Fonte: o próprio autor

Figura 40 – Potência no Circuito 1



Fonte: o próprio autor

Os picos percebidos na Fase 1 coincidem com o ligamento de um chuveiro elétrico na sua potência mais baixa. Além disso, é possível perceber oscilações no consumo do Circuito 1 que se devem aos períodos de atividade do compressor da geladeira da cozinha, controlado pelo termostato interno. Ressalta-se, também, que essas oscilações no Circuito 1 são refletidas nos gráficos da Fase 1.

Após a realização dos testes supracitados, o protótipo continuou instalado na caixa de distribuição da residência. Até o momento da escrita desse trabalho, já haviam se passado 3 meses e o protótipo não apresentou nenhum problema de conexão com o WIFI e, mesmo quando o sinal oscilava, o dispositivo era capaz de se reconectar quando este retornava. O mesmo ocorria com a conexão com o *broker* MQTT, tendo em vista que em nenhum momento foi necessário reiniciar o dispositivo por motivos de má conexão com o servidor.

5 CONCLUSÃO

O objetivo a ser alcançado com este trabalho envolvia o desenvolvimento de um medidor de energia elétrica que pudesse ser instalado em qualquer caixa de distribuição e que trouxesse benefícios ao consumidor. Dentre essas vantagens, priorizou-se um equipamento de baixo custo e que fornecesse, a partir de uma interface amigável, informações do padrão de consumo discriminado do domicílio, criando incentivos a um comportamento mais consciente por parte do usuário. Pode-se dizer que o protótipo foi desenvolvido e validado com sucesso em todas as suas etapas.

Os sensores de corrente de tensão se mostraram precisos para medir cargas resistivas. No entanto, sabe-se que o transformador de corrente apresenta limitações quando utilizado para medir cargas não-lineares, tais quais equipamentos com fontes retificadas ou chaveadas. Tendo em vista que parte considerável do consumo de uma residência dá-se devido a cargas resistivas e a motores, o protótipo cumpre o objetivo de monitorar o consumo de energia elétrica da maior parte dos equipamentos.

Em relação ao microcontrolador utilizado, percebe-se que seu ADC teve uma resolução suficiente para realizar todas as medições de tensão e correntes de forma eficiente. O microcontrolador, ainda, apresentou uma boa performance e sua frequência de *clock* permitiu uma alta taxa de amostragem e um baixo tempo necessário para a realização dos cálculos. Além disso, não foram percebidas oscilações na conexão WIFI, permitindo que a placa se mantivesse conectado ao broker.

O protocolo MQTT apresentou-se como uma boa alternativa para o tráfego das mensagens entre a placa de sensoriamento e o *software*, tendo o *broker* AWS IoT Core se apresentado como uma solução robusta e estável.

O programa de aquisição dos dados cumpriu com seu objetivo, pois foi capaz de se conectar ao broker, obter as medições e salvá-los em um banco de dados. Da mesma maneira, o programa de exibição dos dados se mostrou eficaz em acessar o banco de dados e exibir as medições em tempo real em uma interface gráfica. O fato de todas as medições serem salvas em um arquivo permitiu que esses dados fossem acessados manualmente para outros tipos de análises posteriores.

Ao todo, o trabalho desenvolvido foi de grande valia para a formação, uma vez que oportunou conhecimentos e informações sobre o desenvolvimento de plataformas IoT, o que se mostra de grande importância diante da popularização desses recursos nos dias de hoje. A utilização da plataforma ESP-IDF agregou conhecimentos mais avançados de programação de microcontroladores, devido ao baixo nível de abstração da plataforma.

5.1 Trabalhos Futuros

Como sugestão de trabalhos futuros, a mesma estrutura pode ser associada a mais sensores de corrente para validar a medição da segunda fase de uma residência bifásica, como também o número de portas para medir outros circuitos da casa pode ser expandida com o uso de outros microcontroladores ou através do uso de multiplexadores. Também podem ser incluídos outros tipos de cálculos com as informações disponíveis como, por exemplo, o fator de potência de cada circuito. Além disso, é conveniente a criação de um servidor que permita que esses dados sejam acessados através da internet com autenticação de usuário e em diferentes plataformas, como em dispositivos móveis. Por fim, podem ser implementadas análises avançadas dos dados obtidos, o que seria útil, por exemplo, para localizar aparelhos que começassem a apresentar problemas repentinamente, ou para analisar com mais profundidade o perfil de consumo de cada residência.

REFERÊNCIAS

AMAZON WEB SERVICES. **AWS IoT Core**. 2020. Disponível em: <https://aws.amazon.com/pt/iot-core/>. Acesso em: 2020-02-18.

