UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA CABO DE SANTO AGOSTINHO
BACHARELADO EM ENGENHARIA ELÉTRICA

SÓSTENES MELO VITORIANO DA SILVA

Sistema de monitoramento IoT de energia conectado à *internet* via ESP32

Cabo de Santo Agostinho - PE
2023

SÓSTENES MELO VITORIANO DA SILVA

Sistema de monitoramento IoT de energia conectado à *internet* via ESP32

Trabalho apresentado ao curso de Engenharia Elétrica da Unidade Acadêmica do Cabo de Santo Agostinho, Universidade Federal Rural de Pernambuco, como requisito para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Denis Keuton Alves.

Cabo de Santo Agostinho - PE

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

S586s Silva, Sóstenes Melo Vitoriano da
Sistema de monitoramento IoT de energia conectado à internet via ESP32 / Sóstenes Melo Vitoriano da Silva. - 2023.
106 f. : il.

Orientador: Denis Keuton Alves.
Inclui referências e apêndice(s).

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal Rural de Pernambuco, Bacharelado em Engenharia Elétrica, Cabo de Santo Agostinho, 2023.

1. monitoramento remoto. 2. IoT. 3. ESP32. 4. big data. 5. Firebase. I. Alves, Denis Keuton, orient. II. Título

CDD 621.3

SÓSTENES MELO VITORIANO DA SILVA

Sistema de monitoramento IoT de energia conectado à *internet* via ESP32.

Trabalho apresentado ao Curso de Engenharia Elétrica da Unidade Acadêmica do Cabo de Santo Agostinho da Universidade Federal Rural de Pernambuco como requisito para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovado em: 19 / 09 / 2023

Banca examinadora

_____

Prof. Dr. Denis Keuton Alves, UFRPE
Orientador

_____

Prof. Dr. Felipe Alberto Barbosa Simão Ferreira, UFRPE
Examinador

_____

Prof. Dr. João Henrique Correia Pimentel, UFRPE
Examinador

DEDICATÓRIA

Dedico esse trabalho a todos que contribuíram na minha trajetória acadêmica, profissional e pessoal, em especial aos meus pais Sandra Maria e Celso Vitoriano e aos meus tios Edvanio Sérgio e Cleide Melo.

AGRADECIMENTOS

A Deus, o qual dá sentido a todas as coisas.

Sou grato a minha família por todo o apoio que tem demonstrado durante toda a minha vida. Aos professores da universidade que me guiaram e fizeram desta formação acadêmica mais que uma graduação. A todos os meus companheiros que passaram por inúmeras noites em claro comigo para que chegássemos a aprovação, a saber: Adelson, Cláudio, Eduardo, Maria Eduarda, Nailson e Pedro.

Agradeço ao meu orientador, Prof. Dr. Denis Keuton Alves, por todo esforço, compreensão e paciência em me instruir para a execução deste trabalho.

Agradeço a minha mãe, Sandra Maria de Melo Vitoriano, por todo o amor. Ao meu pai, Celso Vitoriano da Silva, pelo apoio. À minha avó, Maria Severina de Melo, pelo cuidado. Aos meus irmãos Emily Sonally e Sallis Melo, pelo carinho.

Aos meus tios, Edvanio Sérgio do Nascimento e Maria Lucicleide Melo do Nascimento, e aos meus primos Sandy Layla e Eder Sillas, por serem meu lar, em todos os sentidos da vida.

Em especial, agradeço a Emilly Camila Marques Moreira, por ser amável e companheira em todos os momentos, sempre me motivando e sendo representação fiel do cuidado de Deus nos dias mais difíceis.

RESUMO

Em consequência do desenvolvimento tecnológico observado nas últimas décadas, têm sido cada vez mais comum a utilização de dispositivos inteligentes no dia a dia. O conceito de computação pervasiva ou ubíqua explica esse fenômeno ao propor integrar o mundo virtual ao mundo físico, tornando a tecnologia indissociável e imperceptível ao desenvolvimento de atividades cotidianas das pessoas. Alinhado, surge o conceito de IoT (do inglês, *Internet of Things*), impulsionando uma onda de *upgrade* de dispositivos comuns e que são empregados no dia a dia. Utilizando da capacidade computacional, os dispositivos passam a produzir uma quantidade massiva de dados, criando um ambiente favorável para aplicação das técnicas *big data* durante o processo de armazenamento e análise dos dados. Imerso nesse contexto, o presente trabalho tem como objetivo desenvolver um sistema de monitoramento baseado em IoT e que utiliza a comunicação MQTT (do inglês, *Message Queuing Telemetry Transport*) para estabelecer a supervisão remota de qualquer dispositivo de medição conectado ao *broker*, armazenando amostras da medição de grandezas elétricas no banco de dados *Realtime Database* do Firebase para permitir o acesso instantâneo aos dados por meio de uma página *web* de monitoramento remoto. Os resultados demonstraram a eficácia da combinação do protocolo MQTT, do dispositivo ESP32 e do banco de dados do Firebase mediante análise do tempo de processamento e eficiência no armazenamento das informações.

Palavras-chave: monitoramento remoto; IoT; ESP32; *big data; Firebase.*

ABSTRACT

Because of technological development in recent decades, it has become increasingly common to use smart devices in everyday life. The concept of pervasive computing explains this phenomenon by proposing to integrate the virtual world with the physical world, making technology dissociable and imperceptible to the development of people's daily activities. In this way, the concept of IoT (Internet of Things) has emerged, fostering a wave of upgrades in common devices that are used in our daily lives. Equipped with computing capabilities, devices now produce massive amounts of data, creating a favorable environment for applying big data techniques in the process of storing and analyzing data. Immersed in this context, this study aims to develop an IoT-based monitoring system that uses MQTT (Message Queuing Telemetry Transport) communication to establish remote supervision of any measurement device connected to the broker, storing samples of the measurement of electrical quantities in the Firebase Realtime Database to allow instant access to the data via a remote monitoring web page. The results have been demonstrated the effectiveness of the combination of the MQTT protocol, the ESP32 device, and the Firebase database through the analysis of processing time and efficiency in the storage of information.

Keywords: remote monitoring; IoT; ESP32; big data; Firebase.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

PNAD	Pesquisa Nacional por Amostra de Domicílios Contínua
IBGE	Instituto Brasileiro de Geografia e Estatística
IoT	*Internet of Things*
CAGR	*Compound Annual Growth Rate*
MQTT	*Message Queuing Telemetry Transport*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheets*
ARPANET	*Advanced Research Projects Agency Network*
PWM	*Pulse Width Module*
Hz	Hertz
JS	JavaScript
API	*Application Programming Interface*
CLI	*Command Line Interface*
JSON	*JavaScript Object Notation*
IDE	*Integrated Development Environment*
CDN	*Content Delivery Network*
SDK	*Software Development Kit*
CSV	*Comma Separated Values*
RMS	*Root Mean Square*
MATLAB	*MATrix LABoratory*

SUMÁRIO

1 INTRODUÇÃO

1.1 RELEVÂNCIA DO TEMA

Com o poder de conectar pessoas ao redor do mundo, a *internet* demonstrou-se um berço de desenvolvimento para diversas inovações tecnológicas. A partir dela, foi possível inserir as pessoas em um contexto virtualizado, criando um ambiente favorável ao compartilhamento de informações para produção de conhecimento. Segundo dados da Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD Contínua), pesquisa realizada pelo Instituto Brasileiro de Geografia e Estatística - IBGE (2022), em 2021 cerca de 90% dos domicílios brasileiros faziam uso da *internet* no dia a dia, uma evolução de 6 pontos percentuais em comparação com a pesquisa realizada no ano de 2019.

Fruto da ideia de aproximar o mundo virtual do mundo físico, Weiser (1991) apresenta o conceito da computação ubíqua, propondo tornar a tecnologia indissociável do cotidiano das pessoas. A IoT emerge como destaque nesse processo de desenvolvimento, propondo um cenário em que “coisas” pudessem ser conectadas à *internet* e passassem a ser dotadas pela capacidade de se comunicar, cooperando entre si para aumentar a eficiência na execução de tarefas diversas (Cirilo, 2008).

Para Moon (2016), a IoT consiste em um ecossistema em que os objetos físicos conectados geram e consomem informações, produzindo dados a partir de uma cadeia constituída pela coleta, armazenamento e análise de medições, aplicação de fornecimento de serviços ao consumidor, que, por fim, compartilha as análises com outros serviços ou pessoas.

Segundo dados da IoT *Analytics* (Wegner, 2023), o mercado de IoT teve um crescimento de 21,5% em 2022, alcançando o montante de 992 bilhões de reais investidos. A pesquisa estima que a CAGR (do inglês, *Compound Annual Growth Rate*) será de 19,4% no intervalo de 2022 até 2027.

Com o grande fluxo de informações geradas e compartilhadas pelos dispositivos inteligentes, surge uma quantidade massiva de dados complexos relativos ao uso e a condições de qualidade e eficiência do equipamento que não podem ser facilmente gerenciados e armazenados por ferramentas convencionais. Em resposta, emerge o conceito de *big data*, com o objetivo de permitir a entrega de valor de forma sustentada, medir desempenho, criar competências e melhorar o

processo decisório, aumentando a eficiência e eficácia dos processos, fundamentando a tomada de decisão em evidências (McAfee; Brynjolfsson, 2012).

Propõe-se neste trabalho a implementação de um sistema de monitoramento em tempo real e remoto, capaz de receber os dados de um sistema de medição e exibi-los em tempo real por meio de uma aplicação *web*.

1.2 MOTIVAÇÃO

O monitoramento de grandezas elétricas está diretamente atrelado a confiabilidade e qualidade da energia elétrica (Alves, 2015). Sabendo-se que em um cenário tecnológico e em constante evolução, é necessário que o desenvolvimento e aprimoramento de tecnologias seja contínuo, tornando-se imprescindível enxergar potencial e buscar desenvolver conceitos já existentes para propor soluções a desafios contemporâneos, tais como: a implementação das *smart grid*, *cities*, *homes* e monitoramento remoto. Da óptica da coleta de dados, o desenvolvimento de um sistema que armazena medições contribui para um ambiente de desenvolvimento de tecnologias como *big data*, que busca extrair informações relevantes e enxerga correlação entre os parâmetros elétricos e outros dados extraídos de dispositivos inteligentes. Seguindo essa tendência, esse trabalho tem como principal motivação contribuir com o segmento IoT desenvolvendo uma solução para monitoramento em tempo real e remoto, focando no processo de comunicação e entrega dos serviços ao consumidor final.

1.3 OBJETIVOS

Este trabalho tem como objetivo geral desenvolver um sistema de monitoramento remoto que possibilita a conexão com sistemas microcontrolados de medição que permitam a conexão via MQTT, exibindo em plataforma *web* própria os valores de grandezas elétricas em tempo real, em forma gráfica e escrita.

Dentre os objetivos específicos deste projeto, destacam-se:

- Implementar método de comunicação MQTT no microcontrolador ESP32;
- Criar e configurar um banco de dados *Realtime Database* no Firebase;
- Receber e adaptar os dados da comunicação MQTT e enviá-los ao *Realtime Database* via ESP32;
- Desenvolver um *web app* para monitoramento remoto dos dados enviados ao banco de dados em tempo real;

- Simular o envio de medições via comunicação MQTT;
- Analisar o comportamento do sistema quanto a estabilidade e velocidade.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho foi subdividido em sete capítulos, sendo organizado da seguinte forma:

- No capítulo 2 foi abordado o estado da arte, destacando-se conceitos como: IoT, sistemas de monitoramento e *Big Data*, perpassando por importantes questões acerca da evolução histórica, potenciais da tecnologia e os principais dilemas de desenvolvimento do segmento.
- O capítulo 3 traz uma fundamentação sobre teorias de potência, apresentando conceitos e desenvolvendo as expressões algébricas que definem as potências ativa, reativa e aparente, além do fator de potência.
- No capítulo 4 foram apresentadas as partes de *hardware* e *software* que compõem este projeto, analisando as características técnicas dos microcontroladores ESP32 e ESP8266, e do modelo *UNO* da desenvolvedora Arduino. Ademais, foram apresentados os fundamentos sobre as linguagens HTML (do inglês, *HyperText Markup Language*), Javascript e CSS (do inglês, *Cascading Style Sheets*), bem como sobre o Firebase e suas ferramentas. Por fim, apresentou-se a interconectividade entre as partes, trazendo um panorama sobre a estruturação deste sistema de monitoramento.
- O capítulo 5 traz as configurações básicas e o passo-a-passo utilizado no código fonte para implementar a comunicação MQTT, o tratamento dos dados e o envio ao banco por meio do ESP32. Por fim, apresentando o processo de elaboração da aplicação *web*, bem como da sua comunicação com o banco de dados e suas funcionalidades.
- No capítulo 6 são apresentados os resultados de simulação de comunicação MQTT com o ESP32, bem como de performance da aplicação *web*.
- O capítulo 7 traz as considerações finais deste trabalho e algumas sugestões para trabalhos futuros que possam dar continuidade a esse projeto.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 *INTERNET OF THINGS*

Surge, no fim da década de 60, a ARPANET (do inglês, *Advanced Research Projects Agency Network*), como a primeira rede de computadores interligados, criada com o intuito de viabilizar a comunicação para transferência de informações de cunho militar e de pesquisas (MDN Web Docs, 2023). Vendo a possibilidade de utilizar a nova tecnologia como uma ferramenta de difusão e compartilhamento de conhecimento acadêmico, os principais centros de pesquisas e universidades dos Estados Unidos da América (EUA) passaram a se interligar à rede da ARPANET (Abreu, 2009). Em meados de 1980 surge a *internet*, consagrando-se como a “Rede das Redes”, com o diferencial de conectar mundialmente pessoas, pessoas e máquinas, e criando um ambiente favorável para comunicação entre máquinas (Leite; Martins; Ursini, 2017; Quincozes; Tubino; Kazienko, 2019).

Para Mark Weiser (1991), o desenvolvimento da tecnologia não consiste exatamente em colocar os seres humanos dentro de um ambiente virtual, e sim, em trazer o ambiente virtual para o mundo real, de forma que a tecnologia possa ser aplicada no cotidiano das pessoas, de maneira tão diluída que torne-se imperceptível a sua presença no ambiente. Com essa visão, Weiser (1991) apresenta o conceito de computação ubíqua, também chamada de computação pervasiva, que propõe como paradigma tornar a tecnologia peça comum no dia a dia das pessoas, integrando-a a atividades básicas do cotidiano.

De acordo com o *UK Parliamentary Office of Science and Technology* (2006), pode-se seccionar o desenvolvimento da computação pervasiva em três grandes ondas, a primeira onda consiste na era *mainframe computing*, em que um único computador, localizado numa estação de trabalho, era compartilhado entre várias pessoas. A segunda era é denominada de “Era dos computadores pessoais”, em que um computador é utilizado majoritariamente por uma única pessoa, requerendo uma interação mais consciente da máquina com o usuário. Foi nessa era que popularizou-se a utilização dos computadores do tipo *desktop* (computadores de mesa). A terceira e última onda é denominada “Era da computação pervasiva (ubíqua)”. Nela, uma única pessoa utiliza vários computadores e recursos tecnológicos, requisitando da indústria a produção intensiva de milhões de

tecnologias embarcadas por ano. Com esse cenário, a computação se tornaria o plano de fundo para desenvolvimento das mais diversas atividades do ser humano.

Alinhado, Cirilo (2008) argumenta que, segundo o conceito de tecnologia pervasiva, um ambiente inteligente só existe quando este encontra-se saturado por sensores e atuadores, de forma que o usuário encontre naquele espaço um ambiente imerso de tecnologias indissociáveis à ações de cunho comum.

> O conceito de computação pervasiva implica que o computador está embarcado ao ambiente de forma invisível para o usuário, tendo a capacidade de obter informações acerca do ambiente circundante e utilizá-la para controlar, configurar e ajustar a aplicação para melhor se adequar às características do ambiente (Cirilo, 2008, p.1).

De acordo com Hansmann *et al*. (2013), quatro paradigmas norteiam a computação pervasiva: descentralização, diversificação, conectividade e simplicidade. O autor defende que a computação pervasiva deve distribuir a responsabilidade de processamento e tomada de decisões entre uma variedade de dispositivos, cada qual contribuiria de forma autônoma com suas tarefas e funcionalidades específicas, construindo um ambiente heterogêneo de cooperação mútua e consistente. No entanto, um obstáculo surge à descentralização: o paradigma da diversidade dos componentes, tendo em vista que os dispositivos escolhidos para cada tarefa são os que melhor se adequam à execução daquela tarefa em específico, não necessariamente compartilhando da mesma plataforma de comunicação que os outros objetos inteligentes existentes naquele ambiente.

> Questões de especificidade de plataforma são os maiores obstáculos para aplicações e intercâmbio de informações: a capacidade de armazenamento entre os dispositivos são diferentes; processadores distintos impõem diferentes restrições de desempenho e uso de memória; sistemas operacionais são numerosos e frequentemente são executados em um dispositivo em particular; o tamanho e a forma dos dispositivos requerem diferentes plugs, e assim por diante. (Cirilo, 2008, p. 2).

Existem diversas maneiras de conectar um dispositivo a outro, de forma direta ou indireta. Eles podem trocar informações por meio de um *plug* que conecta os dois dispositivos fisicamente, utilizando protocolos MQTT de comunicação, por meio do envio e recebimento de emails, etc. Seja qual for a forma ou as formas de comunicação adotadas entre os dispositivos, a conectividade é a chave para a interoperabilidade entre os equipamentos. Padrões e *softwares* abertos precisam ser

criados para garantir uma comunicação uniforme entre as plataformas, a fim de que as demandas dos mais diversos dispositivos possam ser satisfeitas de forma cooperativa e sincronizada (Hansmann et. al., 2013).

Dessa forma, nasce o conceito de IoT, que é a ideia de que os objetos de uso comum devem comunicar-se entre si, "...munidos de sensores, controladores, circuitos eletrônicos e *softwares*" (Amorim Jr; Ribeiro; Colistete Jr, 2009, p. 1), por meio de uma rede de comunicação, e que, conectados à *internet*, viabilizam o processamento e a troca de informações e de dados de forma automatizada e inteligente.

Destarte, o conceito de IoT soma, nos permitindo criar um ambiente físico inteligente capaz de perceber e se conectar a qualquer aparelho/coisa de uso comum por meio da *internet*, desde que estes tenham capacidade computacional e de comunicação para essa finalidade (Cirilo, 2008).

Para Mattern e Floerkemeier (2010), só é viável falar em IoT graças aos avanços das últimas décadas nas áreas da microeletrônica, tecnologia da comunicação e da informação. Segundo eles, a capacidade de incorporar a objetos comuns, processadores, módulos de comunicação e outros componentes eletrônicos é peça fundamental para o desenvolvimento de uma realidade com IoT.

> A "atualização digital" de objetos convencionais aprimora sua função física, adicionando as capacidades de objetos digitais, gerando assim um valor agregado substancial. Os precursores desse desenvolvimento já são aparentes hoje - cada vez mais dispositivos, como máquinas de costura, bicicletas ergométricas, escovas de dentes elétricas, máquinas de lavar, medidores de eletricidade e fotocopiadoras estão sendo "computadorizados" e equipados com interfaces de rede (Mattern; Floerkemeier, 2010, p. 243, tradução nossa).

A conexão desses "objetos inteligentes" a uma rede de comunicação, como a *internet,* viabilizaria que atividades desenvolvidas em ambientes variados como o industrial, hospitalar, escolar e até residencial, possam ser efetuadas de forma totalmente autônoma e cooperativa. No entanto, desenvolvimentos técnicos complementares são necessários para tornar menor a lacuna entre o mundo virtual e o mundo físico.

2.2 *BIG DATA*

No mundo contemporâneo, direcionado para uma realidade imersa em tecnologia, um novo produto ocupa papel de destaque no mercado: a informação. A partir do processo de sensoriamento e medição, é possível coletar e armazenar informações que, quando analisadas, podem revelar padrões pertinentes sobre o contexto de dados.

O termo *Big Data* é popularmente utilizado para representar o processo de analisar e extrair informações de uma quantidade massiva de dados variados, que possuem uma estrutura consideravelmente complexa de serem processados por meio de métodos convencionais (Chen; Mao; Liu, 2014).

Segundo Santos *et al.* (2016), o conceito de IoT pode ser detalhado em três camadas: Captação, Rede e Processamento. Na camada de captação ocorre a captura dos dados por meio de sensores; enquanto que, na camada de rede ocorrem as transmissões e processamentos das informações da camada anterior. Na terceira e última camada, ocorrem as análises e os processamentos, que se integram às ferramentas *Big Data*.

Quando correlacionados, o grande volume de informações e a heterogeneidade dos dados podem ser utilizado para direcionar políticas de investimento em organizações privadas, prever comportamentos do mercado, auxiliar na concepção de produtos e soluções, e até auxiliar setores governamentais em políticas voltadas para segurança pública, saúde e educação (Chen; Mao; Liu, 2014).

2.3 MONITORAMENTO

O consumo da energia elétrica é comumente utilizado como parâmetro para quantificar o desenvolvimento econômico de uma nação. Sociedades tidas como “em processo de desenvolvimento industrial”, por exemplo, exprimem uma relação de proporcionalidade entre o seu crescimento econômico e o seu consumo energético. Tal relação se dá pelo fato de que o crescimento industrial de um país ou região está diretamente interligado à disponibilidade e exploração de seus insumos, dentre eles o energético (Ferreira Neto; Corrêa; Perobelli, 2016).

Em contraste ao crescimento dos índices de consumo energético, surgem discussões acerca de assuntos como: sustentabilidade e preservação do meio ambiente. Os grandes impactos ambientais decorrentes da instalação e operação de

usinas nucleares, grandes usinas hidrelétricas e termoelétricas alertam sobre a sensível relação de coexistência entre desenvolvimento e sustentabilidade (Santos *et al*., 2020).

Diante do exposto, entende-se que a demanda humana por um consumo consciente e eficiente da energia elétrica desempenha importante papel no mundo contemporâneo. O desenvolvimento de ferramentas que contribuam com o uso racional e eficiente da energia elétrica é de suma importância. Em paralelo, nas grandes cidades, há uma tendência cada vez mais crescente para a modernização do sistema de distribuição de energia elétrica, tanto para monitoramento e faturamento remoto, quanto para acompanhamento dos indicadores de qualidade da energia.

A estas necessidades, os dispositivos baseados em IoT de monitoramento remoto surgem como parte que agrega a uma solução maior, sendo possível desenvolver, a partir desta, importantes ferramentas complementares baseadas em inteligência artificial, para, por exemplo, subsidiar medidas de conscientização adotadas por parte do consumidor ou para a modernização do sistema de distribuição como um todo.

3 TEORIA DAS POTÊNCIAS

Denomina-se potência elétrica o valor mensurável de energia que é absorvida por equipamentos elétricos para executar uma funcionalidade durante um intervalo de tempo predeterminado. Todo equipamento elétrico possui uma potência nominal, de tal forma que, negligenciar os limites estabelecidos pelo fabricante do equipamento pode acarretar em danos. Foi com base neste princípio que se pôde desenvolver estudos e estabelecer modelos de transmissão e distribuição de energia elétrica, que tem por finalidade transferir a potência elétrica da fonte geradora até a carga demandante, o consumidor final (Alexander; Sadiku, 2013). No Brasil, convencionou-se que a energia entregue ao usuário do sistema se daria por meio da corrente alternada, senoidal, com frequência nominal de 60 Hz.

Aritmeticamente, a potência elétrica é dada pelo produto entre tensão elétrica e a corrente absorvidas pela carga,

$$p(t) = v(t) \times i(t), \quad (1)$$

em que $p(t)$ é a potência instantânea absorvida pela carga; $v(t)$ e $i(t)$ representam a tensão instantânea aplicada sobre a carga e a corrente instantânea que passa pela carga, respectivamente.

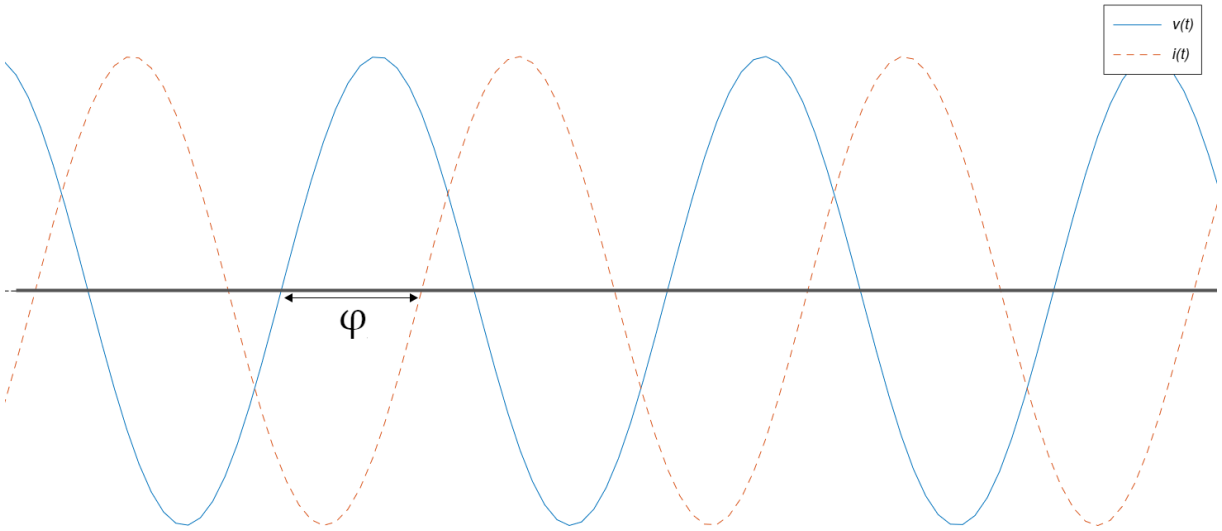
O processo de geração da energia é modelado mecanicamente de forma que o sinal de saída possua características de uma onda senoidal. Assim, as grandezas derivadas desse processo de geração, a saber: tensão e corrente elétrica, também possuirão a mesma característica senoidal e podem ser representadas por:

$$v(t) = V_m cos(\omega t), \quad (2)$$

$$i(t) = I_m cos(\omega t + \varphi), \quad (3)$$

em que V_m e I_m representam a amplitude da tensão e da corrente elétrica, respectivamente; ω é a frequência angular do sinal, dada pelo inverso da frequência elétrica ($\frac{1}{f}$) e medida em rad/s; φ é o ângulo de fase da corrente em relação a tensão, como exemplificado na Figura 1:

Figura 1 - Ângulo de fase do sinal de corrente em relação ao sinal de tensão



Fonte: O autor, 2023.

Substituindo as equações (2) e (3) em (1), obtém-se:

$$p(t) = v(t) \times i(t) = V_m I_m cos(\omega t) cos(\omega t + \varphi) . \quad (4)$$

Aplicando a identidade trigonométrica:

$$cos(A)cos(B) = \frac{1}{2}[cos(A - B) + cos(A + B)], \quad (5)$$

Chega-se às seguintes expressões:

$$p(t) = \frac{1}{2}V_m I_m cos(\varphi) + \frac{1}{2}V_m I_m cos(2\omega t + \varphi) ,$$

$$p(t) = \frac{1}{2}V_m I_m cos(\varphi) + \frac{1}{2}V_m I_m (cos(2\omega t)cos(\varphi) - sen(2\omega t)sen(\varphi)) ,$$

$$p(t) = \frac{1}{2}V_m I_m cos(\varphi) + \frac{1}{2}V_m I_m cos(2\omega t)cos(\varphi) - \frac{1}{2}V_m I_m sen(2\omega t)sen(\varphi) ,$$

$$p(t) = \frac{1}{2}V_m I_m cos(\varphi)[1 + cos(2\omega t)] - \frac{1}{2}V_m I_m sen(2\omega t)sen(\varphi) . \quad (6)$$

Sabendo que a amplitude da tensão (V_m) e da corrente (I_m) podem ser expressos em função dos seus valores eficazes V_{ef} e I_{ef} :

$$V_m = \frac{V_{ef}}{\sqrt{2}}, \tag{7}$$

$$I_m = \frac{I_{ef}}{\sqrt{2}}, \tag{8}$$

e substituindo (7) e (8) em (6), obtém-se:

$$p(t) = V_{ef} I_{ef} cos(\varphi)[1 + cos(2\omega t)] - V_{ef} I_{ef} sen(\varphi) sen(2\omega t)\,, \tag{9}$$

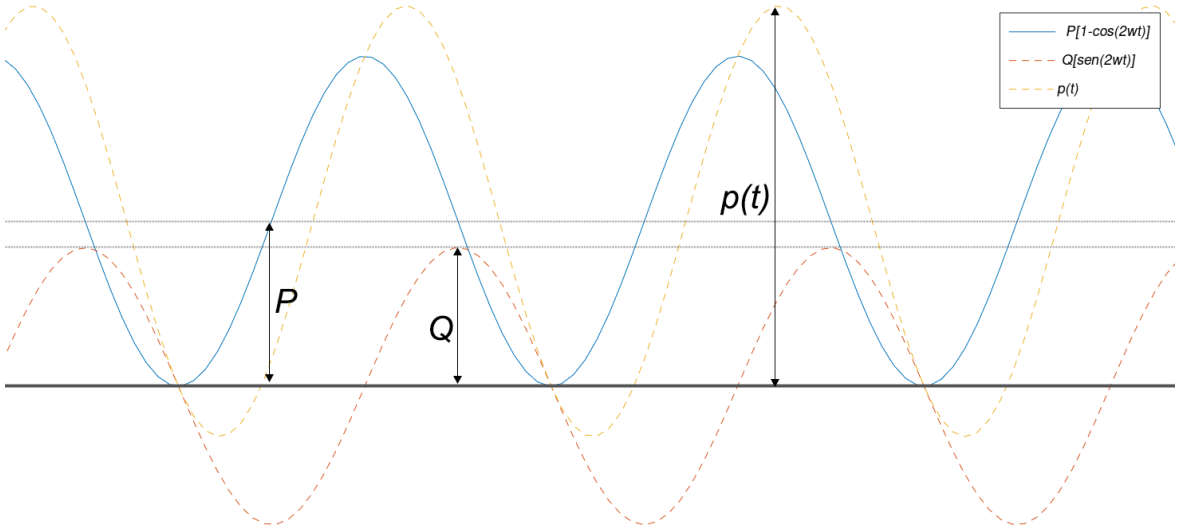
que pode ser reescrita explicitando as parcelas oscilatórias:

$$p(t) = \underbrace{P[1 + cos(2\omega t)]}_{(I)} \underbrace{- Q sen(2\omega t)}_{(II)}\,, \tag{10}$$

em que P é a potência ativa (real) medida em Watt (W), e Q é a potência reativa medida em VAr (Volt-Ampère reativo).

A equação (10) pode ser subdivida em dois termos, (I) e (II). No primeiro termo (I), a potência instantânea oscila com duas vezes a frequência da rede em torno do valor médio P, com amplitude de $2P$. No segundo termo (II), a potência instantânea também oscila com o dobro da frequência da rede, com amplitude igual a Q, mas com valor médio nulo, como representado na Figura 2.

Figura 2 - Decomposição do sinal de potência instantânea



Fonte: O autor, 2023.

Quando em seu semiciclo positivo, a carga drena potência da fonte de energia; enquanto que, em seu semiciclo negativo, a carga transfere potência para a fonte, descarregando seus elementos de armazenamento. Essa característica da potência instantânea sugere que a única parcela da potência que efetivamente é transformada em trabalho é a da potência ativa, tendo em vista que a parcela da potência reativa drenada da fonte, retorna para a fonte a cada semiciclo da potência instantânea, perpetuando um movimento de carregamento e descarregamento dos indutores e capacitores do circuito.

A partir de (9), é possível definir a potência aparente, como descreve:

$$P^2 + Q^2 = [V_{ef}I_{ef}cos(\varphi)]^2 + [V_{ef}I_{ef}sen(\varphi)]^2,$$
$$P^2 + Q^2 = (V_{ef}I_{ef})^2[cos^2(\varphi) + sen^2(\varphi)],$$
$$P^2 + Q^2 = (V_{ef}I_{ef})^2, \qquad (11)$$

em que, o produto entre V_{ef} e I_{ef} é a potência aparente $|S|$, cuja unidade no sistema internacional é VA (Volt-Ampère).

Tomando os sinais de tensão e corrente, $v(t)$ e $i(t)$, respectivamente, em suas formas vetoriais, define-se S como a potência complexa, expressa por:

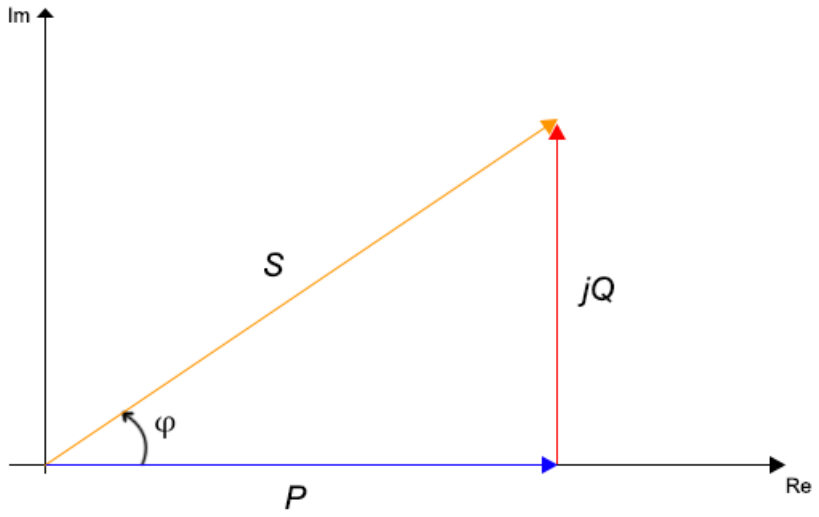
$$S = (V\angle\theta_v)(I\angle -\theta_i) = \underbrace{VIcos(\theta_v - \theta_i)}_{P} + \underbrace{jVIsen(\theta_v - \theta_i)}_{Q}, \qquad (12)$$

em que $V\angle\theta_v$ e $I\angle\theta_i$ são os módulos e ângulos dos números complexos que representam o fasor de tensão e corrente, respectivamente. O módulo da potência complexa é o módulo da potência aparente, e pode ser dada por:

$$|S| = \sqrt{[VIcos(\theta_v - \theta_i)]^2 + [VIsen(\theta_v - \theta_i)]^2}. \qquad (13)$$

O conceito de fator potência pode ser visualizado graficamente por meio do triângulo de potências, como ilustrado na Figura 3:

Figura 3 - Diagrama fasorial da potência aparente, ativa e reativa



Fonte: O autor, 2023.

Neste, a razão entre a potência ativa e o módulo da potência aparente pode ser expressa como:

$$FP = cos(\varphi) = cos(\theta_v - \theta_i) = \frac{P}{|S|} = \frac{P}{\sqrt{P^2+Q^2}}. \quad (14)$$

Esse conceito é utilizado para indicar a eficiência no uso da energia elétrica, ao passo que, quando em seu valor máximo (um), representa a máxima eficiência do sistema. Quando em seu valor mínimo (zero), representa uma baixa eficiência.

O fator de potência está diretamente vinculado à defasagem entre o sinal de tensão (referência) e o sinal de corrente. Estando a corrente em fase com a tensão, têm-se um fator de potência unitário, representando uma carga puramente resistiva. Quando a corrente encontra-se atrasada em relação à referência, o fator de potência é dito indutivo ou atrasado, enquanto que, quando o sinal de corrente encontra-se adiantado em relação à referência, o fator de potência é dito como capacitivo ou adiantado (Alexander; Sadiku, 2013).

4 O SISTEMA DE MONITORAMENTO DE ENERGIA ELÉTRICA

Neste capítulo serão apresentadas as ferramentas utilizadas, características técnicas e funcionalidades para realizar uma comunicação entre um dispositivo de medição e o usuário final, criando um ambiente de monitoramento remoto por meio de uma página da *internet*.

4.1 *HARDWARE*

4.1.1 Microcontrolador (ESP32)

Amplamente implementados no mercado tecnológico, os microcontroladores têm os microprocessadores como peça central para a realização de cálculos e tomada de decisões lógicas. Os microcontroladores são conhecidos por reunirem em um único circuito integrado diferentes dispositivos eletrônicos capazes de processar e transmitir informações com rapidez, eficiência e praticidade. Além do processador, a memória de armazenamento, circuitos de conversão analógico-digital (AD), circuitos PWM (do inglês, *Pulse Width Module*), portas digitais de entrada e saída, e outros, são exemplos de recursos integrados ao microcontrolador que corroboram para a versatilidade e aplicabilidade desta tecnologia (Rios, 2019).

Requisitados para soluções de baixo custo, os microcontroladores das fabricantes *Espressif Systems* e Arduino dominam o mercado por oferecerem produtos que combinam alto poder de processamento e uma vasta gama de recursos capazes de satisfazerem aplicações de simples implementação, como um controle de relés por meio de comandos físicos, até aplicações mais sofisticadas como aquisição, tratamento e monitoramento remoto de dados.

Dentre os microcontroladores, destacam-se os mais populares: ESP32, ESP8266 e Arduino UNO. Em pesquisa realizada utilizando o mecanismo de busca *Google* Acadêmico, constatou-se, segundo o Quadro 1, uma discrepância considerável na quantidade de trabalhos acadêmicos (livros, monografias, artigos, etc.) e citações encontradas pela plataforma.

Quadro 1 - Quantidade de trabalhos acadêmicos localizados na plataforma “Google Acadêmico”

PALAVRA-CHAVE	RESULTADOS
Arduino UNO	279.000
ESP8266	41.900
ESP32	21.300

Fonte: O autor, 2023.

Segundo a ferramenta de busca do *Google*, a plataforma Arduino UNO é majoritariamente detentora do maior número de reproduções acadêmicas encontradas pela plataforma, tendo seis vezes mais conteúdos localizados que o ESP8266 e mais de dez vezes a quantidade de resultados obtidos em comparação ao ESP32. Porém, tal comportamento não se repete quando comparadas suas características técnicas, como: poder de processamento e de aplicabilidade dos três microcontroladores.

Quadro 2 - Comparativo de parâmetros técnicos entre microcontroladores

	ESP32	**ESP8266**	**ARDUINO UNO**
Núcleos	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	240MHz	160MHz	16MHz
Módulo *Wi-fi*	Presente	Presente	Ausente
Bluetooth	Presente	Ausente	Ausente
RAM	512KB	160KB	2KB
Flash	16Mb	16Mb	32Kb
GPIO	36	17	14
ADC	18	1	6
DAC	2	0	0

Fonte: Adaptado de https://www.fernandok.com/2017/11/introducao-ao-esp32.html, 2017.

Enquanto a plataforma ESP32 dispõe de dois núcleos de processamento, os modelos ESP8266 e Arduino UNO contam com um único núcleo, tornando a frequência de processamento desses modelos consideravelmente discrepantes: 240MHz de capacidade máxima para o ESP32, 160MHz para o ESP8266 e apenas 16MHz no Arduino UNO.

Ademais, ambas as plataformas da Espressif Systems contam com módulo integrado de conectividade *Wi-fi* 2.4GHz, possibilitando tanto a conexão da plataforma com a rede local quanto a criação de um *AccessPoint*. Na plataforma Arduino, é necessário adquirir um módulo externo específico para se obter tais funcionalidades, tendo em vista que ela não possui este módulo integrado.

O comparativo entre os *datasheets* de ambos os modelos é unânime, que a plataforma ESP32 é a tecnologia mais promissora da categoria, contando com maior espaço de memória volátil de leitura e escrita, maior número de entradas digitais e analógicas, etc. Diante do exposto, para este projeto, adotou-se a versão ESP-WROOM-32 da fabricante Espressif Systems como microcontrolador responsável pela manipulação e envio dos dados de interesse ao banco de dados.

4.2 SOFTWARE

4.2.1 A Linguagem de Programação

A lógica de desenvolvimento desta aplicação foi subdividida em duas seções: *server-side* e *client-side*, que podem ser definidas como a programação que é executada do lado do servidor e a que é executada do lado do usuário, respectivamente.

Entende-se como “lado do servidor” toda linha de código que é executada pelo microprocessador e Firebase com a finalidade de receber, processar, tratar e enviar os dados das medições ao banco de dados e o processamento realizado no servidor pelo sistema *web*.

Quanto ao *client-side*, pode-se entender como a parte do código que é executado pela máquina do usuário, com a finalidade única e exclusiva de requisitar, receber e exibir informações para o cliente.

Para que o usuário acompanhe em tempo real as medições, diagnósticos resumidos, e outros utilitários, desenvolveu-se uma página *web* por meio da linguagem de marcação HTML, integrada ao mecanismo de estilização baseado em CSS.

As páginas *web* podem ser classificadas em dois tipos: estáticas e dinâmicas. Estáticas são as páginas geralmente construídas por meio de HTML e estilizadas com CSS, de conteúdo fixo, alocados a posições específicas e com uma estrutura bem definida, não reagindo a interações do usuário com o conteúdo. Já nas páginas dinâmicas, têm-se a influência de uma linguagem de programação que interage com o HTML e com o CSS, atualizando o conteúdo da página conforme a interação do usuário com elementos nela contidos, criando um efeito dinâmico de atualização dos dados (MDN Web Docs, 2023).

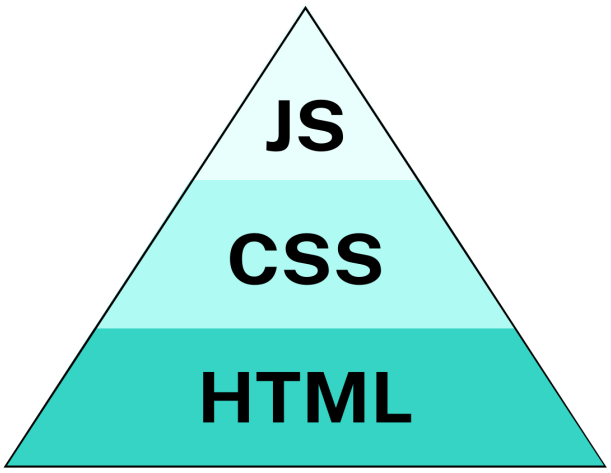
> A palavra dinâmico é usada para descrever tanto o JavaScript *client-side* como o *server-side* — essa palavra se refere a habilidade de atualizar a exibição de uma página web/app para mostrar coisas diferentes em circunstâncias diferentes, gerando novo conteúdo como solicitado. Código do lado do servidor dinamicamente gera novo conteúdo no servidor, puxando dados de um banco de dados, enquanto que JavaScript do lado do cliente dinamicamente gera novo conteúdo dentro do navegador do cliente, como criar uma nova tabela HTML com dados recebidos do servidor e mostrar a tabela em uma página *web* exibida para o usuário (MDN Web Docs, 2023, n.p).

A linguagem escolhida para que a página estática, construída por meio do HTML e CSS, pudesse ser atualizada de forma dinâmica com base nos dados de

medições e requisições do usuário, característica de um *web app*, foi o JS (JavaScript). Tal escolha se deu pelo fato da linguagem ser de alto nível, ou seja, apresentar uma sintaxe de baixa complexidade, proporcionando que implementações complexas sejam feitas com relativa praticidade (MDN Web Docs, 2023).

Destarte, o *front-end,* interpretado pelo *client-side*, pode ser resumido a três camadas estruturadas: HTML, CSS e o JavaScript que interage com a página *web*, como descrito pela Figura 4, sendo o HTML responsável pela estrutura da página, o CSS pela estilização da página e ao JavaScript a função de interagir com o HTML e CSS, criando um efeito dinâmico para a aplicação.

Figura 4 - Composição básica da estrutura *front-end*



Fonte: O autor, 2023.

O *server-side* é constituído pela lógica armazenada na memória do microcontrolador e pelas ferramentas do Firebase.

Com o microcontrolador ESP32 é possível programar em linguagens como: C, C++ e Python, além de desenvolver aplicações em uma variedade de ambientes de programação. Neste projeto utilizaremos o ambiente de desenvolvimento integrado Arduino IDE, a fim de estabelecer a comunicação com o microcontrolador por meio da linguagem C++.

4.2.2 O Firebase

Desenvolvida pelo Google*,* a ferramenta denominada Firebase consiste em um conjunto de serviços multiplataforma disponibilizados pela empresa com a missão de permitir que o desenvolvimento de aplicativos *web* ou móveis se dê de forma mais rápida, efetiva e simples.

O Firebase se propõe estar presente em todas as fases do desenvolvimento: desde a criação, implementação e hospedagem, até a distribuição, monitoramento e promoção da aplicação. Com uma interface intuitiva e de aplicabilidade facilitada, o Firebase busca oferecer a solução para qualquer tipo de desenvolvedor.

Na fase de criação e desenvolvimento de aplicativos, o Firebase traz poderosas ferramentas para suprir as necessidades inerentes a uma infraestrutura *back-end*, dispensando o gerenciamento de múltiplos servidores, por exemplo*.*

Dentre os diversos produtos oferecidos, destacam-se: *Realtime Database, Cloud Firestore, Authentication*, *Hosting* e *Cloud Storage*. O *Realtime Database* e o *Cloud Firestore* são soluções em banco de dados baseados no armazenamento em nuvem. Ambas as soluções possuem planos gratuitos de armazenamento e oferecem suporte a atualização em tempo real, diferenciando-se pelas características dos recursos. A desenvolvedora recomenda o *Realtime Database* para aplicações que exigem uma menor latência e que serão alvo de consultas básicas, enquanto que o *Cloud Firestore* é mais recomendado para aplicações que exigem uma maior capacidade de armazenamento e que são alvo de um grande fluxo consultas, classificações e transações avançadas.

A plataforma inova ao desenvolver um mecanismo de sincronização instantânea para seus bancos de dados, os chamados *triggers*, em que todos os dispositivos conectados podem receber em tempo real quaisquer atualizações feitas no banco de dados, sem que declaradamente haja uma estrutura de requisição implementada no *client-side*. Além disso, o Firebase já disponibiliza uma *Application Programming Interface* (API) que dispensa o processo *back-end* para as transações entre o cliente e os seus serviços.

No quesito segurança, é utilizada a linguagem declarativa, desenvolvida e utilizada pelo Google, para restringir o acesso às informações com base nos dados de identidade e nos padrões e correspondências entre elas. Além disso, o Firebase conta com um sofisticado sistema de autenticação multiplataforma de usuários, facilitando o processo de *login* para o usuário e incentivando o aumento no número

de inscrições na aplicação, ao passo que permite que a verificação de identidade ocorra por meio de email e senha, autenticação por conta *Google*, telefone, *Facebook* e outros.

Para hospedagem é oferecido o Firebase *Hosting*, recurso disponível para aplicações *web* que se integra ao ambiente de desenvolvimento local por meio do Firebase CLI (do inglês, *Command Line Interface*). Com o *Hosting* é possível colocar a aplicação *online* e disponibilizá-la de forma gratuita por meio de um *link* com domínio gratuito do Firebase ou vincular a página a um domínio próprio e personalizado.

Referente aos planos de uso, a Firebase disponibiliza duas assinaturas: o Plano *Spark* e o Plano *Blaze*. No Plano *Spark* é possível utilizar de forma gratuita as ferramentas da plataforma dentro dos limites estabelecidos, discriminados no Quadro 3.

Quadro 3 - Capacidade dos serviços Firebase no plano gratuito

Planos de preços - Firebase		
Plano Spark (gratuito)		
Realtime Database	Conexões simultâneas	100 conexões
Realtime Database	Armazenamento	1Gb
Realtime Database	Transferências	10Gb/mês
Authentication	Sem custos financeiros	
Cloud Messaging (FCM)	Sem custos financeiros	
Crashlytics	Sem custos financeiros	
In-App Messaging	Sem custos financeiros	
Monitoramento de desempenho	Sem custos financeiros	

Fonte: Adaptado de Planos de Preços https://firebase.google.com/pricing?hl=pt-br, 2023.

4.2.3 O protocolo MQTT

O MQTT é um protocolo de mensagens cliente-servidor, de código aberto e de estrutura leve, apropriado para a comunicação M2M (máquina-máquina) e para o contexto IoT (Oasis Open, 2014). Para Conceição e Costa (2018, p. 4): “Do ponto de vista da infraestrutura de comunicação, o MQTT pode ser implementado sobre qualquer tecnologia”. Deste ponto de vista, infere-se que vários dispositivos podem se comunicar por meio de uma estrutura cabeada, por *bluetooth*, ou até mesmo por

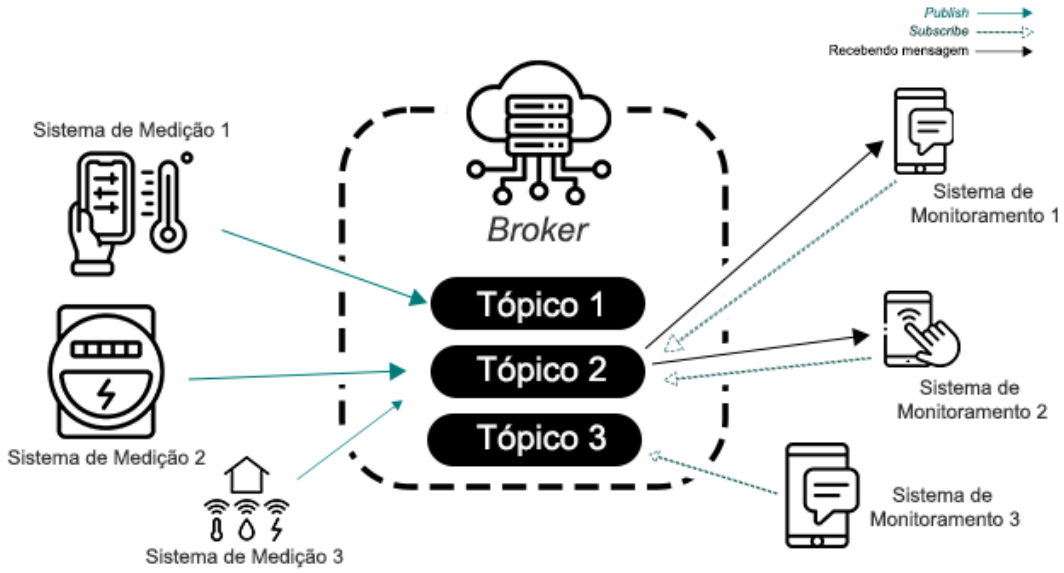
intermédio de redes sem fio utilizando este protocolo. A arquitetura da comunicação MQTT é baseada no paradigma *Publish-Subscriber*, em que mensagens são transmitidas de forma intermitente do emissor para uma lista de clientes inscritos em um tópico (Quincozes; Tubino; Kazienko, 2019). Neste arranjo, o dispositivo remetente envia (*publish*) uma mensagem ao *broker* (servidor MQTT) que, por sua vez, retransmite a mensagem a todos os inscritos (*subscribers*) do tópico a que a mensagem está endereçada.

A partir de um único *broker* é possível conectar múltiplos dispositivos remetentes e receptores para compartilhar informações diversas. Tal fenômeno é possível porque o *broker* pode conter simultaneamente mais de um tópico, gerenciando separadamente os clientes de acordo com o interesse do dispositivo cliente (Quincozes; Tubino; Kazienko, 2019).

Na Figura 5 é ilustrado um cenário hipotético com três sistemas de medição e três dispositivos de monitoramento conectados por meio de comunicação MQTT. O Sistema de Medição 1 envia sua leitura mediante o comando *publish* endereçado ao Tópico 1 do *broker*. Os Sistemas de Medição 2 e 3 encaminham suas medições ao Tópico 2. Paralelamente, os Dispositivos de Monitoramento 1 e 2 solicitam ao *broker* assinatura ao Tópico 2, enquanto que o Dispositivo de Monitoramento 3 solicita a assinatura ao Tópico 3. Finalizado o credenciamento dos dispositivos, o *broker* passa a transmitir todas as mensagens endereçadas ao Tópico 2 para os Dispositivos 1 e 2, enquanto que as mensagens do Tópico 3 são encaminhadas ao Dispositivo 3.

Na pilha de protocolos adotados para as aplicações IoT, o MQTT está na camada de aplicação, sendo responsável por fornecer com alta confiabilidade e qualidade a entrega da informação solicitada pelo dispositivo final. Para tal, geralmente é adotado o *Transmission Control Protocol* (TCP) como camada subjacente, garantindo a entrega de um fluxo de bytes ordenados e sem perdas (Al-Fuqaha *et al.*, 2015).

Figura 5 - Exemplo da estrutura de um sistema de comunicação MQTT



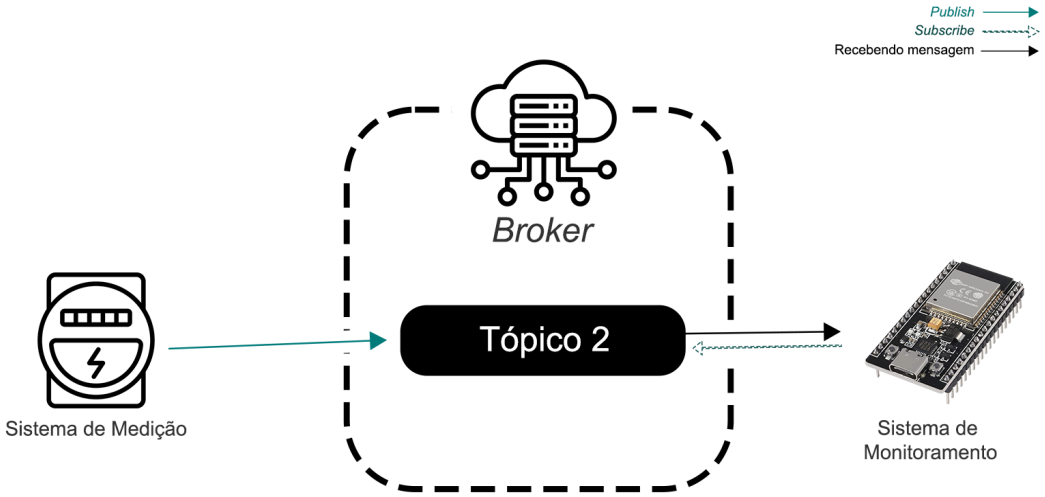
Fonte: O autor, 2023.

Uma forma de implementar o *broker* é por meio da ferramenta Mosquitto, pacote de serviço MQTT da *Eclipse Foundation*. A hospedagem desse servidor pode ser feita em dispositivo físico, permitindo que clientes conectados se comuniquem por meio de mensagens ou pode-se utilizar gratuitamente serviços de *broker* hospedados em nuvem, como o da *Eclipse IoT*, possibilitando que a comunicação ocorra entre quaisquer dispositivos conectados à *internet* (Quincozes; Tubino; Kazienko, 2019).

4.3 A INTERCONECTIVIDADE

A concepção deste projeto parte do pressuposto de que, estabelecida a comunicação por meio de protocolo MQTT entre o sistema de medição e este sistema de monitoramento, como demonstra a Figura 6, transmita-se em tempo real parâmetros da rede elétrica, como: corrente, tensão e defasagem elétrica, além de potência ativa, potência reativa e potência aparente, por meio do microcontrolador ESP32, utilizando o provedor MQTT gratuito desenvolvido e disponibilizado na *internet* pela *Eclipse IoT*.