_____. **Certificado do cliente X.509 - AWS IoT**. 2020. Disponível em: <https://docs.aws.amazon.com/pt{_}br/iot/latest/developerguide/x509-client-certs.h>. Acesso em: 2020-02-19.

CORBETT, N. **Understanding the AWS IoT Security Model |**. 2017. Disponível em: <https://aws.amazon.com/pt/blogs/iot/understanding-the-aws-iot-security-model/>.

ELMASRI, R.; NAVATHE, S. **Fundamentals of Database Systems**. 6th. ed. [S.l.]: Addison-Wesley Publishing Company, 2010. ISBN 0136086209.

ESPRESSIF. **Analog to Digital Converter — ESP-IDF Programming Guide**. 2019. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/adc.html>. Acesso em: 2020-02-24.

_____. **Cloud Frameworks — ESP-IDF Programming Guide**. 2019. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/libraries-and-frameworks/cloud-frameworks.html>. Acesso em: 2020-02-26.

_____. **ESP32 Overview**. 2019. Disponível em: <https://www.espressif.com/en/products/hardware/esp32/overview>. Acesso em: 2020-02-16.

_____. **Wi-Fi — ESP-IDF Programming Guide**. 2019. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/network/esp_wifi.html>. Acesso em: 2020-02-25.

FERRERO, A. Voltage Meter Measurement. In: WEBSTER, J. G.; EREN, H. (Ed.). **Measurement, Instrumentation, and Sensors Handbook**. 2. ed. Boca Raton: CRC Press LLC, 2014.

FRADEN, J. **Handbook of Modern Sensors: Physics, Designs, and Applications (Handbook of Modern Sensors)**. 3. ed. San Diego: Springer, 2003. ISBN 0387007504.

FRAMINGHAM, M. **The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast**. 2019. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS45213219>. Acesso em: 2020-02-16.

FREESCALE SEMICONDUCTOR. **How to Increase the Analog-to-Digital Converter Accuracy in an Application**. [S.l.], 2016. Disponível em: <https://www.nxp.com/docs/en/application-note/AN5250.pdf>.

GORALSKI, W. **The Illustrated Network: How TCP/IP Works in a Modern Network**. 2. ed. Elsevier Science, 2017. ISBN 9780128110287. Disponível em: <https://books.google.com.br/books?id=0IzfDQAAQBAJ>.

HEATH, S. **Embedded Systems Design**. 2. ed. USA: Butterworth-Heinemann, 2002. ISBN 0750655461.

HILLAR, G. C. **MQTT Essentials - A Lightweight IoT Protocol**. 1. ed. Birmingham: Packt Publishing, 2017. ISBN 9781787285149. Disponível em: <https://books.google.com.br/books?id=40EwDwAAQBAJ>.

IBM. **Qualidades de serviço fornecidas por um cliente MQTT**. 2020. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SSFKSJ_8.0.0/com.ibm.mq.dev.doc/q029090_.htm>. Acesso em: 2020-02-16.

MALVINO, A. P.; BROWN, J. A. **Digital Computer Electronics**. 3. ed. Glencoe/McGraw-Hill, 1993. ISBN 9780074622353. Disponível em: <https://books.google.com.br/books?id=mUWN0aIIqzkC>.

MCNUTT, D. P. Current Measurement. In: WEBSTER, J.; EREN, H. (Ed.). **Measurement, Instrumentation, and Sensors Handbook**. 2. ed. Boca Raton: CRC Press LLC, 2014.

BANKS, A. et al. (Ed.). **MQTT Version 5.0**. [S.l.], 2019. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.

PYTHON SOFTWARE FOUNDATION. **About Python**. 2020. Disponível em: <https://www.python.org/about/>. Acesso em: 2020-02-23.

RITCHIE, H.; ROSER, M. Access to Energy. **Our World in Data**, sep 2020. Disponível em: <https://ourworldindata.org/energy-access>.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Operating System Concepts**. 9. ed. [S.l.]: Wiley Publishing, 2012. ISBN 1118063333.

SOCOLOFSKY, T. J.; KALE, C. J. **RFC1180: TCP/IP Tutorial**. USA: RFC Editor, 1991.

SQLITE. **About SQLite**. 2020. Disponível em: <https://www.sqlite.org/about.html>. Acesso em: 2020-02-19.