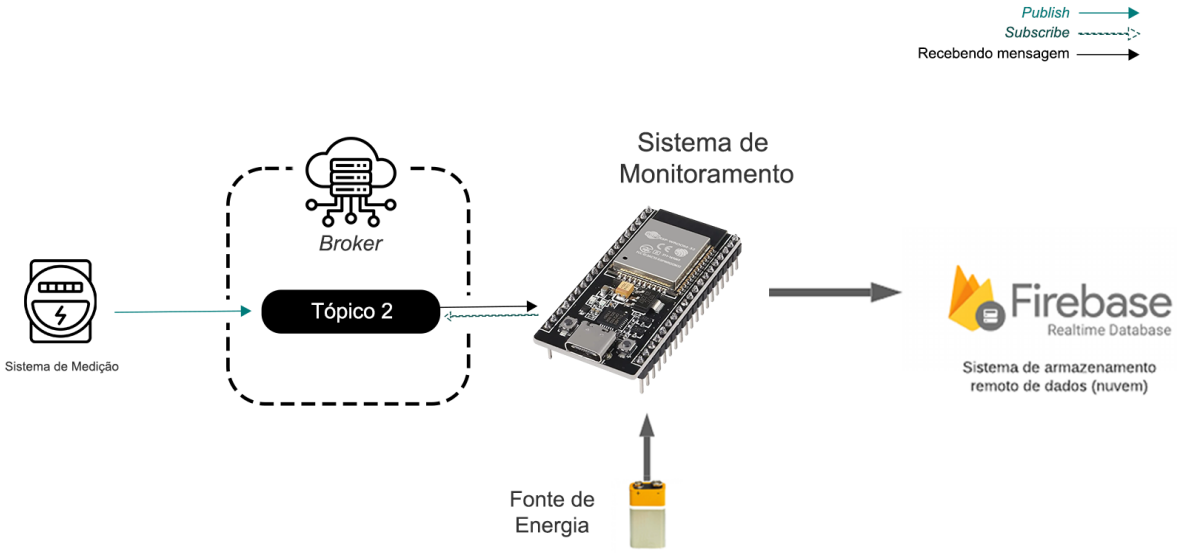
Figura 6 - Esquematização da comunicação entre o Sistema de Medição e o ESP32 via MQTT



Fonte: O autor, 2023.

O banco de dados *Realtime Database,* do Firebase, foi utilizado como meio de armazenamento das informações, como esquematizado pela Figura 7.

Figura 7 - Esquematização da comunicação do Sistema de Medição ao *Realtime Database*



Fonte: O autor, 2023.

O armazenamento dos dados se deu em uma estrutura de árvore denominada formato JSON, em que cada medição recebida é estruturada em uma ramificação e catalogada com nome de identificação única e cronológica gerada pelo método .*push* do próprio Firebase, como demonstrado pelo exemplo subsequente:

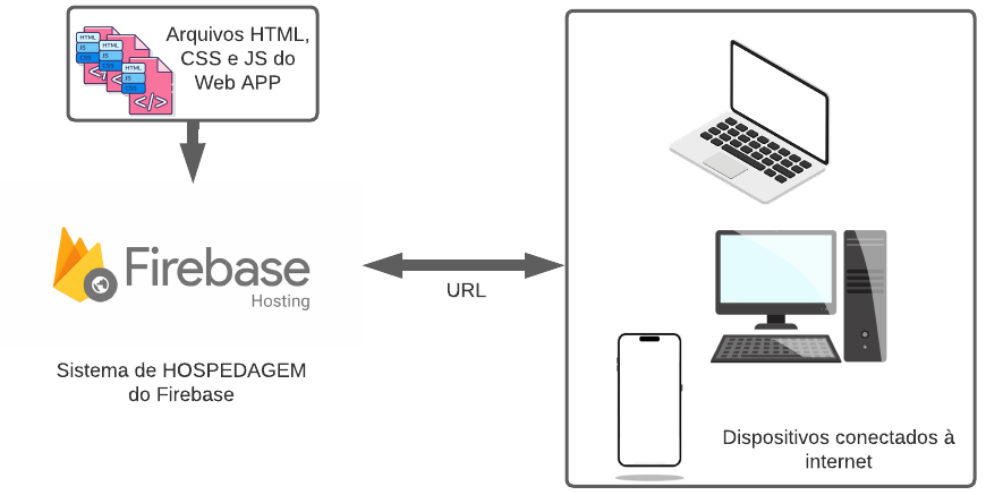
{

```
  "-Nbum23YC5YB2srNe9a8": {
    "Aef": 5.80,
    "Corrente": 8.20,
    "FP": 0.96,
    "PotenciaAparente": 873.47,
    "PotenciaAtiva": 839.64,
    "PotenciaReativa": 240.76,
    "Tensao": 106.41,
    "Timestamp": 1692129569,
    "Vef": 75.24
  },

  "-Nbum2Mk6EdzESkVf8V4": {
    "Aef": 8.48,
    "Corrente": 12,
    "FP": 0.955,
    "PotenciaAparente": 1866.76,
    "PotenciaAtiva": 1785.11,
    "PotenciaReativa": 545.78,
    "Tensao": 155.56,
    "Timestamp": 1692129571,
    "Vef": 110
  },
}
```


Concomitantemente, permitiu-se ao usuário conectar-se à aplicação por meio de *link* fornecido pelo serviço de hospedagem do Firebase *Hosting* (vide Figura 8), no qual se disponibiliza acesso a instruções e a informações do monitoramento de forma remota e em tempo real.

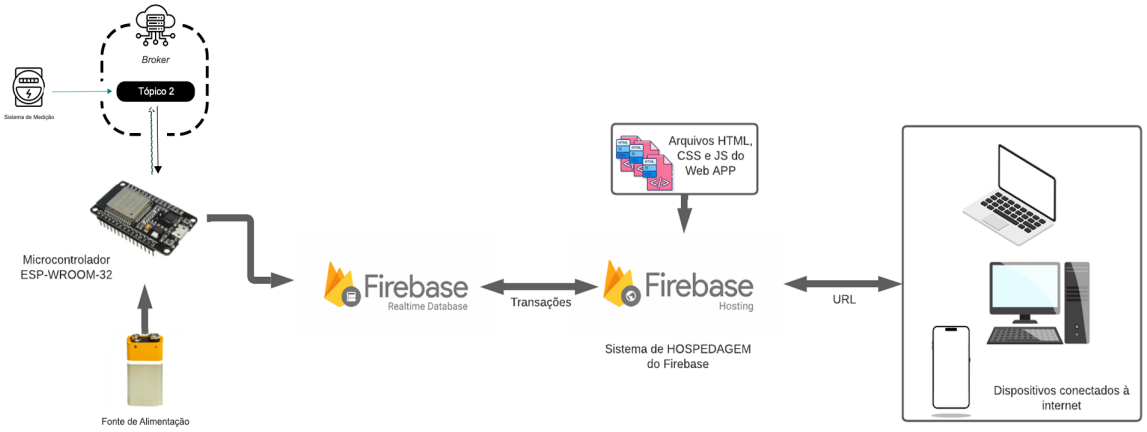
Figura 8 - Esquematização da comunicação entre o cliente e o *webapp*



Fonte: O autor, 2023.

Os dispositivos conectados ao *web app*, denominados adiante de clientes, são capazes de requisitar as informações disponibilizadas pelo servidor a qualquer tempo, a partir do menu de navegação e dos botões de filtragem, que retornam informações exibidas em gráficos específicos para cada grandeza, concretizando a comunicação entre o cliente e o sistema de medição (vide Figura 9).

Figura 9 - Esquematização da comunicação desde o Sistema de Medição ao cliente final.



Fonte: O autor, 2023.

As atualizações das medições para o cliente se dão em tempo real, respondendo à inserção de quaisquer novos dados no banco de dados por meio dos *triggers*. Quando desejar, o cliente também poderá solicitar, em página específica, todas as informações registradas no banco de dados em forma de tabela.

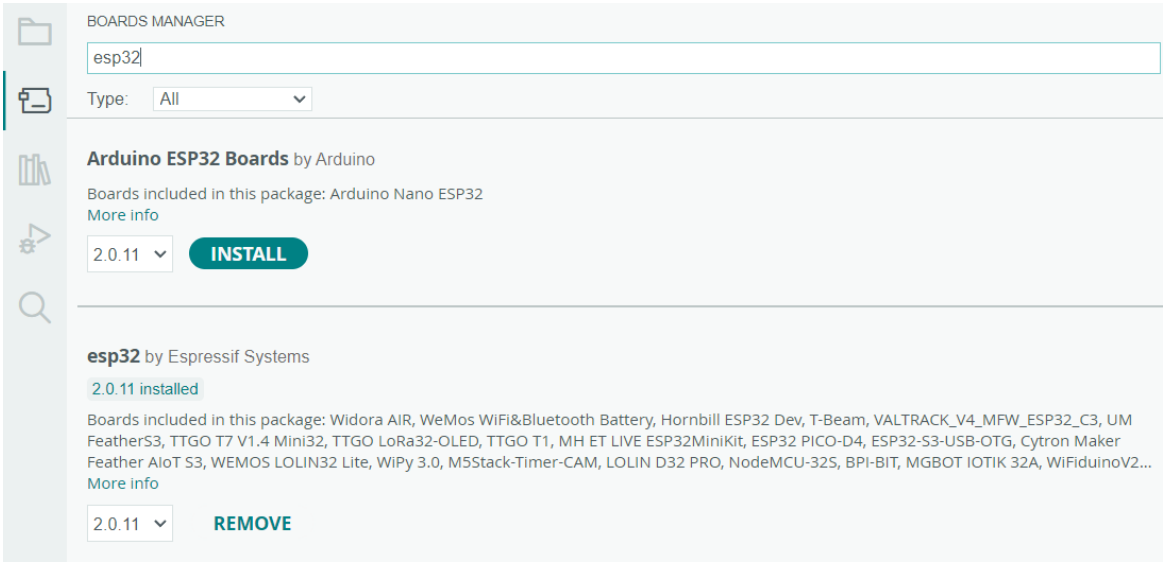
5 DESENVOLVIMENTO E IMPLEMENTAÇÃO DO SISTEMA

5.1 SERVER-SIDE

5.1.1 Arduino IDE e configurações

Para possibilitar a implementação da lógica no microcontrolador ESP32, utilizou-se o ambiente de desenvolvimento integrado Arduino IDE versão 2.1.1, disponibilizado gratuitamente pelo *site* oficial da empresa. Após instalado, se fez necessário informar à IDE qual o dispositivo físico foi utilizado por meio do *Board Manager*, instalando o pacote de comunicação do dispositivo, ou seja, o ESP32. Na Figura 10 é ilustrada a aba *"Board Manager"* do Arduino.

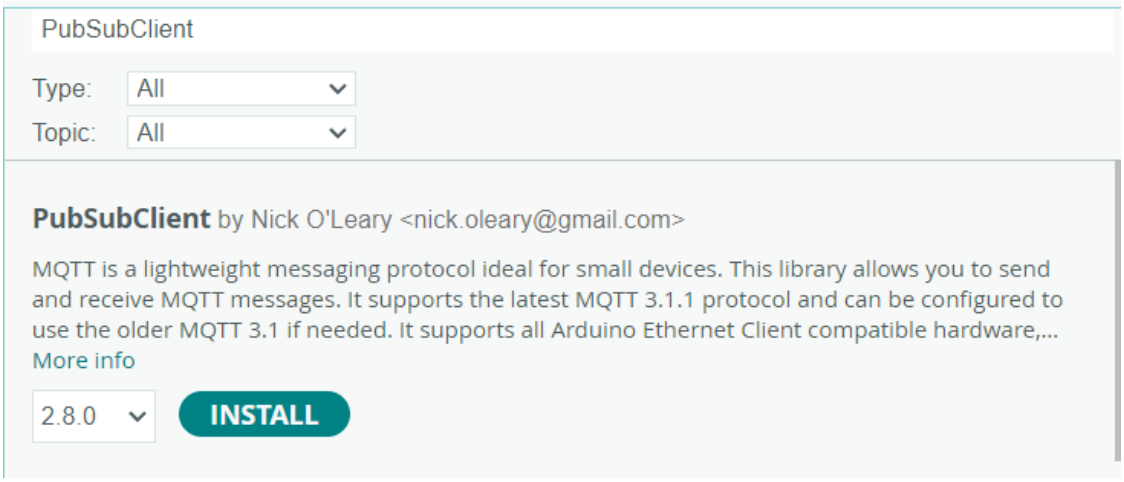
Figura 10 - Instalação do pacote ESP32 no Arduino IDE



Fonte: Aba *Board Manager* do Software Arduino IDE, 2023.

Para estabelecer a comunicação MQTT do microcontrolador foi instalada a biblioteca *PubSubClient* (vide Figura 11), permitindo trabalhar com a versão 3.1.1 do protocolo.

Figura 11 - Instalação da biblioteca *PubSubClient* no Arduino IDE



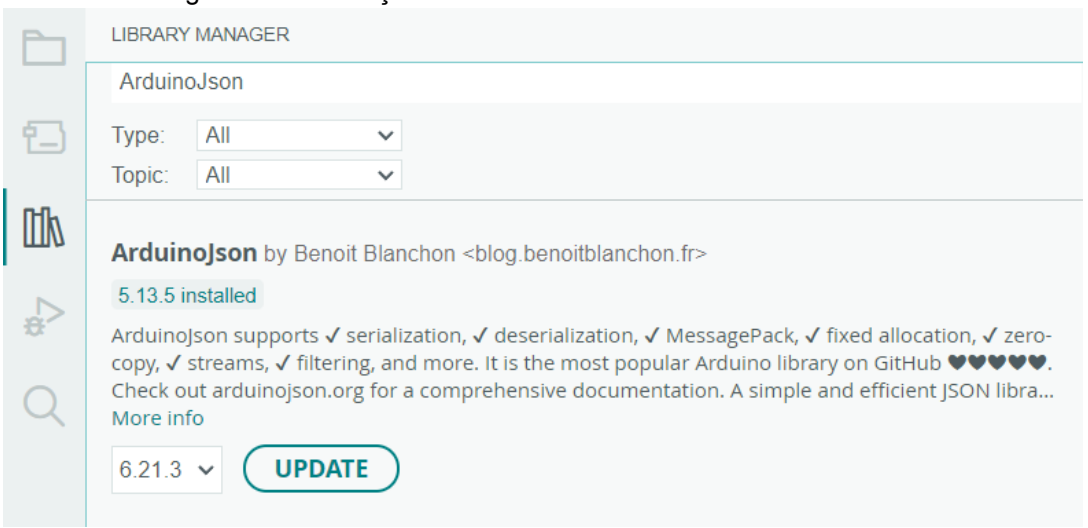
Fonte: Aba *Library Manager* do Software Arduino IDE, 2023.

O *Realtime Database* estrutura os dados em formato JSON, ou seja, o fluxo de informação entre o microcontrolador e o banco de dados se deu por meio dessa mesma estrutura. Para isso, fez-se necessário instalar junto à IDE duas bibliotecas:

a) *ArduinoJson*

O *ArduinoJson* é uma biblioteca desenvolvida para a linguagem C++ aplicada ao Arduino, que possibilitou estruturar e manipular arquivos em formato JSON. Ela foi utilizada para criar, converter e incluir informações das medições numa estrutura de árvore, exportando-as para uma variável específica da memória. Esta biblioteca encontra-se disponível no repositório da própria IDE, o *Library Manager*. Para este projeto foi utilizada a versão 5.13.5 da biblioteca, conforme ilustrado na Figura 12.

Figura 12 - Instalação da biblioteca *ArduinoJson* no Arduino IDE

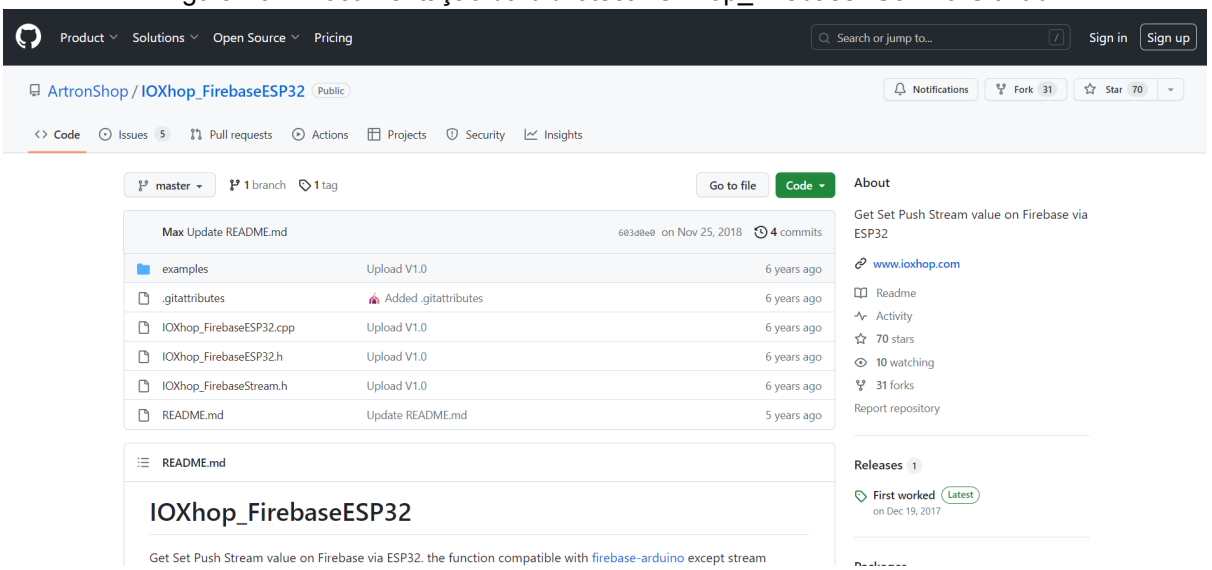


Fonte: Aba *Library Manager* do Software Arduino IDE, 2023.

b) IOXhop_FirebaseESP32

Após a finalização dos processos de tratamento e estruturação das informações recebidas pelo sistema de medição, foi dado início ao processo de envio ao *Realtime Database*. Para tanto, utilizou-se a biblioteca *IOXhop_FirebaseESP32*, que disponibiliza métodos de envio e leitura de informações para o banco de dados do Firebase. Esta biblioteca não encontra-se disponível no repositório do Arduino, portanto, sua instalação se deu de forma manual. Com esse fim, foi baixado o arquivo *Zip* da biblioteca no repositório do *Github* (vide Figura 13).

Figura 13 - Documentação da biblioteca IOXhop_FirebaseES32 no GitHub

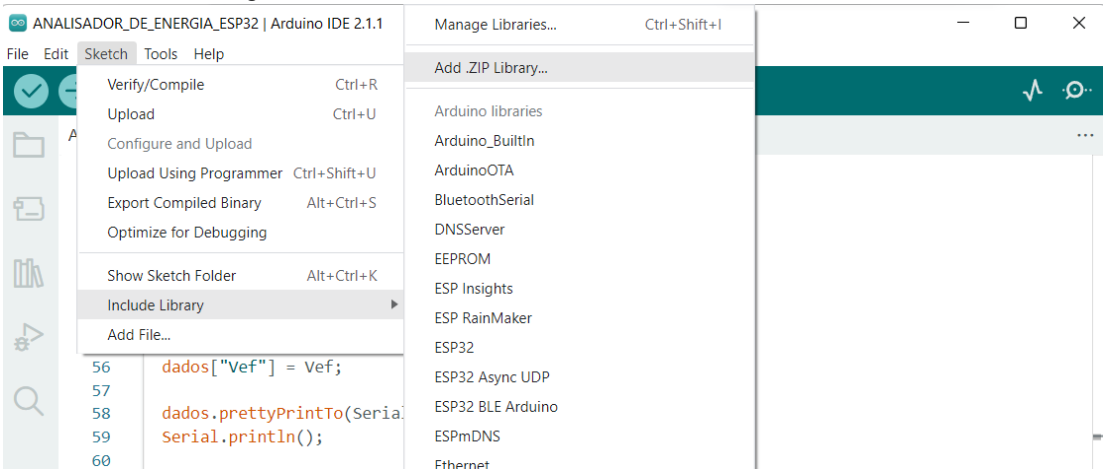


Fonte: Página da biblioteca IOXhop_FirebaseESP32 no GitHub[1], 2023.

A inclusão do arquivo baixado pôde ser feita por meio do menu superior *"Sketch > Include Library > Add .ZIP Library"*, como demonstrado na Figura 14.

[1] Imagem obtida do site https://github.com/ArtronShop/IOXhop_FirebaseESP32.

Figura 14 - Adicionar biblioteca externa no Arduino IDE



Fonte: Menu suspenso *Sketch* do Software Arduino IDE, 2023.

5.1.2 O Projeto do Firebase e o Banco de Dados

Para integrar a plataforma do Google Firebase à aplicação, foi necessário criar um novo projeto. Para tanto, acessou-se o sítio eletrônico do Firebase, realizando o devido processo de identificação e credenciamento, e criando um novo projeto na página do *console*, como mostrado na Figura 15.

Figura 15 - Criação de um novo projeto no Firebase

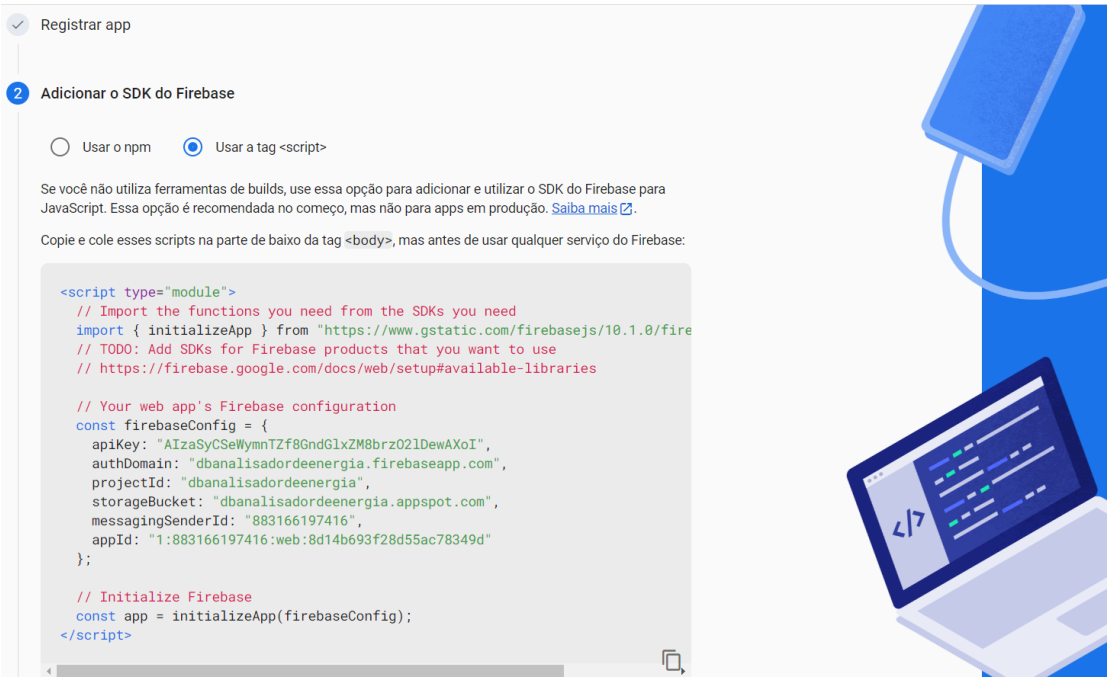
Fonte: Página *Console* do Firebase[2], 2023.

Em seguida, foram configuradas em três etapas as informações básicas do projeto, concedendo as permissões necessárias às ferramentas de análise da plataforma. Posteriormente, foi realizada a vinculação ao Firebase por meio da adição de uma aplicação *web*, configurando as credenciais do projeto, no qual foi

[2] Imagem obtida do site https://console.firebase.google.com.

possível obter o *Content Delivery Network* (CDN) em JavaScript que estabelece a comunicação entre o *web app* e o banco de dados via API, como mostrado na Figura 16.

Figura 16 - *Software Development Kit* (SDK) do Firebase para JavaScript



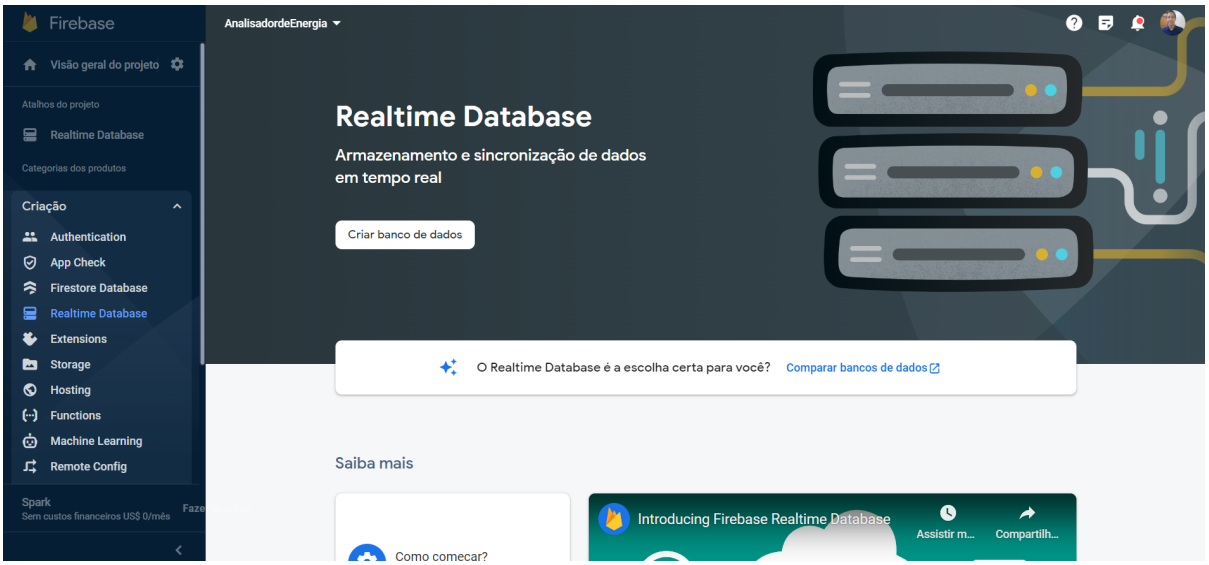
Fonte: Página de configuração SDK do Firebase[3], 2023.

Configuradas as credenciais, foi adicionado o *Realtime Databese* aos serviços do projeto, criando um banco de dados por meio da página de apresentação do recurso. A Figura 17 exibe o painel inicial do *Realtime Database.*

Para fins de desenvolvimento, declararam-se como públicas as regras de segurança do banco de dados, permitindo assim que o microcontrolador e qualquer outra aplicação possa receber/enviar dados livremente do/para o *Realtime Database*. Para tal, as regras "*.read"* e "*.write"* contidos na estrutura *"rules"* do banco de dados foram alteradas de *false* para *true* (Figura 18).

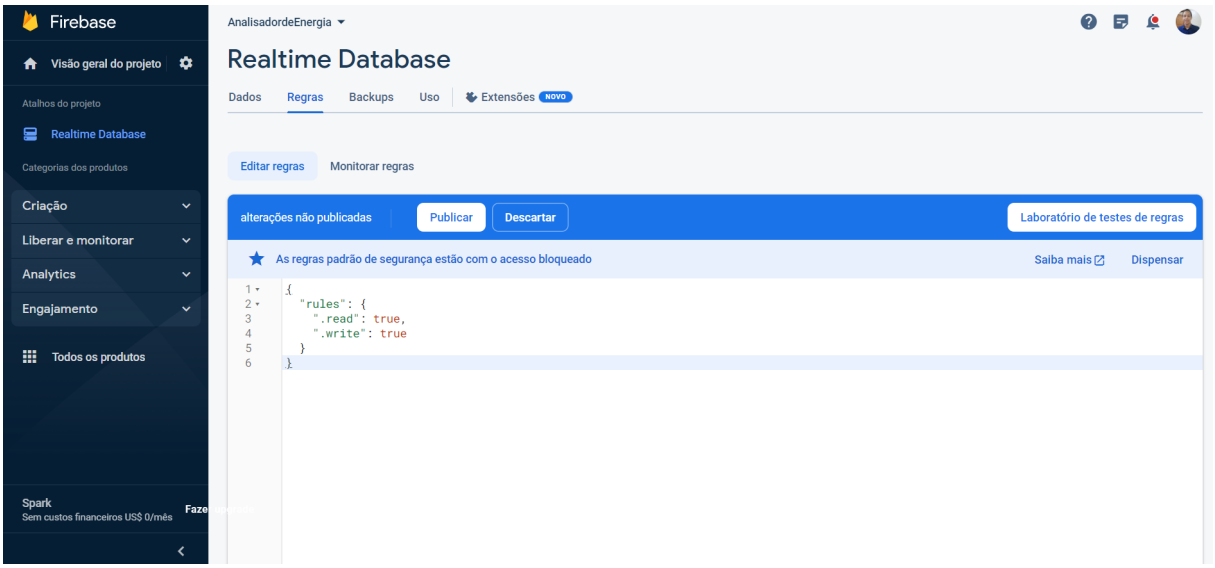
[3] Imagem obtida do site https://console.firebase.google.com.

Figura 17 - Painel inicial do *Realtime Database*



Fonte: Página do *Realtime Database* do Firebase[4], 2023.

Figura 18 - Regras de segurança do banco de dados



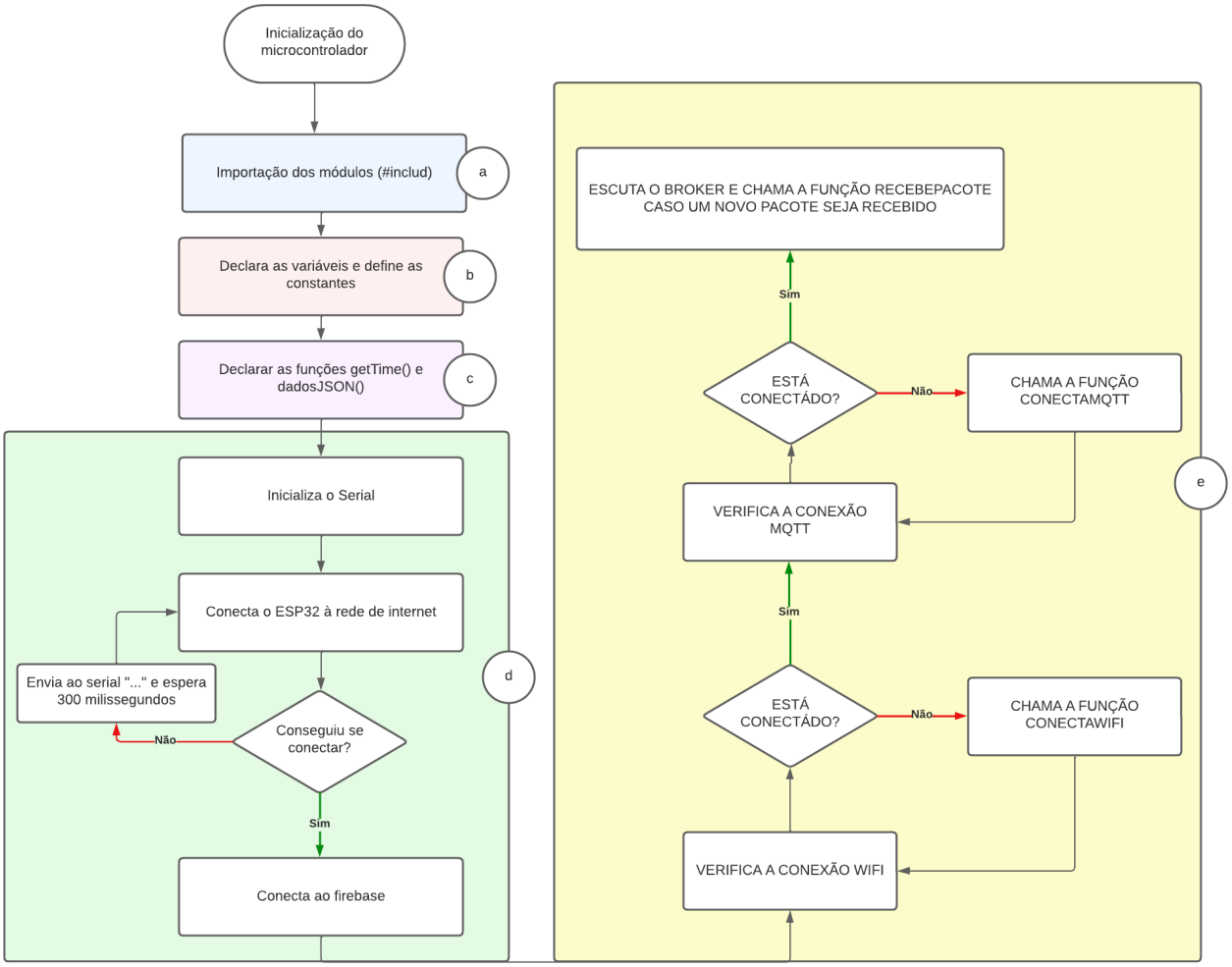
Fonte: Aba regras da página console do *Realtime Database*[5], 2023.

5.1.3 A lógica da programação

O diagrama da Figura 19 esquematiza cronologicamente os passos executados pelo microcontrolador desde sua inicialização até o *loop* de tratamento e envio das informações finais ao banco de dados.

Figura 19 - Rotina da programação implementada no ESP32

[4] Imagem obtida do site https://console.firebase.google.com/project/database.
[5] Imagem obtida do site https://console.firebase.google.com/project/database/rules.



Fonte: O autor, 2023.

Para tornar clara a compreensão, seccionou-se o fluxograma da execução geral em blocos rotulados por letras de “a” a “e”, sendo apresentada uma breve descrição da função do bloco e do respectivo código inerente à etapa:

a) Importação dos módulos:

O bloco de importação dos módulos foi o responsável por incorporar à execução da lógica as propriedades, métodos e funções das bibliotecas:

```
#include <WiFi.h>
#include <IOXhop_FirebaseESP32.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>
```

- *WiFi:* possibilita a conexão ESP32 à rede *wireless* de *internet*;

- *IOXhop_FirebaseESP32:* estabelece a comunicação do microcontrolador com o Firebase e disponibiliza métodos para leitura e escrita no *Realtime Database*;
- *ArduinoJson:* permite a criação e manipulação de dados do tipo JSON;
- *PubSubClient*: estabelece a comunicação MQTT no ESP32;

b) Declara as variáveis e define as constantes:

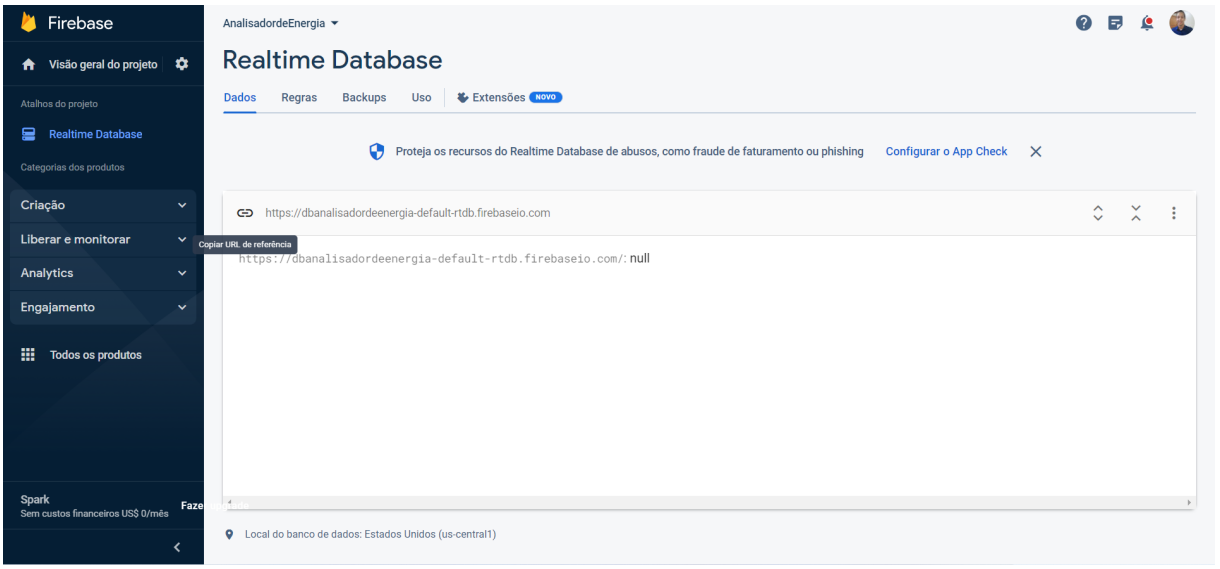
Nesta etapa, definiram-se as constantes *WIFI_SSID* e *WIFI_PASSWORD* que são, respectivamente, *login* e senha da rede *wireless* disponível no local em que o dispositivo será instalado.

```
#define WIFI_SSID "Analisador";
#define WIFI_PASSWORD "senhadowifi";
#define FIREBASE_HOST "https://dbanalisadordeenergia-default-rtdb
.firebaseio.com/"
#define FIREBASE_AUTH "Ir0KVdbNqCSbEKfXXMxVvXxSVtMxO9lu8KdlQiuZ"
```

As constantes *FIREBASE_HOST e FIREBASE_AUTH* dizem respeito ao acesso do banco de dados. A constante *FIREBASE_HOST* é o endereço de referência do banco, estando localizada no título da tabela de dados do *Realtime Database*, conforme demonstrado na Figura 20.

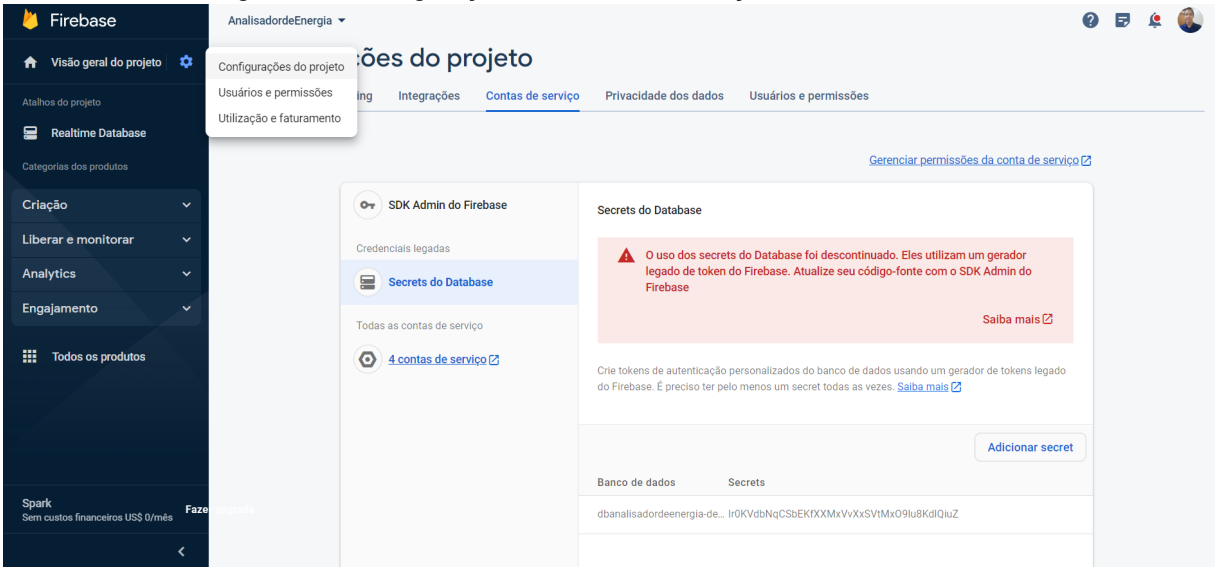
Já a constante *FIREBASE_AUTH* foi encontrada na página *Contas e serviços*, em *Configurações do projeto* > *Contas e serviço* > *Secrets do Database*, como ilustra a Figura 21. O valor atribuído à constante é uma sequência alfanumérica utilizada como chave secreta para acesso ao banco de dados, denominada de *Secrets*.

Figura 20 - Armazenamento de dados do *Realtime Database*



Fonte: Página de dados do *Realtime Database*[6], 2023.

Figura 21 - Configuração de contas de serviço do banco de dados



Fonte: Página “Configurações de projetos” do *Realtime Database*[7], 2023.

As constantes *BROKER* e *PORT* foram utilizadas para armazenar as informações de URL e porta do provedor de serviço MQTT. Em *ID_MQTT* e *TOPIC_SUBSCRIBE* foram guardados o nome do dispositivo e o nome do canal de comunicação que será utilizado, denominado de “tópico”.

```
const char* BROKER = "mqtt.eclipseprojects.io";
int PORT = 1883;
```

[6] Imagem obtida do site https://console.firebase.google.com/projec/database/.
[7] Imagem obtida do site https://console.firebase.google.com/project//settings/serviceaccounts.

```
#define ID_MQTT  "CentralDeMonitoramento"
#define TOPIC_SUBSCRIBE "medicao"
PubSubClient MQTT(wifiClient);
```

c) Declara as funções:

Para manter um fluxo organizado foram definidas quatro funções: *conectaWiFi, conectaMQTT, mantemConexoes* e *recebePacote*.

A função *conectaWiFi* foi implementada para verificar o estado de conexão do ESP32 com a rede de *internet*. Não havendo conexão estabelecida, o microcontrolador envia ao serial uma mensagem de carregamento e faz novas tentativas, até que se efetive, como demonstra o código a seguir:

```
void conectaWiFi() {

  if (WiFi.status() == WL_CONNECTED) {
     return;
  }

  Serial.print("Conectando-se à rede");
  WiFi.begin(SSID, PASSWORD); // Conecta na rede WI-FI
  while (WiFi.status() != WL_CONNECTED) {
      Serial.print(".");
      delay(300);
  }

  Serial.println();
  Serial.print("Conectado com sucesso, na rede: ");
  Serial.print(SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());
}
```

Em *conectaMQTT*, foram estabelecidas as conexões do protocolo MQTT com o provedor do *broker* e com o tópico de comunicação.

```
void conectaMQTT() {
    while (!MQTT.connected()) {
        Serial.print("Conectando ao Broker MQTT: ");
        Serial.println(BROKER);
        if (MQTT.connect(ID_MQTT)) {
            Serial.println("Conectado ao Broker com sucesso!");
```

```
            MQTT.subscribe(TOPIC_SUBSCRIBE);
        }
        else {
            Serial.println("Não foi possível se conectar ao broker.");
            Serial.println("Nova tentativa de conexão em 10s");
            delay(10000);
        }
    }
}
```

A função *mantemConexoes* verifica o *status* de conectividade entre o microcontrolador e o *broker*, e invoca a função *conectaWiFi* para verificar a conexão com a rede *wireless*.

```
void mantemConexoes() {
    if (!MQTT.connected()) {
       conectaMQTT();
    }
    conectaWiFi();
}
```

Por último, a função *recebePacote* será utilizada como *callback* da comunicação MQTT, recebendo as informações pela variável *payload* em formato *byte*. Convertida a mensagem para o formato JSON, por meio do método *.parseObject* da biblioteca *ArduinoJson*, a função finaliza com o envio da medição para o serial e para o *Realtime Database*, por meio do método *.push* da biblioteca *IOXhop_Firebase32*, como descreve o código:

```
void recebePacote(char* topic, byte* payload, unsigned int length){
    DynamicJsonBuffer jBuffer;
    JsonObject& medMQTT = jBuffer.parseObject(payload);
    medMQTT.prettyPrintTo(Serial);
    Firebase.push("/", medMQTT);
}
```

Para que o método *.parseObject* tivesse êxito, foi necessário estabelecer que a informações enviada pelo sistema de medição externo seja do tipo *string* formatada com a sintaxe de um arquivo JSON, como segue o exemplo:

```
{
Medição1: Valor1,
```

```
Medição2: Valor2,
MediçãoN: Valor N
}
```

d) Bloco *Setup*

Definidas as funções, variáveis e constantes, invocou-se a função *setup*, responsável por definir as configurações para operação cíclica do microcontrolador. Após a inicialização do *serial* com o método *.begin*, foi invocada a função *conectaWifi*, estabelecendo assim a conexão do microcontrolador com a rede *wireless* de nome armazenado na constante *WIFI_SSID* com a senha registrada em *WIFI_PASSWORD*. Até que a conexão fosse estabelecida com sucesso, é enviado ao *serial* a mensagem “Conectando ao *wifi*” acrescida de uma cadeia de caracteres “.” (ponto), informando o estágio de processamento da conexão. Estabelecida a conexão com a *internet*, o ESP32 inicializa a comunicação com o Firebase por meio do método *Firebase.begin*, passando como parâmetro o endereçamento de referência do banco de dados e a chave *secrets* de autenticação, conforme:

```
void setup() {
  Serial.begin(115200);
  Serial.println();

  conectaWiFi();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  MQTT.setServer(BROKER, PORT);
  MQTT.setCallback(recebePacote);
}
```

Por fim, as credenciais do *broker* do servidor são passadas para o objeto MQTT, e definida a função *recebePacote* como *callback* da comunicação.

e) Entra na função *loop*:

A palavra “*Loop”* é reservada pela desenvolvedora Arduino à função que é chamada ciclicamente, repetindo a rotina escrita em seu corpo enquanto o dispositivo estiver energizado, permitindo assim um controle ativo do microcontrolador. A rotina de execução do ESP32 foi definida para que as conexões

com a *internet* e com o servidor do *broker* fossem verificadas, restabelecendo-as em caso de falha, e para que a conexão MQTT fosse mantida, invocando a função *recebePacote* sempre que qualquer nova mensagem fosse enviada ao tópico "*medicao"*.

```
void loop() {
  mantemConexoes();
  MQTT.loop();
}
```

5.2 CLIENT-SIDE

5.2.1 O ambiente de desenvolvimento e as configurações iniciais

Para desenvolvimento do código fonte das páginas de monitoramento *web*, foi utilizado o editor da *Microsoft*, o *Visual Studio Code*. A escolha deste ambiente de desenvolvimento se deu pela praticidade em incorporar *plugins* auxiliares como o *Live-Server*, permitindo simular em tempo real as alterações realizadas no código fonte. Também foi necessário instalar o *Node.js*, interpretador de códigos JavaScript fora do navegador, para viabilizar a comunicação com o CLI (do inglês, *Command Line Interface*) do Firebase.

5.2.2 A elaboração da aplicação *web*

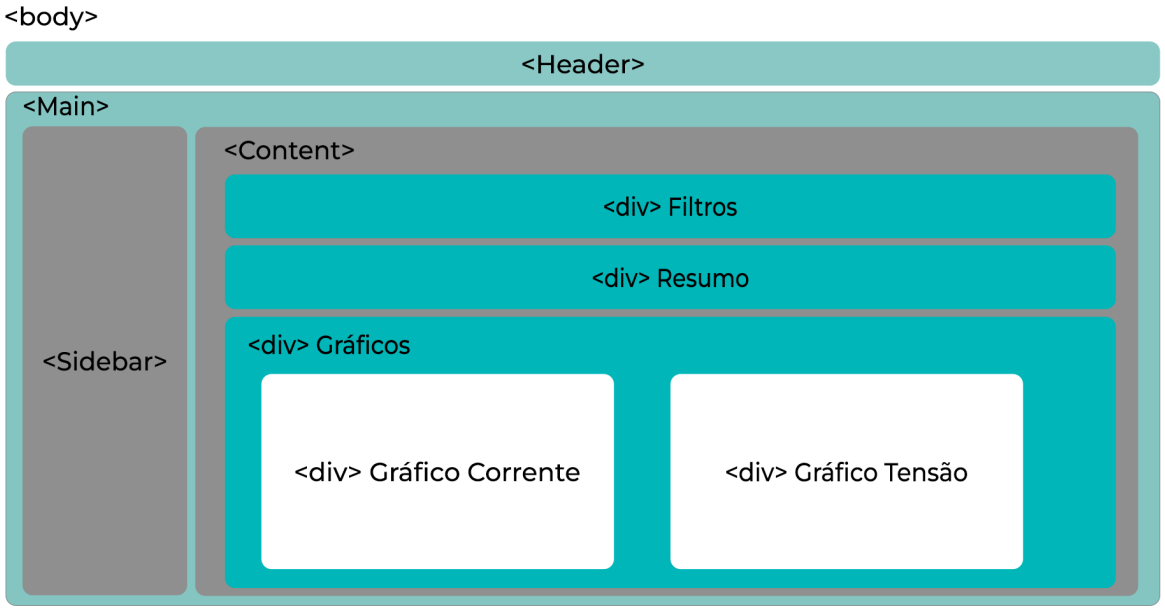
A estruturação do *web app* foi feita para garantir uma experiência visual agradável e acesso facilitado às informações sobre a medição. Por meio das páginas de configurações, buscou-se aprimorar a experiência do usuário, disponibilizando conteúdos informativos, tais como: instruções para interpretar os dados, formas de interagir com o microcontrolador e disponibilizar meios de contato com o suporte técnico.

5.2.2.1 Página Inicial

A página inicial foi desenvolvida para permitir a visualização em tempo real das medições de corrente e tensão elétrica, grandezas que fundamentam as análises de potência e diagnosticam a disponibilidade e a qualidade do fornecimento

de energia elétrica ao ponto de observação. A semântica da página inicial é ilustrada na Figura 22.

Figura 22 - Estrutura semântica da Página Principal



Fonte: O autor, 2023.

Utilizando HTML5, adotou-se como subdivisão padrão do template duas seções: cabeçalho (*tag header*) e corpo (*tag main*), como descreve o código a seguir:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <title>Analisador TCC</title>
      <link rel="stylesheet" href="index.css"/>
  </head>
  <body id="body">
      <header class="topo">
      </header>
      <section class="main">
      </section>
  </body>
</html>
```

O cabeçalho consiste em uma *tag* h4 contendo o título da página, enquanto que no corpo, estruturam-se os blocos de menu (*tag sidebar*) e o bloco de conteúdo (*tag content)*.

No sidebar, implementou-se um menu vertical, dando ao usuário a possibilidade de navegar em dois grupos de opções: Monitoramento e Configurações. O subconjunto de monitoramento dá acesso às páginas início, medições e potências. Já o subconjunto de configurações permite ao usuário navegar entre as páginas de instruções, de parametrização e de suporte. Tal disposição foi implementada pelo código seguinte:

```
<div class="sidebar">
      <div class="menu">
      <h3>Monitoramento</h3>
           <hr>
           <a href="/index.html">Página Inicial</a>
           <a href="/consumo/consumo.html">Consumo</a>
           <a href="/medicoes/medicoes.html">Medições</a>
           <a href="/potencias/potencias.html">Potências</a>
      </div>
     <div class="menu">
            <h3>Configurações</h3>
           <hr>
           <a href="/instrucoes/instrucoes.html">Instruções</a>
           <a href="/parametros/parametros.html">Parâmetros</a>
           <a href="/suporte/suporte.html">Suporte</a>
      </div>
</div>
```

Já na *tag Content*, três *divs* foram criadas, a primeira destinada a filtros, na qual o usuário poderá delimitar as informações gráficas em um intervalo específico de data e hora; na segunda serão exibidas informações resumidas sobre as medições; a terceira e última seção será reservada para os gráficos de tensão e corrente. Para implementar a estrutura descrita, foi criado o seguinte código:

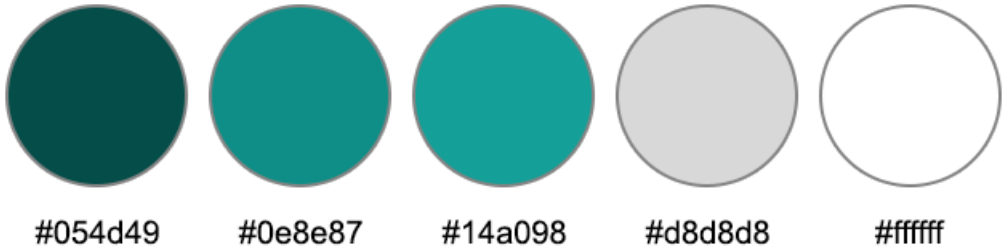
```
<div class="content">
      <div class="filtros">
```

```
                    <h5>Período de observação</h5>
                    <div class="calendario">
                          <p>Início:</p><input type="datetime-local"
class="filtrodata" id="dataInicialFiltro"></input>
                          <p>Fim:</p><input type="datetime-local"
class="filtrodata" id="dataFinalFiltro"></input>
                          <button class="filtrodata btnfiltro"
id="aplicarFiltro">Atualizar Medições</button>
                          <button class="filtrodata btnfiltro"
id="excluirFiltros">Excluir Filtros</button>
                    </div>
              </div>

              <div class="resumomedicoes">
                    <div class="boxmedmax" id="maxTensao"></div>
                    <div class="boxmedmax" id="maxCorrente"></div>
                    <div class="boxmedmax" id="maxPotencia"></div>
                    <div class="boxmedmax" id="consumoMedido"></div>
              </div>
              <div class="graficos">
                    <div class="painelgrafico">
                          <p>Gráfico de Tensão (V)</p>
                          <canvas class="tensao" id="graphTensao"></canvas>
                    </div>
                    <div class="painelgrafico">
                          <p>Gráfico de Corrente (A)</p>
                          <canvas class="corrente"></canvas>
                    </div>
              </div>
        </div>
```

Quanto à estilização da página, explorou-se uma paleta de cores frias, como ilustra a Figura 23, priorizando um ambiente agradável à vista do usuário, sem abrir mão de uma aparência moderna, que explora o contraste entre a cor dos textos e dos elementos.

Figura 23 - Paleta de cores utilizada com os códigos hexadecimais das cores



Fonte: O autor, 2023.

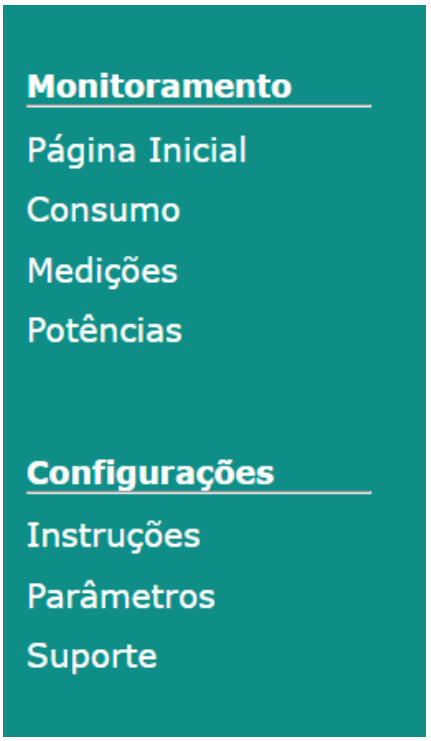
A fim de não limitar a exibição da página *web*, foram adotadas medidas para preservar a responsividade dos elementos. Para isso, determinou-se que as dimensões dos elementos seriam dadas em termos percentuais, passando ao interpretador da página o trabalho de redimensioná-los de acordo com o tamanho disponível para exibição da página. Ao bloco de menu (*sidebar*), por exemplo, foi determinada a largura de 15%, enquanto que para o *content*, a dimensão de 75%, preenchendo, desta forma, a tela do usuário de forma proporcional, como descreve o código CSS:

```
.main{
	display: flex;
	flex-wrap: wrap;
	width: 100%;
	height: 100%;
}
.sidebar{
	background: #0e8e87;
	color: white;
	width: 15%;
	height: 100%;
	padding: 2%;
}
```

Além disso, a altura (*height*) foi definida como 100%, determinando que a altura do elemento se ajuste à altura da tela.

Aos elementos do menu (*sidebar*), foi implementado o *text-decoration: none* e *display: block* para ocultar a estilização padrão dos endereçadores e para posicionar os itens numa disposição vertical, respectivamente, como ilustra a Figura 24:

Figura 24 - Menu de navegação do *web app*



Fonte: O autor, 2023.

Os elementos da seção *content* foram estilizados com bordas arredondadas e sombreamento, de forma a criar uma visualização em *dashboard* com *widgets* flutuantes. Às cores, adotou-se o modelo gradiente para as caixas de resumo das medições, enquanto que aos gráficos se manteve o *background* na cor branca para direcionar a atenção do usuário à informação, conforme o exposto na Figura 25.

Figura 25 - Seção de resumo das medições do *webapp*

Fonte: O autor, 2023.

Destaca-se ainda os elementos da seção gráficos e da seção de resumo das medições, que dispõe seus componentes automaticamente dentro da página por meio da propriedade *display: flex*, alinhando-se horizontalmente com espaçamento uniforme entre eles e a página, por meio da propriedade *justify-content: space-around*, ou priorizando o espaçamento entre os próprios elementos, com a propriedade *justify-content: space-between*, como demonstra-se:

```
.graficos{
```

```
        display: flex;
        justify-content: space-around;
        text-align: center;
        margin-top: 10px;
        }
.painelgrafico{
        width: 49%;
        background: #ffffff;
        border-radius: 10px;
        padding: 10px;
        box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
        }
.resumomedicoes{
        display: flex;
        justify-content: space-between;
        width: 100%;
        height: 60px;
        margin-top: 10px;
        }
```

Para a exibição das medições, utilizou-se a biblioteca *chart-js*, biblioteca *open source* desenvolvida para javascript que possibilita a criação de gráficos modernos e flexíveis (CHART.JS, 2023). Nesse sentido, efetuou-se a importação da CDN (do inglês, *Content Delivery Network*) da biblioteca e de suas dependências, implementado por meio do código JS:

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<script src="https://cdn.jsdelivr.net/npm/luxon"></script>
<script src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon"> </script>
<script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-streaming">
</script>
<script src="https://momentjs.com/downloads/moment.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns/dist/chartjs-adapt
er-date-fns.bundle.min.js"></script>
```

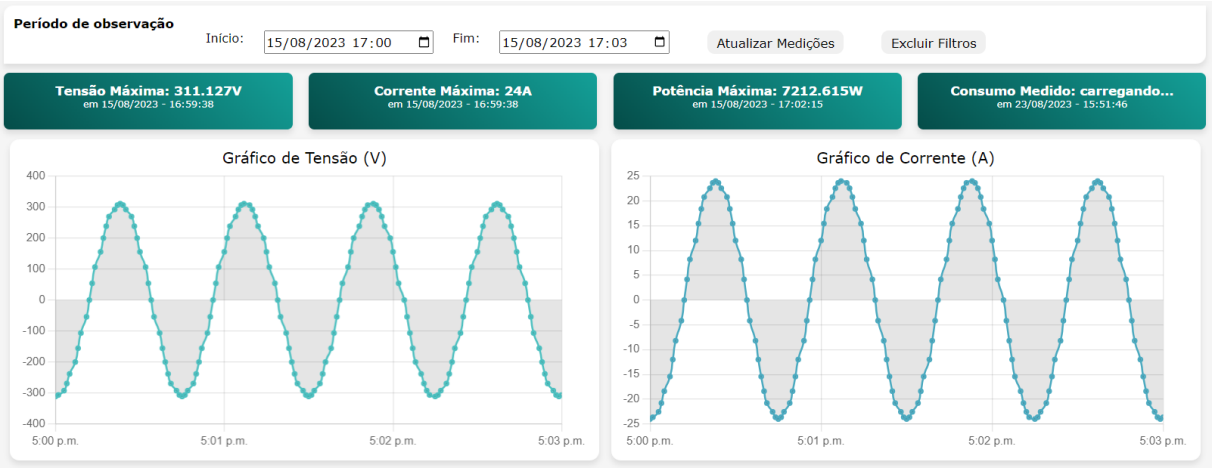
Para a exibição das medições foram implementados dois gráficos de linhas com eixo das abscissas do tipo *realtime*, que permitem ao usuário visualizar a *streaming* das medições em tempo real. O código abaixo descreve a configuração

padrão adotada para ambos, no qual a propriedade *config.plugins.streaming* define a duração e a frequência de atualização das informações exibidas, enquanto que *config.scales.x* define o tipo *realtime* e a quantidade máxima de *labels* na janela:

```
var config = {
      animation: false,
      plugins:{
        legend: false,
        streaming:{
            duration: 100000,
            frameRate: 60}},
      scales: {
         x:{
            type:'realtime',
            ticks: {
                  maxTicksLimit: 5}},
         y: {
            beginAtZero: false}
          }
      }
```

Para permitir que o usuário visualize de forma estática dados referentes a um intervalo específico de tempo, o botão “Atualizar Medições” altera o tipo da propriedade *config.scales.x* para *time*, e adiciona as propriedades *min* e *max* à *config.scales.x*, especificando um valor mínimo e máximo, respectivamente, ao eixo das abscissas, como demonstra a Figura 26 por meio de dados fictícios.

Figura 26 - Visualização gráfica das medições com filtro de datas do *webapp*

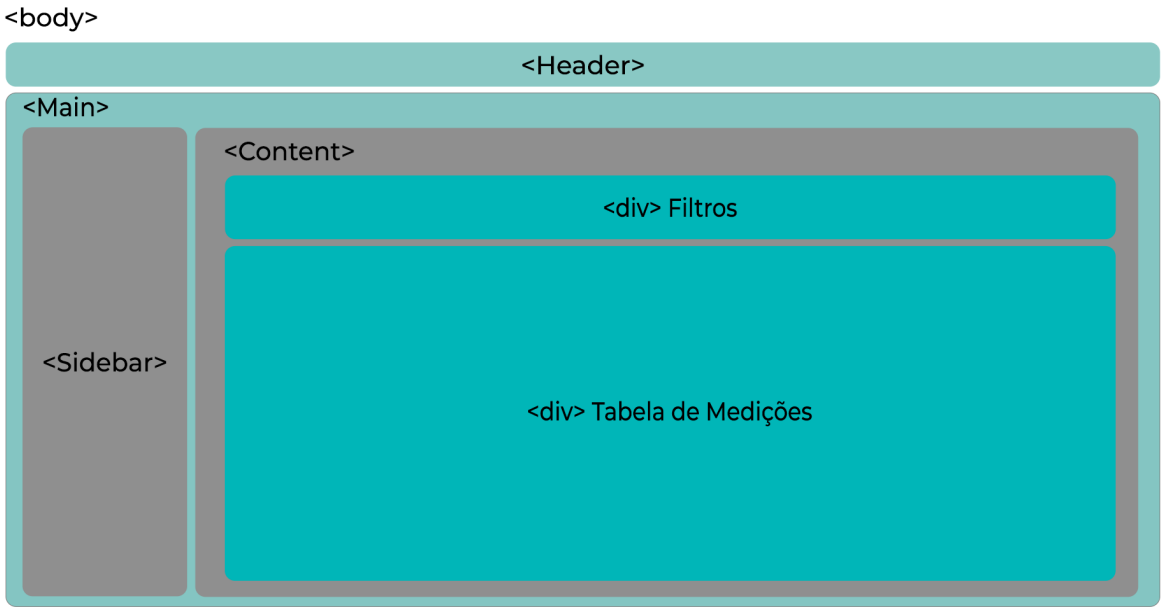


Fonte: O autor, 2023.

5.2.2.2 Página Medições

A página medição foi utilizada para exibir ao usuário uma tabela contendo todas as informações contidas no *Realtime Database*, possibilitando ao usuário detectar eventuais falhas e realizar um diagnóstico assertivo sobre anomalias detectadas por meio da exibição gráfica. A semântica e estilização da página seguem semelhantes à página inicial, substituindo-se o container de gráficos pela tabela de medições, como mostra a Figura 27.

Figura 27 - Estrutura semântica da página “Medições



Fonte: O autor, 2023.

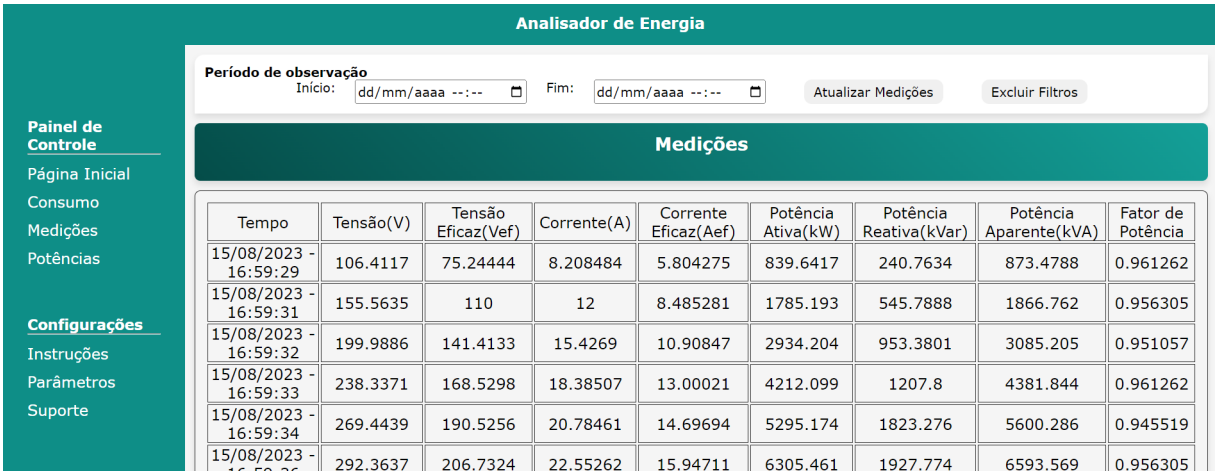
A tabela de medições é estruturada de forma que cada linha represente uma medição instantânea. O código JS que incrementa a tabela segue descrito abaixo:

```
get(banco).then((snapshot)=>{const snap = snapshot.val();
   for (const data in snap){
      if (dataInicialFiltro.value !="" && snap[data].Timestamp*1000 < new
Date(dataInicialFiltro.value).getTime()){
      console.log("Deletado: "+new Date(dataInicialFiltro.value).getTime() );
         continue
      }
        if (dataFinalFiltro.value !="" && snap[data].Timestamp*1000 > new
Date(dataFinalFiltro.value).getTime()){
        console.log("Deletado: "+new Date(dataFinalFiltro.value).getTime() );
          continue
      }
```

```
        tabela += "<tr><td>" + moment(new Date(snap[data].Timestamp*1000)).
format("D/MM/YYYY - HH:mm:ss")+"</td><td>"+snap[data].Tensao+"</td><td>" +
snap[data].Vef+"</td><td>"+snap[data].Corrente+"</td><td>"+snap[data].Aef+"</
td><td>"+snap[data].PotenciaAtiva+"</td><td>"+snap[data].PotenciaReativa+"</t
d><td>"+snap[data].PotenciaAparente+"</td><td>"+snap[data].FP+"</td></tr>";}
        tabelaDados.innerHTML += tabela;
```

O trecho condicional da lógica permite que os filtros de data e hora sejam utilizados para delimitar uma janela de observação, permitindo ao usuário selecionar as medições a partir de uma data mínima, máxima ou por ambos os critérios simultaneamente. Não preenchendo nenhum dos critérios, todas as medições existentes são exibidas (vide Figura 28).

Figura 28 - Página “Medições” do *site* exibindo dados fictícios



Tempo	Tensão(V)	Tensão Eficaz(Vef)	Corrente(A)	Corrente Eficaz(Aef)	Potência Ativa(kW)	Potência Reativa(kVar)	Potência Aparente(kVA)	Fator de Potência
15/08/2023 - 16:59:29	106.4117	75.24444	8.208484	5.804275	839.6417	240.7634	873.4788	0.961262
15/08/2023 - 16:59:31	155.5635	110	12	8.485281	1785.193	545.7888	1866.762	0.956305
15/08/2023 - 16:59:32	199.9886	141.4133	15.4269	10.90847	2934.204	953.3801	3085.205	0.951057
15/08/2023 - 16:59:33	238.3371	168.5298	18.38507	13.00021	4212.099	1207.8	4381.844	0.961262
15/08/2023 - 16:59:34	269.4439	190.5256	20.78461	14.69694	5295.174	1823.276	5600.286	0.945519
15/08/2023 -	292.3637	206.7324	22.55262	15.94711	6305.461	1927.774	6593.569	0.956305

Fonte: O autor, 2023.

5.2.2.3 Página Potências

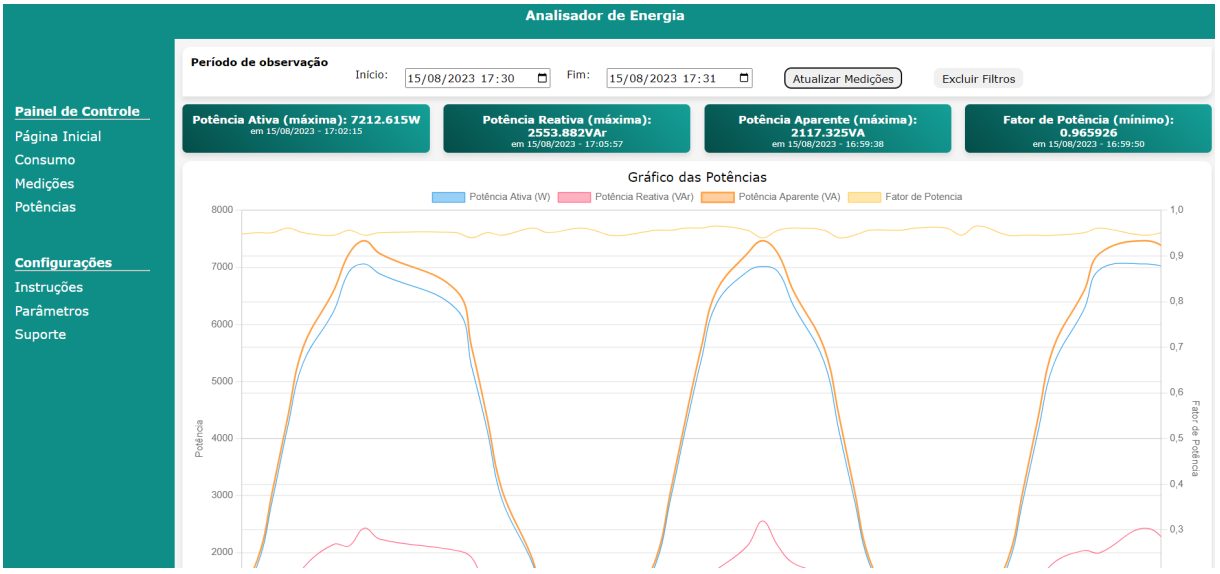
Para uma análise mais detalhada, a página de potências exibe na *tag content* uma marcação para filtros, resumo das medições e outra para gráficos, como demonstra a Figura 29. No bloco resumo, foram selecionadas as informações das máximas medidas de potência ativa, reativa e aparente, além do menor fator de potência, acompanhados pela data e hora ao qual a respectiva medição tenha sido registrada.

Figura 29 - Estrutura semântica da página “Potências”

Fonte: O autor, 2023.

O gráfico das potências exibe, simultaneamente, as curvas de potência ativa (azul), reativa (vermelha), aparente (laranja) e fator de potência (amarela), possibilitando ao usuário ocultá-las por meio da legenda. A escala posicionada à esquerda do usuário, ilustrada na Figura 30, se refere às potências ativa, reativa e aparente, enquanto que, a escala à direita varia de zero à unidade e é pertinente à curva do fator de potência.

Figura 30 - Página “Potências” do *webapp exibindo curvas fictícias*



Fonte: O autor, 2023.

5.2.2.4 As Páginas: instruções, parâmetros e suporte

Nas páginas de instruções foram inseridas orientações básicas acerca do funcionamento do sistema de monitoramento, suas ferramentas e como utilizá-las. Já em parâmetros foram inseridas informações sobre a comunicação MQTT. Em suporte foram disponibilizados meios de contato com o desenvolvedor para eventuais consultas técnicas.

5.2.2.5 A comunicação com o banco de dados

Para estabelecer a comunicação da aplicação *web* com o *Realtime Database*, foi implementado o SDK do projeto, disponibilizado pelo Firebase, no JS da aplicação, a partir do qual foi possível importar os métodos *get, onChildAdded* e outros.

O método *get* foi utilizado para requisitar todos os valores armazenados na estrutura do banco de dados de uma única vez, sendo possível a partir disso, selecionar o conteúdo que será exibido na interface da página. Este método foi implementado na página de medições, para ser chamado sempre que o usuário clicar no botão “Atualizar Medições” e exibir as informações de acordo com os critérios adotados pelos filtros de data e hora, como descreve a Figura 31.

Figura 31 - Filtragem das medições da página medições

Período de observação Início: 15/08/2023 17:30 Fim: 15/08/2023 17:31 Atualizar Medições Excluir Filtros

Medições

Tempo	Tensão(V)	Tensão Eficaz(Vef)	Corrente(A)	Corrente Eficaz(Aef)	Potência Ativa(kW)	Potência Reativa(kVar)	Potência Aparente(kVA)	Fator de Potência
15/08/2023 - 17:30:01	-155.5635	-110	-12	-8.485281	1775.396	576.8612	1866.762	0.951057
15/08/2023 - 17:30:02	-199.9886	-141.4133	-15.4269	-10.90847	2934.204	953.3801	3085.205	0.951057
15/08/2023 - 17:30:03	-238.3371	-168.5298	-18.38507	-13.00021	4212.099	1207.8	4381.844	0.961262
15/08/2023 - 17:30:04	-269.4439	-190.5256	-20.78461	-14.69694	5326.188	1730.582	5600.286	0.951057
15/08/2023 - 17:30:06	-292.3637	-206.7324	-22.55262	-15.94711	6234.342	2146.656	6593.569	0.945519
15/08/2023 - 17:30:07	-306.4003	-216.6577	-23.63539	-16.71274	6925.452	2117.325	7241.889	0.956305
15/08/2023 - 17:30:08	-311.127	-220	-24	-16.97056	7060.233	2431.033	7467.048	0.945519
15/08/2023 - 17:30:09	-306.4003	-216.6577	-23.63539	-16.71274	6887.446	2237.865	7241.889	0.951057
15/08/2023 - 17:30:14	-292.3637	-206.7324	-22.55262	-15.94711	6270.857	2037.524	6593.569	0.951057
15/08/2023 - 17:30:15	-269.4439	-190.5256	-20.78461	-14.69694	5262.547	1915.411	5600.286	0.939693
15/08/2023 -	-238.3371	-168.5298	-18.38507	-13.00021	4167.381	1354.063	4381.844	0.951057

Fonte: O autor, 2023.

Para tornar a *streaming* dos gráficos dinâmica, à página inicial e potências, foi implementado o *trigger onChildAdded* , do SDK, para que, sempre que um novo nó fosse incluído à árvore principal, a função fosse chamada, adicionando aos gráficos um novo par ordenado de medições, diminuindo o volume de informações transferidos do *Realtime Database* para o cliente à cada nova atualização, como descreve o código da página inicial, por exemplo:

```
onChildAdded(banco, (snapshot)=>{
   const data = snapshot.val();
   gerarParOrdenado(data.Timestamp,data.Corrente,gCorrente.data.datasets[0]
.data);
   gerarParOrdenado(data.Timestamp,data.Tensao,gTensao.data.datasets[0]
.data);
```

Dentro do *trigger*, foi chamada a função *gerarParOrdenado* que recebe como parâmetros: tempo, valor e uma lista, os quais dizem respeito ao instante que a medição foi efetuada, ao valor da medição propriamente e a variável que o par ordenado deve ser adicionado, respectivamente. A definição da função foi implementada pelo seguinte código:

```
function gerarParOrdenado(date,valor,listaSaida){
    let dataehora = date * 1000;
    listaSaida.push({x: dataehora, y: valor});
}
```

Por fim, foi utilizado o CLI do Firebase para fazer a hospedagem da aplicação, para tal, utilizou-se o comando "*npm install -g firebase-tools*" no prompt para aceder com a instalação da central de comandos, possibilitando que o *deploy* da aplicação executada por meio do comando "*firebase deploy*". Concluída a hospedagem da aplicação, foi disponibilizado pelo *Hosting* o *link* de acesso para a página *web.*

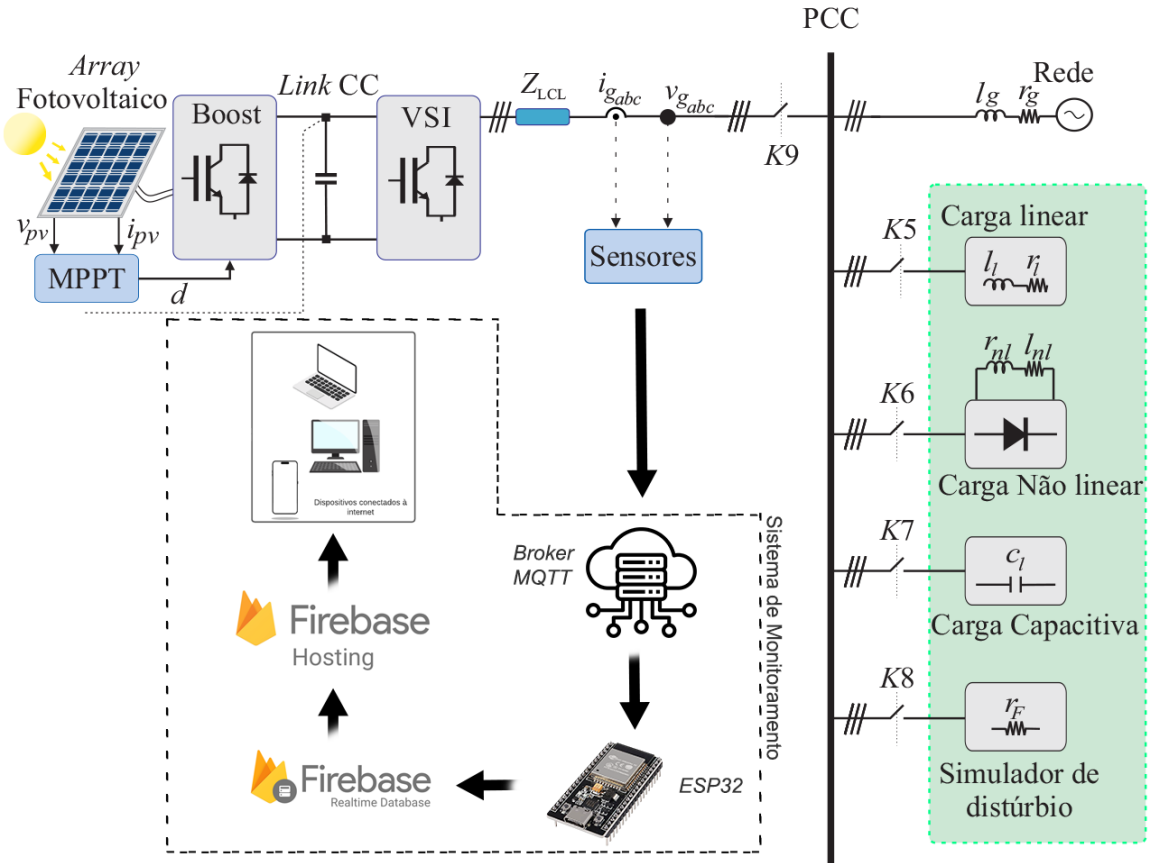
6 SIMULAÇÕES, RESULTADOS E DISCUSSÕES

6.1 PARAMETRIZAÇÃO E CONDIÇÕES INICIAIS

Para simular o funcionamento do sistema de monitoramento, fez-se necessário implementar uma lógica que pudesse reproduzir o envio das medições de forma remota por meio do protocolo MQTT de comunicação. Para tanto, foi utilizada a linguagem de programação Python, versão 3.11.5.

Os dados da medição foram extraídos a partir de Alves (2019) e representam o contexto experimental de uma planta de geração fotovoltaica conectada à rede elétrica, o qual alimenta cargas lineares, não lineares, capacitivas, e um simulador de distúrbios, conforme ilustrado na Figura 32. As medições foram exportadas e armazenadas em um arquivo tipo CSV (do inglês, *Comma-Separated Values*), contendo as amostras de tempo, tensão, corrente, potência ativa, reativa, aparente e fator de potência.

Figura 32 - Contextualização dos dados da simulação



Fonte: Adaptado de Alves, 2019.

A comunicação MQTT foi estabelecida por meio da biblioteca *Eclipse Paho*, informando ao método *connect* de *paho.mqtt* o endereço e a porta do *broker* utilizado pelo sistema de monitoramento, o *Eclipse IoT*.

```
from paho.mqtt import client as mqtt
import time
clientId = "Simulação"
port = 1883
broker = "mqtt.eclipseprojects.io"

client = mqtt.Client(clientId)
client.connect(broker,port)
```

Para mensurar a qualidade do sinal *wireless* de *internet* disponível no local de conexão do ESP32, foram utilizados dois dispositivos: *notebook* e *smartphone*, que realizaram medições simultâneas da qualidade do sinal (vide Figura 33).

Figura 33 - Disposição dos dispositivos durante a simulação

Fonte: O autor, 2023.

Por meio do comando “*netsh wlan show interfaces*” foi possível visualizar as principais propriedades da rede local pelo *notebook*, destacando-se o nível de 87% de sinal *wireless* a uma taxa de transmissão e recepção de 144,4Mbps, conforme ilustrado na Figura 34.

Figura 34 - Retorno da análise de rede feita por meio do Prompt de Comando

```
Nome                   : Wi-Fi
Descrição              : Intel(R) Wi-Fi 6 AX201 160MHz
GUID                   : 9a55b500-9ee5-41bc-bf7f-3fff4f067ad4
Endereço físico        : 14:75:5b:fc:f7:56
Tipo de interface      : Primário
Estado                 : Conectado
SSID                   : Edvanio
BSSID                  : b4:0f:3b:ee:af:d1
Tipo de rede           : Infraestrutura
Tipo de rádio          : 802.11n
Autenticação           : WPA2-Personal
Criptografia           : CCMP
Modo de conexão        : Perfil
Faixa                  : 2,4 GHz
Canal                  : 3
Taxa de recepção (Mbps)    : 144.4
Taxa de transmissão (Mbps)   : 144.4
Sinal                  : 87%
Perfil                 : Edvanio
```

Fonte: Windows 11, 2023.

No segundo teste foi utilizado o aplicativo android *Wifiman*, retornando uma intensidade de sinal de -53 dBm (decibel miliwatt), valor considerado “Excelente” pelo aplicativo, como demonstra a Figura 35.

Figura 35 - Retorno da análise de rede feita por meio de aplicativo Android

Fonte: Aplicativo Android *Wifiman*, 2023.

6.2 SIMULAÇÃO DO SISTEMA

Para analisar o tempo de processamento das instruções de *Callback* da comunicação MQTT e de envio da informação ao Firebase, foram adicionadas funções para exibir no *serial* do Arduino IDE o tempo, em milissegundos, do início e do término de ambas as instruções. Além disso, foram adicionados à lógica do ESP32 instruções para calcular e exibir o tempo de reconexão do microcontrolador com o *broker*, bem como a quantidade de vezes em que houveram desconexões e o número de medições enviadas ao Firebase.