_____. **SQLite Is Serverless**. 2020. Disponível em: <https://www.sqlite.org/serverless.html>. Acesso em: 2020-02-19.

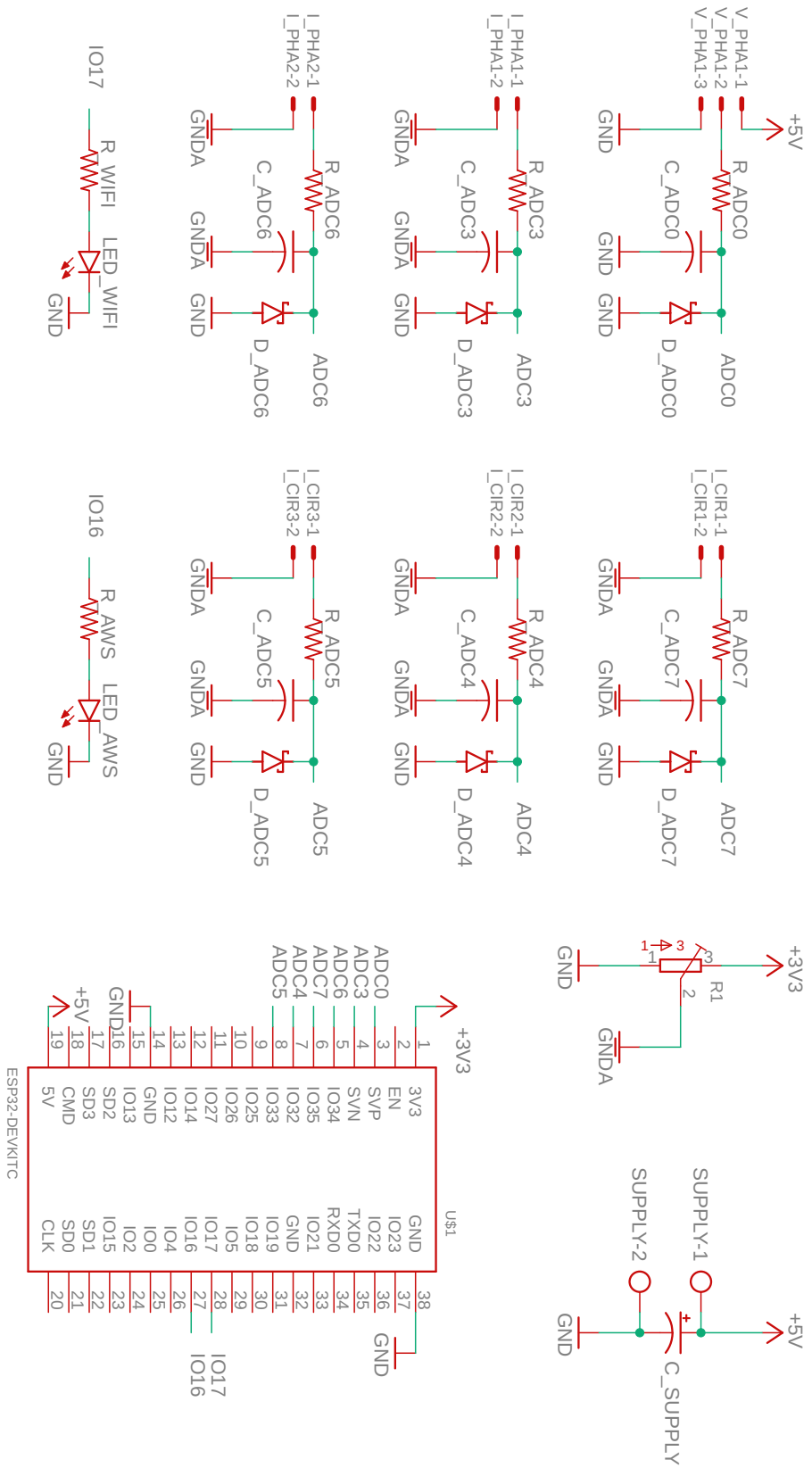
TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Digital systems principles and applications**. 10. ed. [S.l.]: Pearson Prentice Hall, 2017. 822–826 p. ISBN 0-13-172579-3.

WAHER, P. **Learning Internet of Things**. 1. ed. Birmingham: Packt Publishing Ltd, 2015. ISBN 978-1-78355-353-2.

WHITE, E. **Making Embedded Systems: Design Patterns for Great Software**. 1. ed. [S.l.]: O'Reilly Media, 2011. v. 2011. 328 p. ISBN 1449320589.

Apêndices

APÊNDICE A – ESQUEMÁTICO DA PLACA DE SENSOREAMENTO



APÊNDICE B – CÓDIGOS

Os três códigos desenvolvidos nesse protótipo podem ser consultados no repositório virtual: https://github.com/brunomoreirab/Monitor-de-energia