As simulações foram realizadas variando a quantidade de amostras enviadas por segundo, com a finalidade de avaliar a estabilidade e a eficiência da retransmissão das informações pelo ESP32. Para isso, foi utilizado o método *sleep* da biblioteca *time* do Python, visando adicionar um intervalo de milissegundos entre o processamento de cada envio.

```
data = open(r"G:\Documents\UACSA\TCC\SIMULAÇÃO\dados.csv")
a = 0
for valor in data:
        if a != 0:
            dados = valor.replace("\n","").split(",")
            client.publish('medicaodados','''{"Timestamp":'''+'''{:.4f},
"Tensao":{:.2f}, "Corrente":{:.2f}, "Aef":{:.2f}, "FP":{:.1f},
"PotenciaAparente":{:.2f}, "PotenciaAtiva":{:.2f}, "PotenciaReativa":{:.2f},
"Vef":{:.2f}'''.format(time.time(),float(dados[1]),float(dados[2]),float(dado
s[3]),float(dados[4]),float(dados[5]),float(dados[6]),float(dados[7]),float(d
ados[8]))+"}")
        a+=1
        time.sleep(0.0166)
```

6.2.1 Resultados da simulação

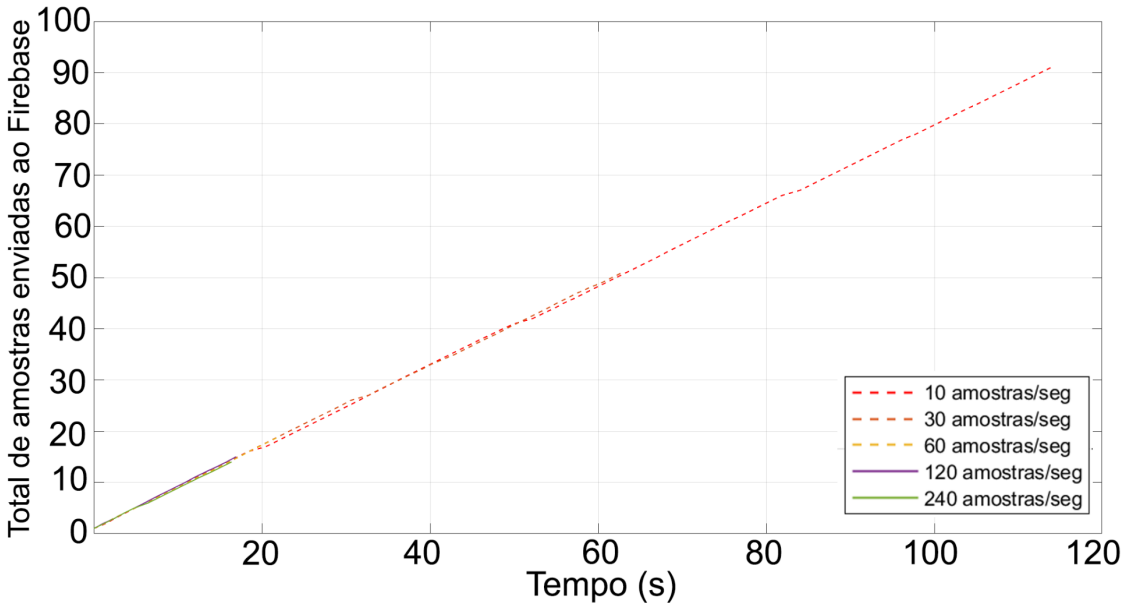
O processo de inicialização do microcontrolador reproduziu na saída *serial* da IDE que o tempo de conexão com o *broker* foi equivalente a 417 ms, conforme:

```
22:53:03.821 -> .......
22:53:05.925 -> Conectado com sucesso, na rede: Edvanio. IP obtido:
192.168.0.116
22:53:05.925 ->
22:53:05.925 -> A conexão MQTT desligada em: 3196ms
22:53:05.925 -> Conectando ao Broker MQTT: mqtt.eclipseprojects.io
22:53:06.333 -> Conectado ao Broker com sucesso!
```

```
22:53:06.333 -> A conexão MQTT ligada em: 3613ms
22:53:06.333 -> A reconexão durou: 417ms
```

A Figura 36 mostra a quantidade de amostras efetivamente enviadas ao Firebase em relação ao tempo de execução a partir da primeira comunicação MQTT recebida pelo microcontrolador. Para uma simulação de envio de 1000 amostras a uma frequência de 10 amostras por segundo, foram efetivamente enviadas 91 medições ao Firebase, resultando em um aproveitamento de 9,1%, sendo este o melhor resultado obtido entre as frequências simuladas.

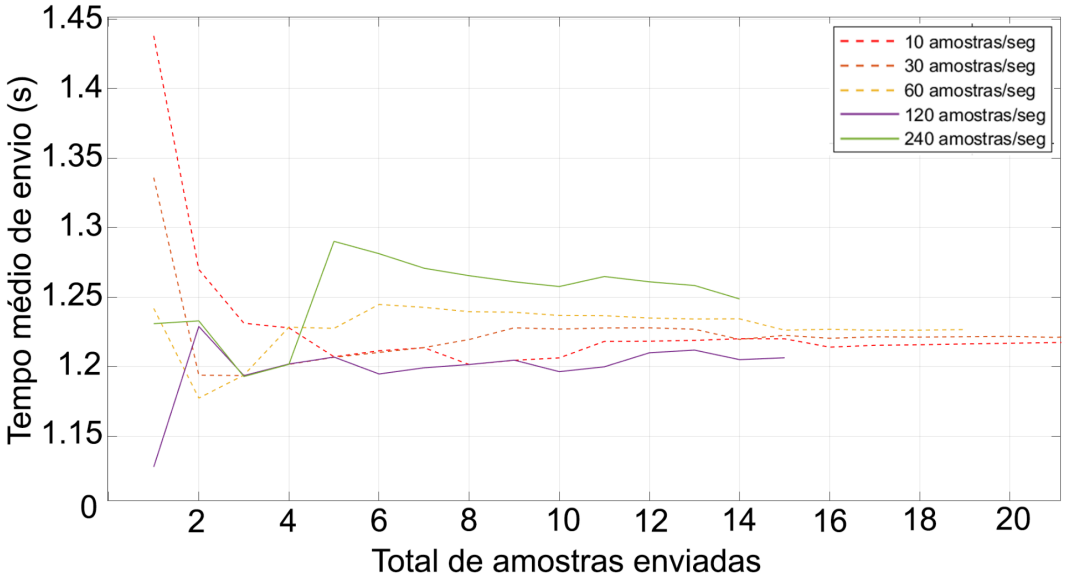
Figura 36 - Relação entre a quantidade de amostras enviadas e o tempo



Fonte: O autor, 2023.

A fim de diagnosticar a baixa eficiência do sistema, investigou-se o intervalo de processamento e envio das medições, constatando-se, a partir do resultado, que o tempo médio oscilou em torno de 1,23 segundos. Tal análise foi realizada com base na observação das curvas apresentadas na Figura 37, que expressam o tempo médio de envio das amostras ao Firebase por frequência de simulação.

Figura 37 - Tempo médio de envio em relação ao total de amostras enviadas



Fonte: O autor, 2023.

Em razão do tempo médio de processamento e envio ao Firebase ser consideravelmente maior do que o tempo médio em que as amostras são recebidas pelo microcontrolador por meio da comunicação MQTT, notou-se a ocorrência de uma sobrecarga no transporte dos pacotes de comunicação, provocando a perda de conexão entre o ESP32 e o *broker* por falta de sincronismo e a perda considerável de pacotes.

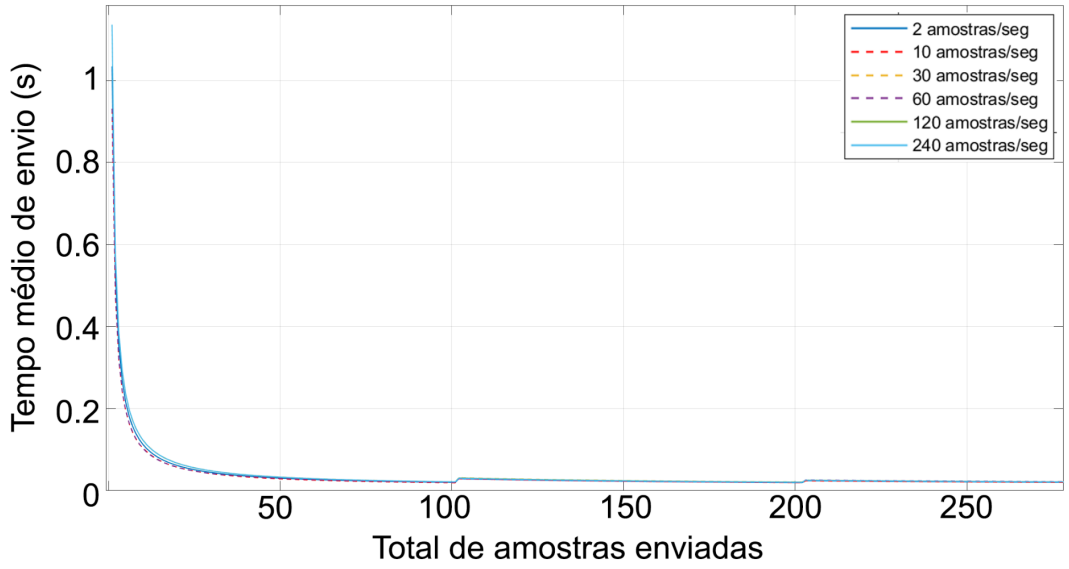
6.2.2 Aprimorando a comunicação com o Firebase

Para diminuir o número de medições perdidas no processo de transmissão ao *Realtime Database*, foi necessário substituir a biblioteca *IOXhop_Firebase* pela biblioteca *FirebaseESP32*. A mudança permitiu que o processamento de envio das amostras fosse realizado de forma assíncrona pelo método *.pushJSONAsync*, não impedindo que novas comunicações do *broker* MQTT fossem recebidas, manipuladas e incluídas em uma fila virtual assíncrona de processamento de envio ao Firebase. Ademais, a própria biblioteca *FirebaseESP32* disponibiliza métodos próprios para manipulação de dados JSON, dispensando a importação da biblioteca *ArduinoJson*.

A Figura 38 retrata o tempo médio de envio ao *Realtime Database* em relação ao número total de amostras enviadas para uma simulação de 5000 amostras. Destaca-se a semelhança entre as curvas para as frequências de 2, 10, 60, 120 e

240 amostras por segundo, demonstrando um tempo médio de envio de 0,021 segundos.

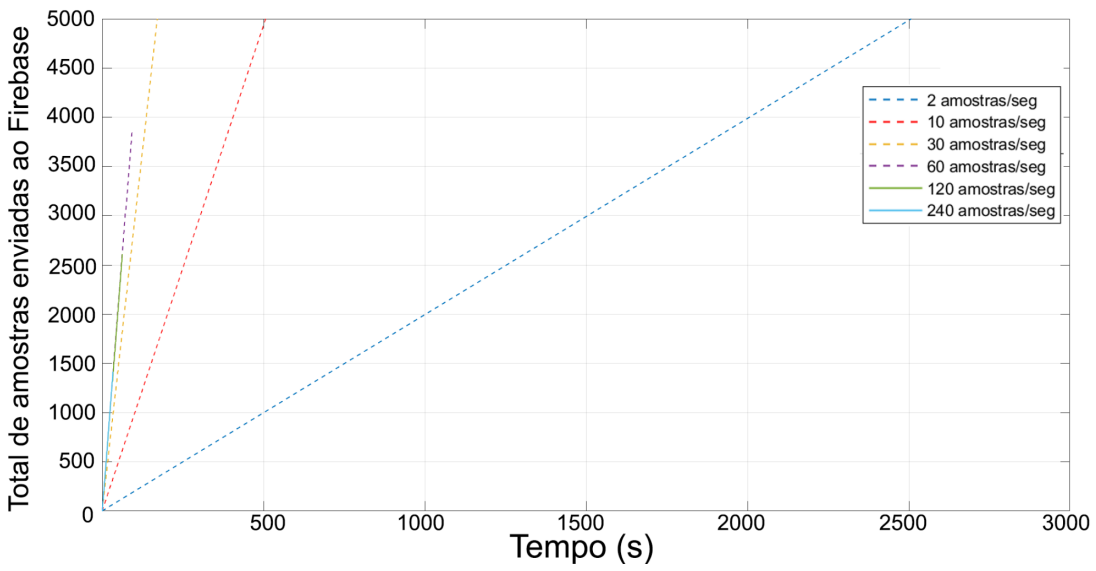
Figura 38 - Relação entre o tempo médio de envio e o total de amostras enviadas



Fonte: O autor, 2023.

Na Figura 39 é ilustrado o total de amostras enviadas ao Firebase em relação ao tempo. Com base nos dados apresentados na Figura 39, houve um aumento significativo na eficiência do sistema de transmissão das informações, observando-se uma entrega efetiva ao Firebase de aproximadamente 99% das 5000 amostras simuladas com frequência de envio de 2, 10 e 30 amostras por segundo. Para as frequências de 60, 120 e 240, o aproveitamento foi de 77,72%, 52,18% e 28,3%, respectivamente.

Figura 39 - Total de amostras enviadas ao Firebase em relação ao tempo

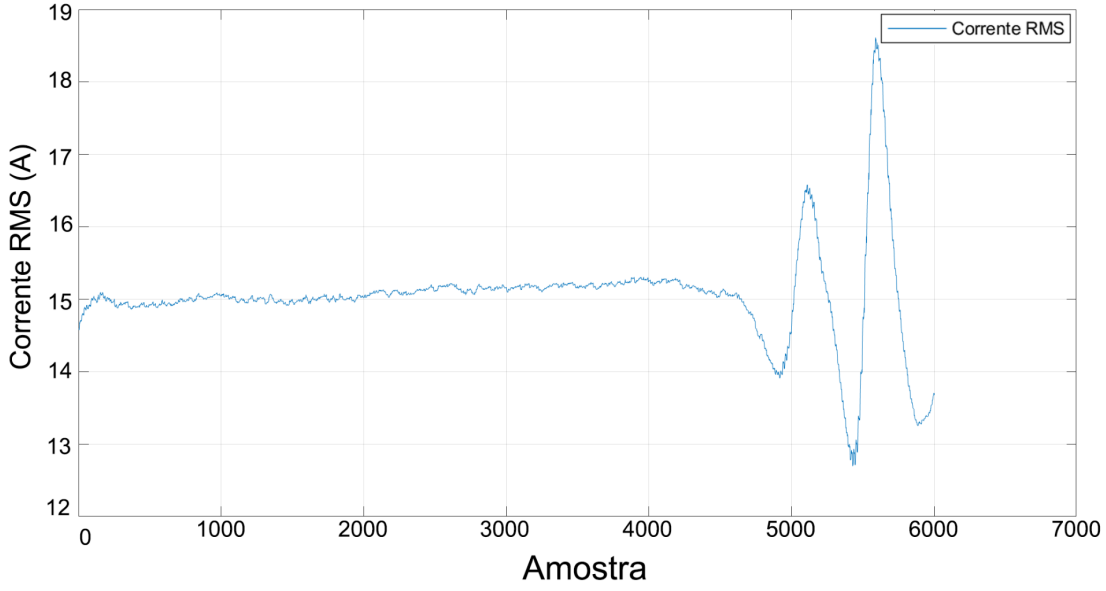


Fonte: O autor, 2023.

6.3 CONSIDERAÇÕES SOBRE OS RESULTADOS OBTIDOS

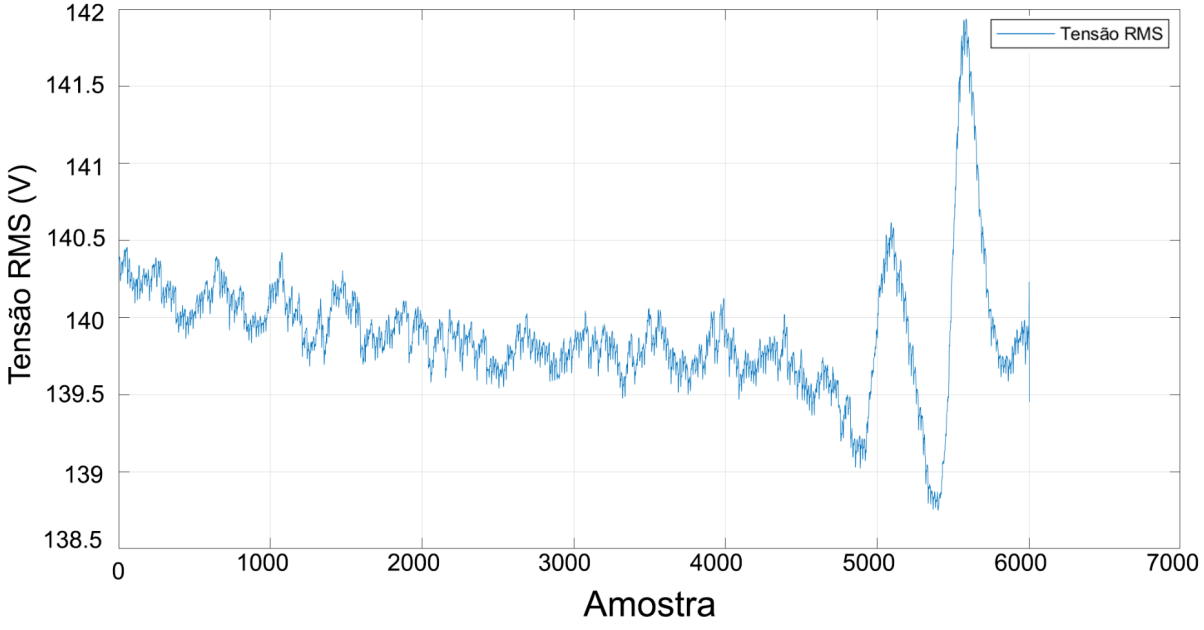
Com a substituição da biblioteca responsável pelo envio das amostras ao Firebase, obteve-se um aumento considerável no número de amostras efetivamente encaminhadas ao banco de dados. As Figuras 40 e 41 foram geradas a partir dos dados extraídos do experimento (vide Figura 32) e geradas no MATLAB, demonstrando os gráficos esperados no *site* de monitoramento.

Figura 40 - Curva de Corrente RMS geradas pelo MATLAB



Fonte: O autor, 2023.

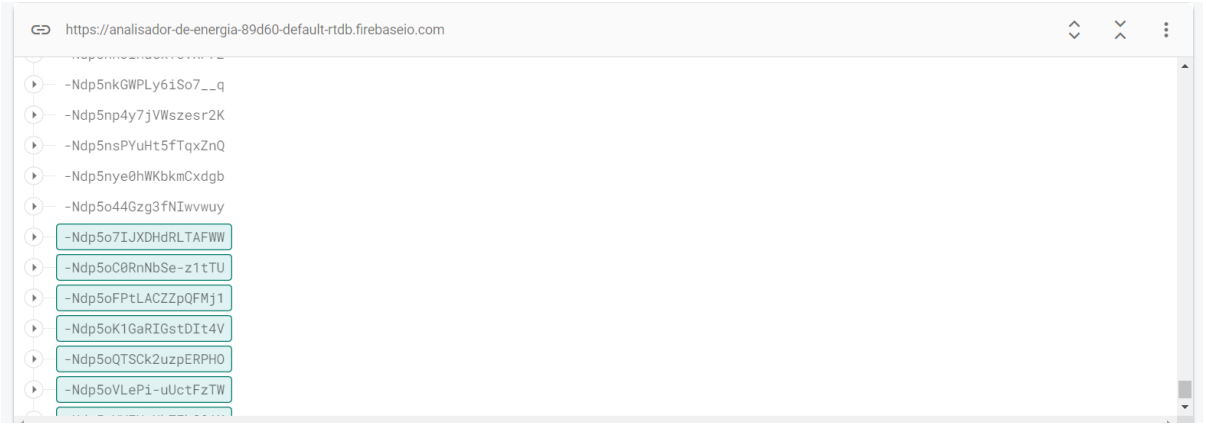
Figura 41 - Curva de Tensão RMS geradas pelo MATLAB



Fonte: O autor, 2023.

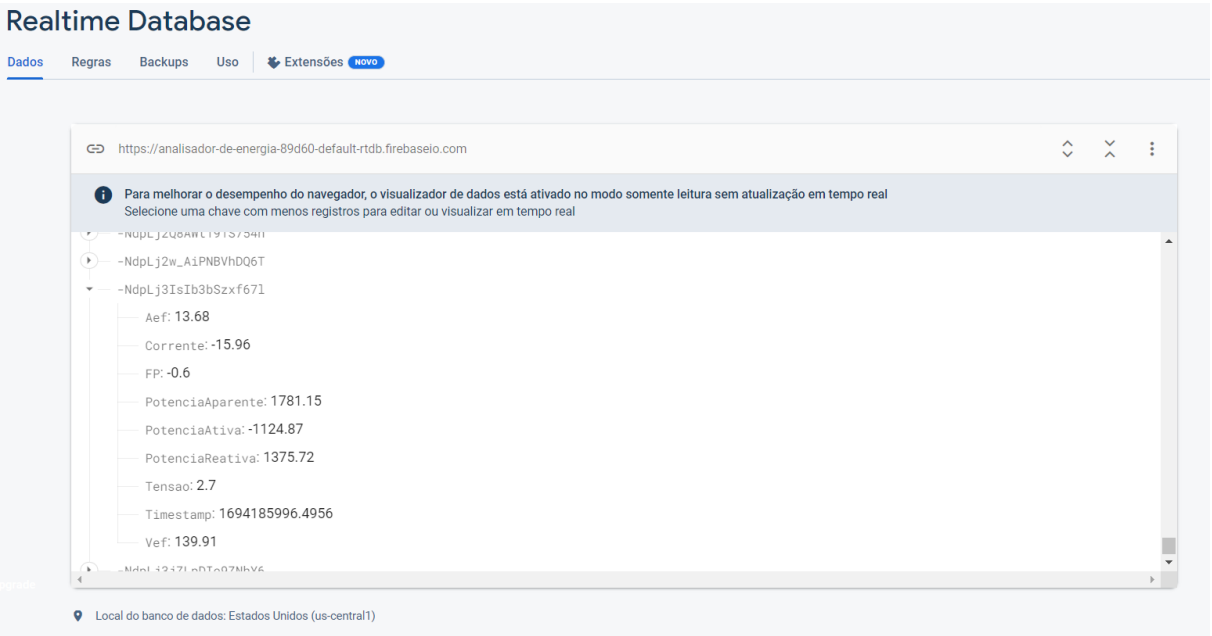
Na Figura 42 é ilustrado os dados sendo adicionados considerando uma taxa de envio de 30 amostras/seg ao armazenamento do Realtime Database após o envio pelo microcontrolador. Ressalta-se que essa taxa é suficiente e adequada para análise de grandezas em regime permanente (valores RMS de tensão e corrente, potência ativa, reativa, aparente, fator de potência, entre outras), uma vez que os dados não precisam ser enviados e processados instantaneamente. No entanto, para o caso de identificação de eventos transitórios (como curtos-circuitos e outros distúrbios) essa taxa não seria adequada. A estrutura JSON das amostras armazenadas contêm chaves descritivas para cada grandeza medida e seus respectivos valores, como observado na Figura 43.

Figura 42 - Adição de novos dados no banco de dados



Fonte: O autor, 2023.

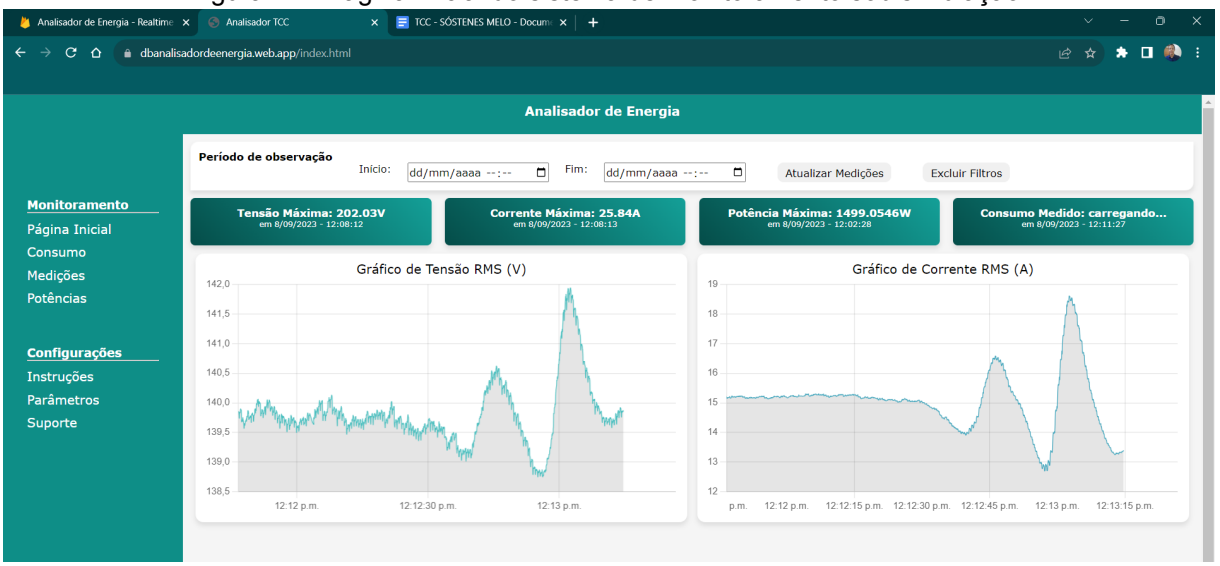
Figura 43 - Estrutura dos dados JSON no banco de dados



Fonte: O autor, 2023.

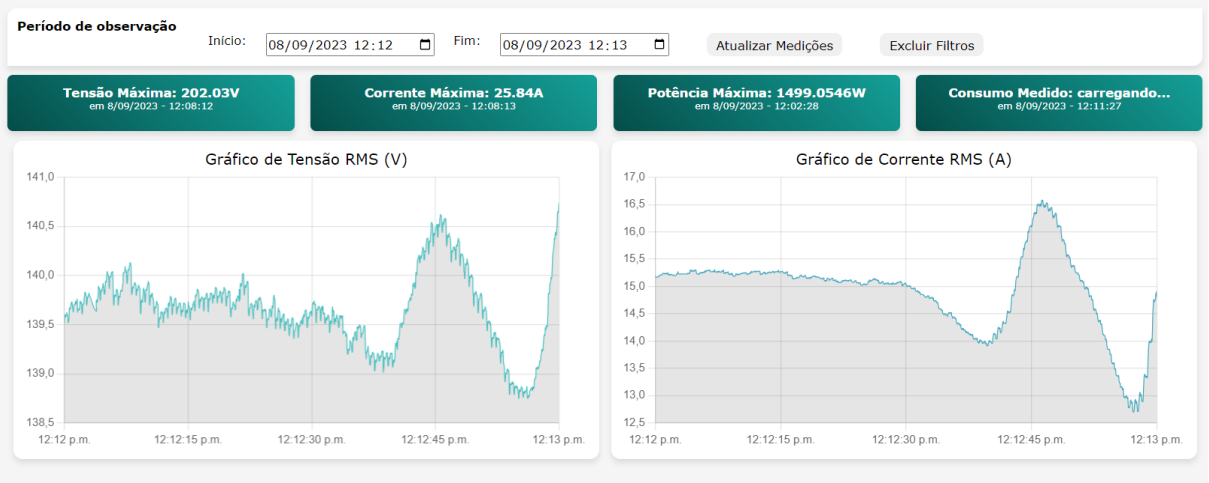
Na página inicial do *site* de monitoramento observou-se a exibição em tempo real dos valores resumidos de tensão máxima, corrente máxima e potência máxima, além das curvas de corrente e tensão RMS (do inglês, *Root Mean Square*) conforme os dados experimentais (vide Figura 44). Na Figura 45 têm-se os filtros de data inicial e final aplicados, resumindo a exibição gráfica a uma janela de observação de 1 minuto.

Figura 44 - Página Inicial do sistema de monitoramento sob simulação



Fonte: O autor, 2023.

Figura 45 - Filtro de observação aplicado à página inicial

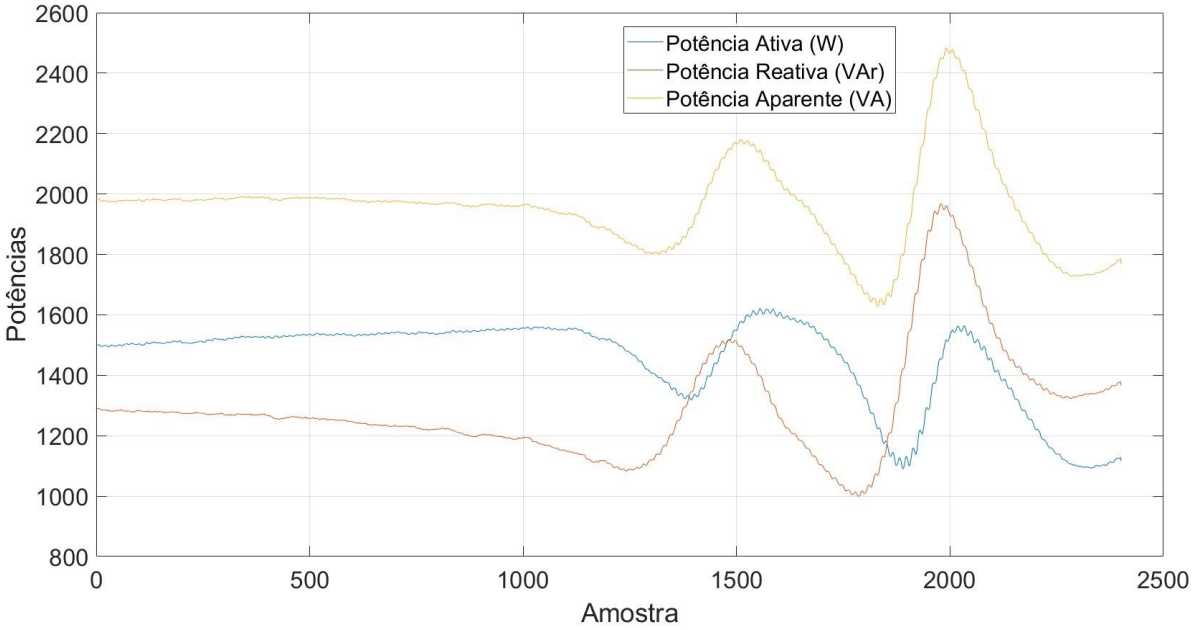


Fonte: O autor, 2023.

Por sua vez, a Figura 46 apresenta as curvas de potência ativa, reativa e aparente geradas com o MATLAB e esperadas para a página de potências do *site* de monitoramento. Enquanto que a Figura 47 mostra o resultado obtido no *site* de

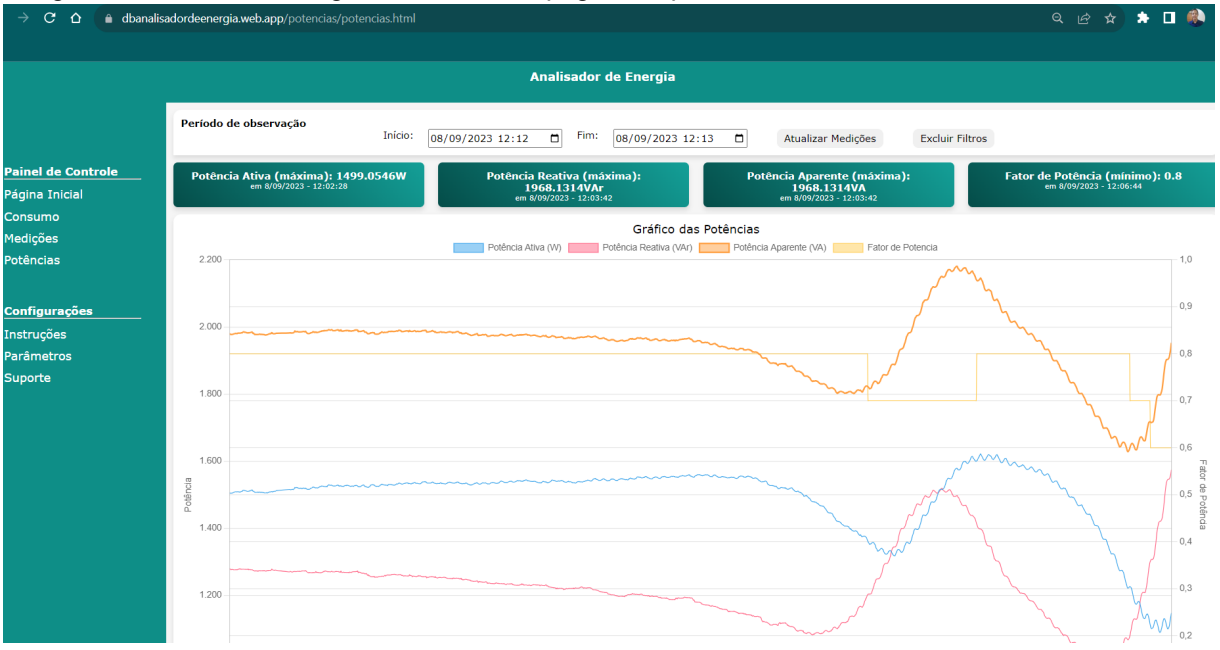
monitoramento quando se aplica o mesmo filtro de observação de 1 minuto utilizado na análise ilustrada pela Figura 45.

Figura 46 - Curva de potência aparente, ativa e reativa geradas pelo MATLAB



Fonte: O autor, 2023.

Figura 47 - Resultado do gráfico exibido na página de potências do *site* utilizando o filtro de datas

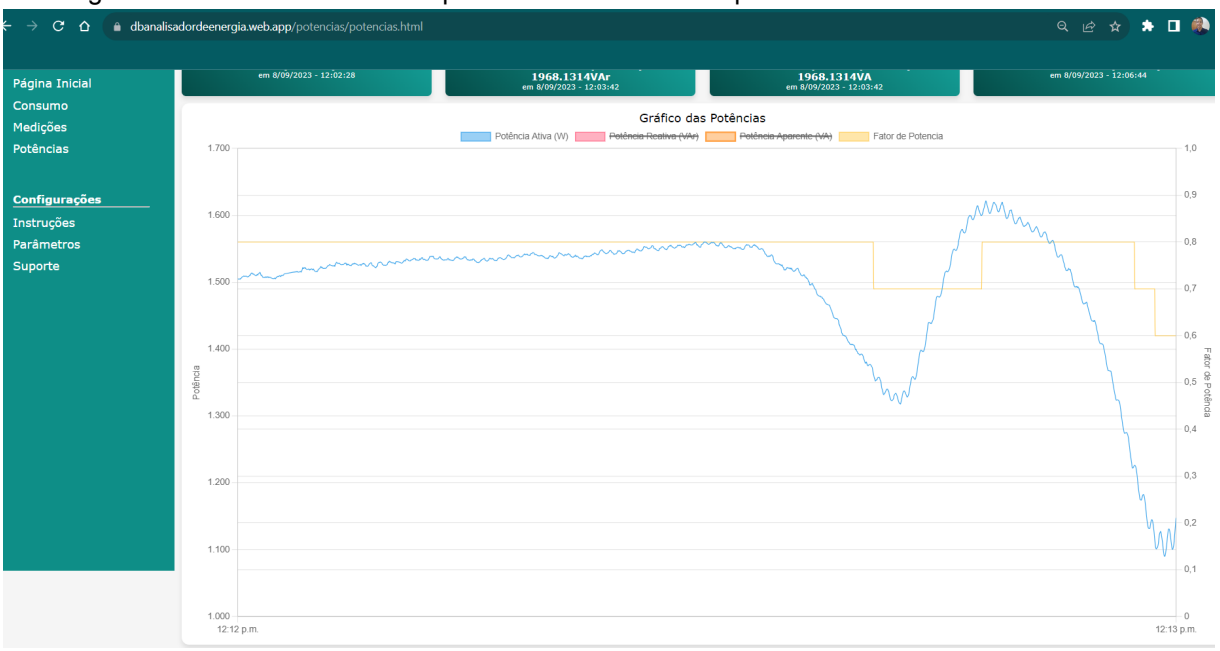


Fonte: O autor, 2023.

Utilizando a interatividade com o gráfico, foi possível detalhar as curvas de potência ativa e fator de potência ao desligar as curvas de potência aparente e

potência reativa por meio da legenda do gráfico. Tal operação pode ser observada por meio da Figura 48.

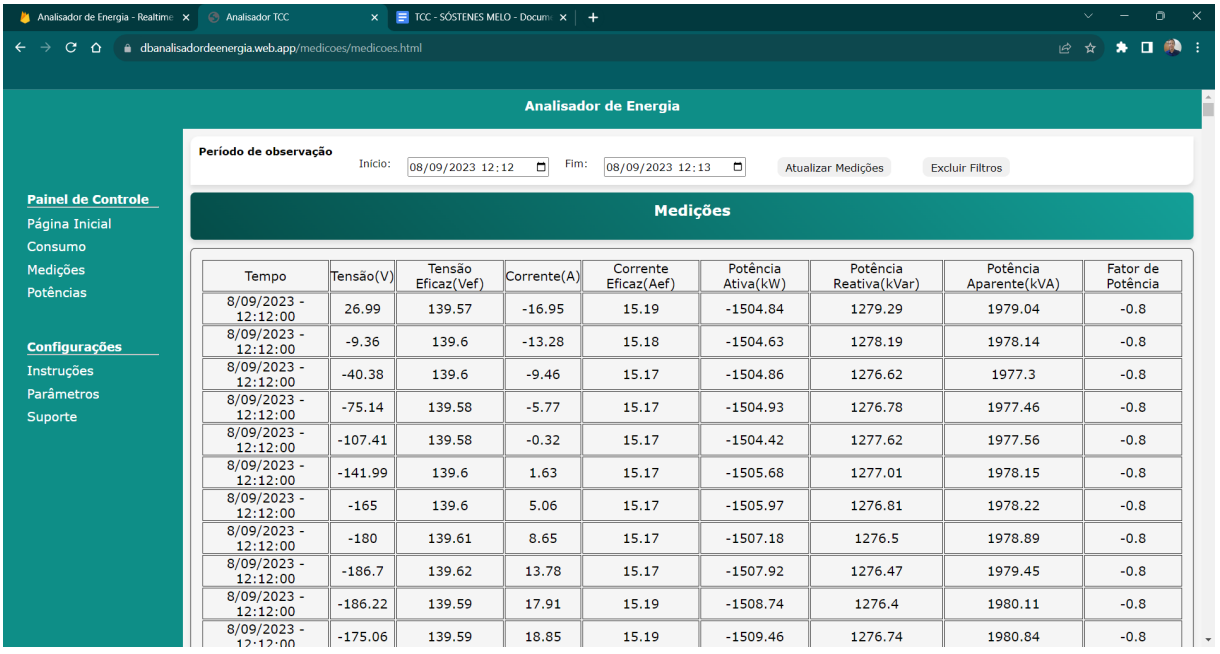
Figura 48 - Foco nas curvas de potência ativa e fator de potência do *site* de monitoramento



Fonte: O autor, 2023.

Por fim, na Figura 49 observam-se em tabela os valores de todas as amostras delimitadas dentro da janela de observação determinada pelos filtros de data, sendo as medições apresentadas em ordem cronológica.

Figura 49 - Página Inicial do sistema de monitoramento sob simulação

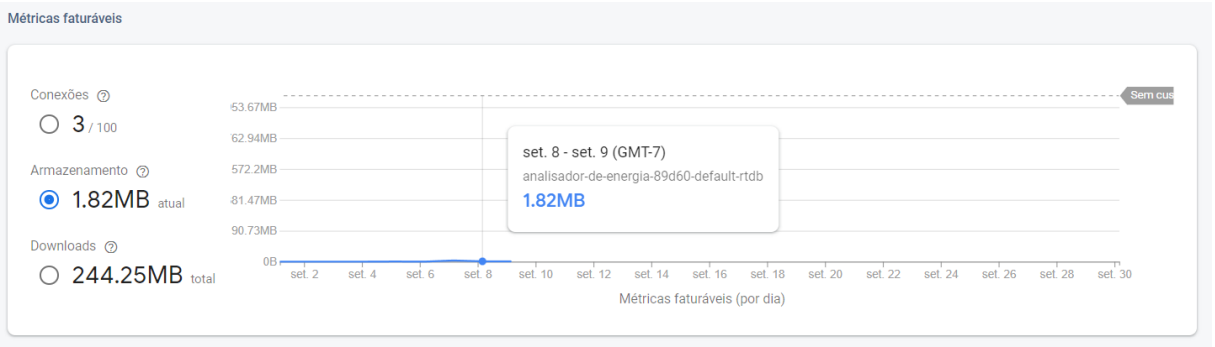


Tempo	Tensão(V)	Tensão Eficaz(Vef)	Corrente(A)	Corrente Eficaz(Aef)	Potência Ativa(kW)	Potência Reativa(kVar)	Potência Aparente(kVA)	Fator de Potência
8/09/2023 - 12:12:00	26.99	139.57	-16.95	15.19	-1504.84	1279.29	1979.04	-0.8
8/09/2023 - 12:12:00	-9.36	139.6	-13.28	15.18	-1504.63	1278.19	1978.14	-0.8
8/09/2023 - 12:12:00	-40.38	139.6	-9.46	15.17	-1504.86	1276.62	1977.3	-0.8
8/09/2023 - 12:12:00	-75.14	139.58	-5.77	15.17	-1504.93	1276.78	1977.46	-0.8
8/09/2023 - 12:12:00	-107.41	139.58	-0.32	15.17	-1504.42	1277.62	1977.56	-0.8
8/09/2023 - 12:12:00	-141.99	139.6	1.63	15.17	-1505.68	1277.01	1978.15	-0.8
8/09/2023 - 12:12:00	-165	139.6	5.06	15.17	-1505.97	1276.81	1978.22	-0.8
8/09/2023 - 12:12:00	-180	139.61	8.65	15.17	-1507.18	1276.5	1978.89	-0.8
8/09/2023 - 12:12:00	-186.7	139.62	13.78	15.17	-1507.92	1276.47	1979.45	-0.8
8/09/2023 - 12:12:00	-186.22	139.59	17.91	15.19	-1508.74	1276.4	1980.11	-0.8
8/09/2023 - 12:12:00	-175.06	139.59	18.85	15.19	-1509.46	1276.74	1980.84	-0.8

Fonte: O autor, 2023.

Quanto à capacidade de armazenamento do plano *Spark* do Firebase, observa-se pela Figura 50 que, após o armazenamento de 3 mil amostras foram ocupados 1,82 MB do banco de dados, sendo o limite sem custos financeiros do plano de 1GB, totalizando a disponibilidade para 1.648.351 amostras. A uma frequência de 30 amostras por segundo, seria possível armazenar dados referentes a aproximadamente 15 horas e 15 minutos.

Figura 50 - Armazenamento utilizado no *Realtime Database*



Fonte: Página “Métricas de Faturamento do Firebase”, 2023.

Assim, considerando os resultados obtidos a partir das simulações supracitadas, verifica-se que o sistema apresentou satisfatória confiabilidade para aplicações que requerem uma frequência de monitoramento em torno de 30 amostras por segundo, enquanto que, para altas frequências de comunicação, ocorreram perdas que podem comprometer análises assertivas.

7 CONSIDERAÇÕES FINAIS

7.1 CONCLUSÕES GERAIS

Neste trabalho foi desenvolvido um sistema de monitoramento remoto utilizando o protocolo de comunicação MQTT público da Eclipse IoT e o Firebase *Realtime Database* como banco de dados para armazenamento das medições. Para exibição dos resultados foi desenvolvido um *site* capaz de apresentar de forma gráfica e em tempo real, dados como corrente, tensão, seus respectivos valores RMS, potência ativa, reativa e aparente, e fator de potência. Os valores máximos medidos são exibidos em um quadro resumido do *site*, acompanhados de data e hora da amostra medida. O microcontrolador ESP32 foi utilizado como elo entre o *broker* e o banco de dados do Firebase, sendo responsável por processar e enviar as amostras encaminhadas pelo servidor MQTT em tempo real e de forma confiável.

Diante dos resultados apresentados, foi possível verificar que o ESP32 demonstra maior aproveitamento quando em comunicações de baixa frequência, efetivando 99% dos envios simulados ao *Realtime Database* para as frequências de 10 e 30 amostras por segundo, revelando maior eficiência do sistema quando utilizado para medições de valores eficazes com janelas mínimas de amostras a partir de 33 milissegundos. A frequência de amostragem utilizada para envio dos dados foi suficiente para acompanhamento de sinais elétricos, uma vez que dados associados a grandezas em regime permanente (valores RMS de tensão e corrente, potência ativa, reativa, aparente, fator de potência, entre outras) não precisam ser enviados e processados instantaneamente. No entanto, para envio de dados associados a fenômenos transitórios como curto-circuito e outros distúrbios, a taxa empregada neste trabalho não seria adequada.

Ademais, o sistema demonstra limitações do ponto de vista da segurança das informações, não sendo implementados no escopo deste projeto métodos de criptografia das mensagens trafegadas pela comunicação MQTT e não sendo declaradas as regras de leitura e escrita ao banco de dados.

7.2 TRABALHOS FUTUROS

Como continuidade dos estudos realizados neste trabalho, sugere-se:

- adaptar e analisar o comportamento do sistema quando dois ou mais dispositivos de medição enviam amostras simultâneas por meio da comunicação MQTT ao microcontrolador ESP32;

- desenvolver um *dashboard* interativo para o site, capaz de permitir a criação de novos tópicos para monitoramento particular dos múltiplos dispositivos de medição;
- adicionar método capaz de realizar o *backup* dos dados do *Realtime Database* e que libere o armazenamento, excluindo os dados armazenados quando atingido o limite do plano sem custos financeiros;
- implementar um sistema de alertas para notificar ao usuário sobre possíveis anormalidades detectadas pela medição;
- aprimorar a segurança do sistema contra ataques cibernéticos a fim de proteger a integridade do sistema e dos dados;
- integrar uma inteligência artificial que possa detectar padrões que representem anormalidade no ponto de observação por meio do processo de *machine-learning*.

REFERÊNCIAS

ABREU, Karen Cristina Kraemer. **História e usos da Internet.** BOCC – Biblioteca Online de Ciências da Comunicação, p. 1-9, 2009. Disponível em: https://www.bocc.ubi.pt/pag/abreu-karen-historia-e-usos-da-internet.pdf. Acesso em: 10 jun. 2023.

ALEXANDER, Charles K.; SADIKU, Matthew N. **Fundamentos de circuitos elétricos**. 5º ed. Porto Alegre: AMGH, 2013.

ALVES, Dênis Keuton. **Estimação em Tempo Real de Potência e Impedância da Rede Elétrica utilizando a Transformada Wavelet Packet Estacionária.** 2019. Tese (Doutorado em Engenharia Elétrica) – Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte, Natal, 2019. Disponível em: https://repositorio.ufrn.br/handle/123456789/20241. Acesso em 13 abr. 2023.

AL-FUQAHA, Ala *et al*. Internet of Things: a survey on enabling technologies, protocols, and applications. **IEEE Communication Surveys & Tutorials**, v. 17, n. 4, p. 2347-2376, 2015.

AMORIM JR, J. P.; RIBEIRO, C. M.; COLISTETE JR, R. Medição Remota de Parâmetros Elétricos Usando IoT Baseada no Microcontrolador ESP32. *In*: ENCONTRO CIENTÍFICO DE FÍSICA APLICADA, 10., 2019, São Paulo. **Anais** […]. São Paulo: Blucher, 2019. p. 109-116. Disponível em: https://pdfs.semanticscholar.org/76d3/0ae96091b4e4ce6d81cece93a944482b7f57.pdf. Acesso em: 03 jun. 2023.

CHEN, Min; MAO, Shiwen; LIU, Yunhao. Big data: A survey. **Mobile Networks and Applications**, v. 19, p. 171-209, 2014.

CIRILO, Carlos Eduardo. **Computação Ubíqua:** definição, princípios e tecnologias. São Carlos: UFSCar, 2008. Disponível em: https://www.academia.edu/1733697/Computa%C3%A7%C3%A3o_Ub%C3%ADqua_defini%C3%A7%C3%A3o_princ%C3%ADpios_e_tecnologias. Acesso em: 11 jun. 2023.

CONCEIÇÃO, Wellington; COSTA, Romualdo. Análise do Protocolo MQTT para Comunicação IoT através de um Cenário de Comunicação. **Caderno de Estudos em Sistemas de Informação**, v. 5, n. 2, 2019.

GOOGLE. **Deixe seu app o melhor possível**, [2023]. Página inicial. Disponível em: https://firebase.google.com/?hl=pt-br. Acesso em: 08 jul. 2023.

HANSMANN, Uwe *et al.* **Pervasive computing handbook.** Springer Science & Business Media, 2013. Disponível em: https://www.biblio.com/book/pervasive-computing-handbook-hansmann-uwe-etal/d/1471157294. Acesso em: 24 jun. 2023.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Pesquisa Nacional por Amostra de Domicílios Contínua:** Acesso à Internet e à televisão e posse de telefone móvel celular para uso pessoal 2021**.** Rio de Janeiro: IBGE, 2022. 12 p. Disponível em: https://biblioteca.ibge.gov.br/index.php/biblioteca-catalogo?view=detalhes&id=2101963. Acesso em: 11 jun. 2023.

KONAYAGE, Fernando. Introdução ao ESP32. **Fernando K Tecnologia,** 2017. Disponível em: https://www.fernandok.com/2017/11/introducao-ao-esp32.html. Acesso em: 10 ago. 2023.

LEITE, J.R. Emiliano; MARTINS, Paulo; URSINI, Edson. A Internet das Coisas (IoT): Tecnologias e Aplicações. In: **Brazilian Technology Symposium**. 2017. Disponível em: https://lcv.fee.unicamp.br/images/BTSym-17/Papers/76926.pdf. Acesso em: 17 abr. 2023.

MATTERN, Friedemann; FLOERKEMEIER, Christian. From the Internet of Computers to the Internet of Things. *In*: SACHS, Kai; PETROV, Ilia; GUERRERO, Pablo (Ed.). **From active data management to event-based systems and more: papers in honor of Alejandro Buchmann on the occasion of his 60th birthday**. Springer, 2010. p. 242-259.

MCAFEE, Andrew; BRYNJOLFSSON, Erik. Big data: the management revolution. **Harvard Business Review**, v. 90, n. 10, p. 60-68, 2012.

MDN WEB DOCS. **Arpanet.** [S. l.], 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary/Arpanet. Acesso em: 3 jul. 2023.

MDN WEB DOCS. **O Que é Javascript?**. [S. l.], 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Acesso em: 17 ago. 2023.

MOON, Bernard. Internet of Things & Hardware Industry Overview 2016. **SparkLabs Global Ventures**, 2016. Disponível em: https://www.slideshare.net/bernardmoon/internet-of-things-hardware-industry-report-2016. Acesso em: 03 jun. 2023.

FERREIRA NETO, Amir; CORRÊA, Wilson; PEROBELLI, Fernando. Consumo de Energia e Crescimento Econômico: uma Análise do Brasil no período 1970-2009. **Análise Econômica**, v. 34, n. 65, 2016.

OASIS OPEN. MQTT Version 3.1.1. **OASIS Standard**, 2014. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. Acesso em: 03 jun. 2023.

PARLIAMENTARY OFFICE OF SCIENCE AND TECHNOLOGY. Pervasive Computing. **POSTnote**, Londres, n. 263, p. 1-4, maio 2006. Disponível em: https://www.parliament.uk/globalassets/documents/post/postpn263.pdf. Acesso em: 08 jun. 2023.

QUINCOZES, Silvio; TUBINO, Emilio; KAZIENKO, Juliano. MQTT protocol: fundamentals, tools and future directions. **IEEE Latin America Transactions**, v. 17, n. 09, p. 1439-1448, 2019.

RIOS, Artur de Almeida. **Aplicação de Internet das Coisas no Monitoramento de Corrente, Tensão e Temperatura em Motor de Indução Trifásico.** 2019. 112 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, Uberlândia, 2019. Disponível em: https://repositorio.ufu.br/handle/123456789/26442. Acesso em: 03 jun. 2023.

SANTOS, Bruno *et al*. Internet das coisas: da teoria à prática. *In:* SIQUEIRA, Frank Augusto; LUNG, Lau Cheuk; GREVE, Fabíola G. P., FREITAS, Allan E. S. (Org.). **Livro Texto Minicursos:** XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. vol. 1. Porto Alegre: SBC, 2016. p. 15–52. Disponível em: https://sbrc2016.ufba.br/downloads/anais/MinicursosSBRC2016.pdf. Acesso em: 08 jun. 2023.

SANTOS, Erik *et al*. Protótipo Trifásico para Monitoramento de Consumo de Energia Elétrica Utilizando A Plataforma Arduino. **Brazilian Applied Science Review**, v. 4, n. 5, p. 2939-2953, 2020.

WEGNER, Philipp. Global IoT market size to grow 19% in 2023 — IoT shows resilience despite economic downturn. **IoT Analytics**, 2023. Disponível em: https://iot-analytics.com/iot-market-size/. Acesso em: 20 ago. 2023.

WEISER, Mark. The Computer for the 21st Century. **Mobile Computing and Communications Review**, v. 3, n. 3, p. 3-11, 1999.

APÊNDICE A: ROTINA PARA SIMULAÇÃO DE ENVIO MQTT

Neste apêndice, apresenta-se a rotina de simulação do envio das medições MQTT ao *broker*, realizado por meio da linguagem de programação Python, como mencionado no trabalho.

```
from paho.mqtt import client as mqtt
import time
clientId = "Simulação"
port = 1883
broker = "mqtt.eclipseprojects.io"

client = mqtt.Client(clientId)
client.connect(broker,port)

data = open(r"G:\Documents\UACSA\TCC\SIMULAÇÃO\dadosMedicao.csv")
a = 0
for valor in data:
        if a != 0:
            dados = valor.replace("\n","").split(",")
            client.publish('medicaodados','''{"Timestamp":'''+'''{:.4f},
"Tensao":{:.2f}, "Corrente":{:.2f}, "Aef":{:.2f}, "FP":{:.1f},
"PotenciaAparente":{:.2f}, "PotenciaAtiva":{:.2f}, "PotenciaReativa":{:.2f},
"Vef":{:.2f}'''.format(time.time(),float(dados[1]),float(dados[2]),float(dado
s[3]),float(dados[4]),float(dados[5]),float(dados[6]),float(dados[7]),float(d
ados[8]))+"}")
            print(a)
        a+=1
        time.sleep(0.0333)
```

APÊNDICE B: ROTINA APRIMORADA DO ESP32

Neste apêndice, apresenta-se a rotina implementada no microcontrolador ESP32 para processamento e envio das medições recebidas por protocolo MQTT ao Firebase *Realtime Database*.

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <PubSubClient.h>

// Provide the token generation process info.
#include <addons/TokenHelper.h>

// Provide the RTDB payload printing info and other helper functions.
#include <addons/RTDBHelper.h>

/* 1. Define the WiFi credentials */
#define WIFI_SSID "Edvanio"
#define WIFI_PASSWORD "cleide1607"
WiFiClient wifiClient;

// For the following credentials, see
examples/Authentications/SignInAsUser/EmailPassword/EmailPassword.ino

/* 2. Define the API Key */
#define API_KEY "AIzaSyCSeWymnTZf8GndGlxZM8brzO2lDewAXoI"

/* 3. Define the RTDB URL */
#define DATABASE_URL
"https://dbanalisadordeenergia-default-rtdb.firebaseio.com/"
//<databaseName>.firebaseio.com or \
                           //<databaseName>.<region>.firebasedatabase.app

int mqttcaiu = -1;
float tempoMQTT = 0;
int enviados = 0;
float tempoMED = 0;
int tempo = 0;
int tempo2 = 0;


/* 4. Define the user Email and password that alreadey registerd or added in
 * your project */
#define USER_EMAIL "sostenesilvaa@gmail.com"
#define USER_PASSWORD "123456"

// Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;


//MQTT Server
const char* BROKER = "mqtt.eclipseprojects.io"; //URL do broker MQTT que se
deseja utilizar
int PORT = 1883;                      // Porta do Broker MQTT
```

```
#define ID_MQTT  "MONITORAMENTO"             //Informe um ID único e seu.
Caso sejam usados IDs repetidos a última conexão irá sobrepor a anterior.
#define TOPIC_SUBSCRIBE "medicaodados"   //Informe um Tópico único. Caso
sejam usados tópicos em duplicidade, o último irá eliminar o anterior.
PubSubClient MQTT(wifiClient);        // Instancia o Cliente MQTT passando o
objeto espClient

//Declaração das Funções
void mantemConexoes();  //Garante que as conexões com WiFi e MQTT Broker se
mantenham ativas
void conectaWiFi();     //Faz conexão com WiFi
void conectaMQTT();     //Faz conexão com Broker MQTT
void recebePacote(char* topic, byte *payload, unsigned int length);

int cont = 0;
void recebePacote(char* topic, byte *payload, unsigned int length)
{
  enviados++;
  Serial.print(millis());
  Serial.print(",");
  Serial.print(enviados);
  Serial.print(",");

  tempo = millis();

  String dadosMQTT;
  for (int i=0;i<length;i++){
    dadosMQTT += ((char)payload[i]);
  }
  Serial.println(dadosMQTT);
  FirebaseJson json(dadosMQTT);
  Firebase.pushJSONAsync(fbdo,"/",json);
  cont++;

  tempo2 = millis()-tempo;

  tempoMED += tempo2;
  //Serial.print("Tempo médio de envio: ");
  Serial.print(tempoMED/enviados);
  Serial.print(",");
  Serial.print(mqttcaiu);
  Serial.print(",");
  Serial.print(tempoMQTT/(mqttcaiu+1));
  Serial.println();
}

void setup() {
  Serial.begin(115200);

  conectaWiFi();

  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the user sign in credentials */
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;
```

```
  /* Assign the callback function for the long running token generation task
*/
  config.token_status_callback = tokenStatusCallback; // see
addons/TokenHelper.h

  Firebase.begin(&config, &auth);

  Firebase.reconnectWiFi(true);

  MQTT.setServer(BROKER, PORT);
  MQTT.setCallback(recebePacote);
}

void mantemConexoes() {
    if (!MQTT.connected()) {
      conectaMQTT();
    }
    conectaWiFi();
}

void conectaWiFi() {

  if (WiFi.status() == WL_CONNECTED) {
     return;
  }

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); // Conecta na rede WI-FI
  while (WiFi.status() != WL_CONNECTED) {
      //Serial.print(".");
      delay(300);
  }

  Serial.print("Conectado com sucesso, na rede: ");
  Serial.print(WIFI_SSID);
  Serial.print("  IP obtido: ");
  Serial.println(WiFi.localIP());*/
}

void loop() {
  mantemConexoes();
  MQTT.loop();
  if (Firebase.ready()){

  }
}

void conectaMQTT() {
  mqttcaiu ++;

  tempo = millis();

  while (!MQTT.connected()) {
      Serial.print("Conectando ao Broker MQTT: ");
      Serial.println(BROKER);
      if (MQTT.connect(ID_MQTT)) {
          Serial.println("Conectado ao Broker com sucesso!");
          MQTT.subscribe(TOPIC_SUBSCRIBE);
      }
      else {
          Serial.println("Não foi possível se conectar ao broker.");
          Serial.println("Nova tentativa de conexão em 10s");
```

```
            delay(3000);
        }
  Serial.print(millis());
  Serial.println("ms");
  Serial.print("A reconexão durou: ");*/
  tempo2 = millis()-tempo;
  Serial.println("ms");
  Serial.print("O tempo médio de reconexão foi de: ");*/
  tempoMQTT += tempo2;
  }
}
```

APÊNDICE C: CÓDIGOS HTML, JAVASCRIPT E CSS DO *SITE* DE MONITORAMENTO

Neste apêndice são apresentados os códigos de estruturação, estilização e interatividade implementados para desenvolvimento da página *web* de monitoramento remoto.

A. Página inicial
   a. Código HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analisador TCC</title>
    <link rel="stylesheet" href="index.css"/>

</head>

<body id="body">

    <!-- HEADER -->
    <header class="topo">
    <!-- LOGOMARCA -->
        <h4>Analisador de Energia</h4>
    </header>
    <!-- FIM DO HEADER -->
    <section class="main">
        <div class="sidebar">
            <div class="menu">
                <h3>Monitoramento</h3>
                <hr>
                <a href="/index.html">Página Inicial</a>
                <a href="/consumo/consumo.html">Consumo</a>
                <a href="/medicoes/medicoes.html">Medições</a>
                <a href="/potencias/potencias.html">Potências</a>
            </div>
            <div class="menu">
                <h3>Configurações</h3>
                <hr>
                <a href="/instrucoes/instrucoes.html">Instruções</a>
                <a href="/parametros/parametros.html">Parâmetros</a>
                <a href="/suporte/suporte.html">Suporte</a>
            </div>
        </div>
        <!-- INICIO DO CONTENT (CONTEÚDO DO DASHBOARD) -->
        <div class="content">

            <!-- INICIO DO FILTRO CALENDARIO -->
            <div class="filtros">
                <h5>Período de observação</h5>
                <div class="calendario">
                    <p>Início:</p><input type="datetime-local"
class="filtrodata" id="dataInicialFiltro"></input>
                    <p>Fim:</p><input type="datetime-local"
class="filtrodata" id="dataFinalFiltro"></input>
```

```
                    <button class="filtrodata btnfiltro"
id="aplicarFiltro">Atualizar Medições</button>
                    <button class="filtrodata btnfiltro"
id="excluirFiltros">Excluir Filtros</button>
                </div>
            </div>

            <!-- INICIO DO PAINEL MEDIÇÕES MÁXIMAS -->
            <div class="resumomedicoes">
                <div class="boxmedmax" id="maxTensao"></div>
                <div class="boxmedmax" id="maxCorrente"></div>
                <div class="boxmedmax" id="maxPotencia"></div>
                <div class="boxmedmax" id="consumoMedido"></div>
            </div>
            <!-- INCIO DOS GRÁFICOS -->
            <div class="graficos">

                <!-- INCIO DO DIV DO GRÁFICO TENSÃO -->
                <div class="painelgrafico">
                    <p>Gráfico de Tensão RMS (V)</p>
                    <canvas class="tensao" id="graphTensao"></canvas>
                   <!-- <canvas class="scrollTensao" ></canvas> -->
                </div>

                <!-- INCIO DO DIV DO GRÁFICO CORRENTE -->
                <div class="painelgrafico">
                    <p>Gráfico de Corrente RMS (A)</p>
                    <canvas class="corrente"></canvas>
                </div>

            </div>
        </div>
    </section>

    <!-- IMPORTAÇÕES DE JS -->
    <script src="https://kit.fontawesome.com/9ade8f3144.js"
crossorigin="anonymous"></script> <!-- PACK DE ICONS -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <!--
BIBLIOTECA CHART JS (GRÁFICOS) -->

    <!-- ADAPTADOR DE DATAS PRO CHART JS -->

    <script src="https://cdn.jsdelivr.net/npm/luxon"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-plugin-streaming"></script>


    <!-- MANIPULAÇÃO DE DATA E HORA -->
    <script src="https://momentjs.com/downloads/moment.min.js"></script>
    <script
src="https://momentjs.com/downloads/moment-timezone-with-data.min.js"></scrip
t>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns/dist/chartjs-adapt
er-date-fns.bundle.min.js"></script>

    <!-- ARQUIVO JAVASCRIPT CONTENDO O FIREBASE E CONFIGURAÇÕES -->
    <script type="module" src="index.js"></script>
```

```
</body>
</html>
```

b. Código CSS:

```
*{
    margin:0;
    padding: 0;
    box-sizing: border-box;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    width: 100%;
}
html,body{
    height: 100%;
    background: #F5F5F5;
}
.topo{
    display: flex;
    flex-wrap: nowrap;
    width: 100%;
    height: 7%;
    background: #0e8e87;
    padding: 10px;
    text-align: center;
    color: white;
}
.main{
    display: flex;
    flex-wrap: wrap;
    width: 100%;
    height: 100%;
}
.sidebar{
    background: #0e8e87;
    color: white;
    width: 15%;
    height: 100%;
    padding: 2%;
}
.menu{
    margin-top: 50px;
}
.menu h3{
    font-size: 15px;
}
.menu a{
    display: block;
    text-decoration: none;
    color: white;
    margin-top: 10px;
}
.content{
    display: flex;
    flex-wrap: wrap;
    flex-direction:column;
    width: 85%;
    padding: 10px;

}
.filtros{
```

```
    background: #ffffff;
    padding: 10px;
    border: 3px;
    border-radius: 10px 0px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.calendario{
    display: flex;
    width: 100%;
    height: fit-content;
    justify-content: center;

}
.calendario p{
    width: fit-content;
    height: fit-content;
    font-size: small;
}
.filtrodata{
    width: fit-content;
    margin-bottom: 0;
    margin-inline: 20px;

}
.resumomedicoes{
    display: flex;
    justify-content: space-between;
    width: 100%;
    height: 60px;
    margin-top: 10px;

}
.boxmedmax{
    background-color: bisque;
    width: 24%;
    height: 100%;
    padding: 10px;
    border-radius: 6px;
    color: white;
    background: linear-gradient(45deg,#054d49,#14A098);
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.boxmedmax p{
    font-size: 10px;
}
.graficos{
    display: flex;
    justify-content: space-around;
    text-align: center;
    margin-top: 10px;
}
.painelgrafico{
    width: 49%;
    background: #ffffff;
    border-radius: 10px;
    padding: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.btnfiltro{
    padding-inline: 10px;
```

```
    height: 25px;
    border-radius: 8px;
    border: 0px;
}
.btnfiltro:hover{
    background: #054d49;
    color: white;
}
```

c. Código JS:

```
// --------------- CONFIGURAÇÃO DO FIREBASE --------------- //
// Import the functions you need from the SDKs you need
import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-app.js";
import { getAnalytics } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-analytics.js";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyAWPDemS-kFBcBJb3e02EypnKD3z9vVn48",
  authDomain: "analisador-de-energia-89d60.firebaseapp.com",
  projectId: "analisador-de-energia-89d60",
  storageBucket: "analisador-de-energia-89d60.appspot.com",
  messagingSenderId: "806459782406",
  appId: "1:806459782406:web:04f0af6e6d3bef909a5a87",
  measurementId: "G-0BXMQXPDQF"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

// --------------- FIM DA CONFIGURAÇÃO DO FIREBASE --------------- //

// --------------- INICIO DA ITERAÇÃO COM O BD --------------- //

import {getDatabase, ref, onChildAdded} from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-database.js";
const db = getDatabase();

const banco = ref(db);

var graphTensão = document.getElementById('graphTensao');
//var scrollTensão = document.getElementsByClassName('scrollTensao');
var graphCorrente = document.getElementsByClassName('corrente');
const dataInicialFiltro = document.getElementById('dataInicialFiltro');
const dataFinalFiltro = document.getElementById('dataFinalFiltro');
const excluirFiltros = document.getElementById('excluirFiltros');
const aplicarFiltro = document.getElementById('aplicarFiltro');

const valorMaxTensao = document.getElementById('maxTensao');
const valorMaxCorrente = document.getElementById('maxCorrente');
const valorMaxPotencia = document.getElementById('maxPotencia');
const valorConsumo = document.getElementById('consumoMedido');
```

```
var maxMedidas =
{TimestampTensao:0,Tensao:0,TimestampCorrente:0,Corrente:0,TimestampPotenciaA
tiva:0,PotenciaAtiva:0,consumo:0};
valorMaxTensao.innerHTML = "<h5>Tensão Máxima: carregando...</h5><p>em
"+moment(new Date()).format("D/MM/YYYY - HH:mm:ss")+"<p>";
valorMaxCorrente.innerHTML = "<h5>Corrente Máxima: carregando...</h5><p>em
"+moment(new Date()).format("D/MM/YYYY - HH:mm:ss")+"<p>";
valorMaxPotencia.innerHTML = "<h5>Potência Máxima: carregando...</h5><p>em
"+moment(new Date()).format("D/MM/YYYY - HH:mm:ss")+"<p>";
valorConsumo.innerHTML = "<h5>Consumo Medido: carregando...</h5><p>em
"+moment(new Date()).format("D/MM/YYYY - HH:mm:ss")+"<p>";



function gerarParOrdenado(date,valor,listaSaida){
    let dataehora = date * 1000;
    listaSaida.push({x: dataehora, y: valor});
}

var config = {
    animation: false,
    plugins:{
        legend: false,
        streaming:{
            duration: 100000,
            ttl: 9999999999999999,
            frameRate: 60,
        },
    },
    scales: {
        x:{
            type:'realtime',
            time: {
            },
            ticks: {
                maxTicksLimit: 5,
            }

        },
        y: {
            beginAtZero: false,
        },
    }
}
const gTensao = new Chart(graphTensão, {
    type: 'line',
    data: {
        datasets: [{
            label: 'Tensão',
            data: [],
            borderColor: 'rgb(75, 192, 192)',
            borderWidth: 1,
            pointRadius: 0,
            tension: 0.7,
            fill: true,
        }]
    },
    options: config,
});
const gCorrente = new Chart(graphCorrente, {
    type: 'line',
```

```
    data: {
        datasets: [{
        label: 'Corrente',
        data: [],
        borderColor: 'rgb(75, 170, 192)',
        borderWidth: 1,
        pointRadius: 0,
        tension: 0.7,
        fill: true,
        }]
    },
    options: config,
});

onChildAdded(banco, (snapshot)=>{
    const data = snapshot.val();

gerarParOrdenado(data.Timestamp,data.Aef,gCorrente.data.datasets[0].data);
    gerarParOrdenado(data.Timestamp,data.Vef,gTensao.data.datasets[0].data);

    // ---------- Filtros ---------- //
    if (data.Corrente > maxMedidas.Corrente){
        maxMedidas.TimestampCorrente = data.Timestamp;
        maxMedidas.Corrente = data.Corrente;
        valorMaxCorrente.innerHTML = "<h5>Corrente Máxima:
"+maxMedidas.Corrente+"A</h5><p>em "+moment(new
Date(maxMedidas.TimestampCorrente*1000)).format("D/MM/YYYY -
HH:mm:ss")+"<p>";
    }
    if (data.Tensao > maxMedidas.Tensao){
        maxMedidas.TimestampTensao = data.Timestamp;
        maxMedidas.Tensao = data.Tensao;
        valorMaxTensao.innerHTML = "<h5>Tensão Máxima:
"+maxMedidas.Tensao+"V</h5><p>em "+moment(new
Date(maxMedidas.TimestampTensao*1000)).format("D/MM/YYYY - HH:mm:ss")+"<p>";
    }
    if (data.PotenciaAtiva > maxMedidas.PotenciaAtiva){
        maxMedidas.TimestampPotenciaAtiva = data.Timestamp;
        maxMedidas.PotenciaAtiva = data.PotenciaAtiva;
        valorMaxPotencia.innerHTML = "<h5>Potência Máxima:
"+maxMedidas.PotenciaAtiva+"W</h5><p>em "+moment(new
Date(maxMedidas.TimestampPotenciaAtiva*1000)).format("D/MM/YYYY -
HH:mm:ss")+"<p>";
    }
});

window.setInterval(()=>{
    gCorrente.update();
    gTensao.update();
},1000);

function filtroChartX(chart,padrao,input){
    if (padrao === "min"){
        config.scales.x.min = input;
    }else{
        config.scales.x.max = input;
    }
    config.scales.x.type = "time";
    chart.update();
}
```

```
aplicarFiltro.addEventListener("click",(event)=>{
    if (dataInicialFiltro.value != ""){
        let dataInicial = new Date(dataInicialFiltro.value).getTime();
        filtroChartX(gTensao,"min",dataInicial);
        filtroChartX(gCorrente,"min",dataInicial);
    }
    if (dataFinalFiltro.value !=""){
        let dataFinal = new Date(dataFinalFiltro.value).getTime()
        filtroChartX(gTensao,"max",dataFinal);
        filtroChartX(gCorrente,"max",dataFinal);
    }
    config.plugins.streaming.duration = 10000000;
    gTensao.update();
    gCorrente.update();
});

excluirFiltros.addEventListener("click",(event)=>{
    dataInicialFiltro.value ="";
    dataFinalFiltro.value = "";
    config.scales.x.type = "realtime";
    gTensao.update();
    gCorrente.update();
    config.plugins.streaming.duration = 100000;
});
```

B. Página de Potências

a. Código HTML

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analisador TCC</title>
    <link rel="stylesheet" href="potencias.css"/>

</head>

<body>

    <!-- HEADER -->
    <header class="topo">
    <!-- LOGOMARCA -->
        <h4>Analisador de Energia</h4>
    </header>
    <!-- FIM DO HEADER -->
    <section class="main">
        <div class="sidebar">
            <div class="menu">
                <h3>Painel de Controle</h3>
                <hr>
                <a href="/index.html">Página Inicial</a>
                <a href="/consumo/consumo.html">Consumo</a>
                <a href="/medicoes/medicoes.html">Medições</a>
                <a href="/potencias/potencias.html">Potências</a>
            </div>
            <div class="menu">
                <h3>Configurações</h3>
                <hr>
                <a href="/instrucoes/instrucoes.html">Instruções</a>
                <a href="/parametros/parametros.html">Parâmetros</a>
```

```
                <a href="/suporte/suporte.html">Suporte</a>

            </div>
        </div>
        <!-- INICIO DO CONTENT (CONTEÚDO DO DASHBOARD) -->
        <div class="content">

            <!-- INICIO DO FILTRO CALENDARIO -->
            <div class="filtros">
                <h5>Período de observação</h5>
                <div class="calendario">
                    <p>Início:</p><input type="datetime-local"
class="filtrodata" id="dataInicialFiltro"></input>
                    <p>Fim:</p><input type="datetime-local"
class="filtrodata" id="dataFinalFiltro"></input>
                    <button class="filtrodata btnfiltro"
id="aplicarFiltro">Atualizar Medições</button>
                    <button class="filtrodata btnfiltro"
id="excluirFiltros">Excluir Filtros</button>
                </div>
            </div>

            <!-- INICIO DO PAINEL MEDIÇÕES MÁXIMAS -->
            <div class="resumomedicoes">
                <div class="boxmedmax" id="maxPotenciaAtiva"></div>
                <div class="boxmedmax" id="maxPotenciaReativa"></div>
                <div class="boxmedmax" id="maxPotenciaAparente"></div>
                <div class="boxmedmax" id="minFP"></div>
            </div>
            <!-- INCIO DOS GRÁFICOS -->
            <div class="graficos">

                <!-- INCIO DO DIV DO GRÁFICO TENSÃO -->
                <div class="painelgrafico">
                    <p>Gráfico das Potências</p>
                    <canvas id="graphPotencias"></canvas>
                </div>
            </div>
        </div>
    </section>

    <!-- IMPORTAÇÕES DE JS -->
    <script src="https://kit.fontawesome.com/9ade8f3144.js"
crossorigin="anonymous"></script> <!-- PACK DE ICONS -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <!--
BIBLIOTECA CHART JS (GRÁFICOS) -->

    <!-- ADAPTADOR DE DATAS PRO CHART JS -->

    <script src="https://cdn.jsdelivr.net/npm/luxon"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-plugin-streaming"></script>


    <!-- MANIPULAÇÃO DE DATA E HORA -->
    <script src="https://momentjs.com/downloads/moment.min.js"></script>
```

```
    <script
src="https://momentjs.com/downloads/moment-timezone-with-data.min.js"></scrip
t>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns/dist/chartjs-adapt
er-date-fns.bundle.min.js"></script>

    <!-- ARQUIVO JAVASCRIPT CONTENDO O FIREBASE E CONFIGURAÇÕES -->
    <script type="module" src="potencias.js"></script>
</body>
</html>
```

b. Código CSS

```
*{
    margin:0;
    padding: 0;
    box-sizing: border-box;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    width: 100%;
}
html,body{
    height: 100%;
    background: #F5F5F5;
}
.topo{
    display: flex;
    flex-wrap: nowrap;
    width: 100%;
    height: 7%;
    background: #0e8e87;
    padding: 10px;
    text-align: center;
    color: white;
}

.main{
    display: flex;
    flex-wrap: wrap;
    width: 100%;
    height: 100%;
}
.sidebar{
    background: #0e8e87;
    color: white;
    width: 15%;
    height: 100%;
    padding: 2%;
}

.menu{
    margin-top: 50px;
}
.menu h3{
    font-size: 15px;
}
.menu a{
    display: block;
    text-decoration: none;
    color: white;
    margin-top: 10px;
}
```

```
.content{
    display: flex;
    flex-wrap: wrap;
    flex-direction:column;
    width: 85%;
    padding: 10px;

}
.filtros{
    background: #ffffff;
    padding: 10px;
    border: 3px;
    border-radius: 10px 0px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.calendario{
    display: flex;
    width: 100%;
    height: fit-content;
    justify-content: center;

}
.calendario p{
    width: fit-content;
    height: fit-content;
    font-size: small;
}
.filtrodata{
    width: fit-content;
    margin-bottom: 0;
    margin-inline: 20px;

}

.resumomedicoes{
    display: flex;
    justify-content: space-between;
    width: 100%;
    height: 60px;
    margin-top: 10px;

}

.boxmedmax{
    background-color: bisque;
    width: 24%;
    height: 100%;
    padding: 10px;
    border-radius: 6px;
    color: white;
    background: linear-gradient(45deg,#054d49,#14A098);
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.boxmedmax p{
    font-size: 10px;
}
.graficos{
    display: flex;
    justify-content: space-around;
```

```
    text-align: center;
    margin-top: 10px;
}
.painelgrafico{
    width: 100%;
    background: #ffffff;
    border-radius: 10px;
    padding: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.btnfiltro{
    padding-inline: 10px;
    height: 25px;
    border-radius: 8px;
    border: 0px;
}
.btnfiltro:hover{
    background: #054d49;
    color: white;
}
```

c. Código JS

```
// --------------- CONFIGURAÇÃO DO FIREBASE --------------- //

// Import the functions you need from the SDKs you need
import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-app.js";
import { getAnalytics } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-analytics.js";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyAWPDemS-kFBcBJb3e02EypnKD3z9vVn48",
  authDomain: "analisador-de-energia-89d60.firebaseapp.com",
  projectId: "analisador-de-energia-89d60",
  storageBucket: "analisador-de-energia-89d60.appspot.com",
  messagingSenderId: "806459782406",
  appId: "1:806459782406:web:04f0af6e6d3bef909a5a87",
  measurementId: "G-0BXMQXPDQF"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

// --------------- FIM DA CONFIGURAÇÃO DO FIREBASE --------------- //

// --------------- INICIO DA ITERAÇÃO COM O BD --------------- //

import {getDatabase, ref, onChildAdded} from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-database.js";
const db = getDatabase();

const banco = ref(db);

var graphPotencias = document.getElementById('graphPotencias');

const dataInicialFiltro = document.getElementById('dataInicialFiltro');
const dataFinalFiltro = document.getElementById('dataFinalFiltro');
```

```
const excluirFiltros = document.getElementById('excluirFiltros');
const aplicarFiltro = document.getElementById('aplicarFiltro');

const maxPotenciaAtiva = document.getElementById('maxPotenciaAtiva');
const maxPotenciaReativa = document.getElementById('maxPotenciaReativa');
const maxPotenciaAparente = document.getElementById('maxPotenciaAparente');
const minFP = document.getElementById('minFP');

var maxMedidas =
{TimestampTensao:0,Tensao:0,TimestampCorrente:0,Corrente:0,TimestampPotenciaA
tiva:0,PotenciaAtiva:0,TimestampPotenciaReativa:0,PotenciaReativa:0,Timestamp
PotenciaAparente:0,PotenciaAparente:0,consumo:0,FP:0, TimestampFP:0};
maxPotenciaAtiva.innerHTML = "<h5>Potência Ativa (máxima):
Carregando...</h5><p>em "+moment(new Date()).format("D/MM/YYYY -
HH:mm:ss")+"<p>"
maxPotenciaReativa.innerHTML = "<h5>Potência Reativa (máxima):
Carregando...</h5><p>em "+moment(new Date()).format("D/MM/YYYY -
HH:mm:ss")+"<p>"
maxPotenciaAparente.innerHTML = "<h5>Potência Aparente (máxima):
Carregando...</h5><p>em "+moment(new Date()).format("D/MM/YYYY -
HH:mm:ss")+"<p>"
minFP.innerHTML = "<h5>Fator de Potência (mínimo): Carregando...</h5><p>em
"+moment(new Date()).format("D/MM/YYYY - HH:mm:ss")+"<p>"


function gerarParOrdenado(date,valor,listaSaida){
    let dataehora = date * 1000;
    listaSaida.push({x: dataehora, y: valor});

}

var config = {
    animation:false,
    plugins:{
        streaming:{
            duration: 100000,
            ttl: 9999999999999999,
            frameRate: 60,
        }
    },
    scales: {
        x:{
            type:'realtime',
            time: {
                unit: 'minute',
            },
            ticks: {
                maxTicksLimit: 5,
            }

        },
        y: {
            title:{
                display:true,
                text:'Potência',
            },
            beginAtZero: false,

        },
        y2:{
            title:{
```

```
                display:true,
                text:'Fator de Potência',
            },
            position:'right',
            min: 0,
            max: 1,

        },
    }
}
const gPotencias = new Chart(graphPotencias, {
    type: 'line',
    data: {
        datasets: [{
            label: 'Potência Ativa (W)',
            data: [],
            borderWidth: 1,
            pointRadius: 0,
            tension: 0.4,
        },
        {
            label: 'Potência Reativa (VAr)',
            data: [],
            borderWidth: 1,
            pointRadius: 0,
            tension: 0.4,

        },
        {
            label: 'Potência Aparente (VA)',
            data: [],
            borderWidth: 2,
            pointRadius: 0,
            tension: 0.4,
            fill: false,
        },
        {
            label: 'Fator de Potencia',
            data: [],
            borderWidth: 1,
            pointRadius: 0,
            tension: 0.4,
            yAxisID: 'y2'

        }]
    },
    options: config,
});


onChildAdded(banco, (snapshot)=>{
    const data = snapshot.val();


gerarParOrdenado(data.Timestamp,(-1*data.PotenciaAtiva),gPotencias.data.datas
ets[0].data);

gerarParOrdenado(data.Timestamp,data.PotenciaReativa,gPotencias.data.datasets
[1].data);
```

```
gerarParOrdenado(data.Timestamp,data.PotenciaAparente,gPotencias.data.dataset
s[2].data);

gerarParOrdenado(data.Timestamp,(-1*data.FP),gPotencias.data.datasets[3].data
);

    if (data.PotenciaAtiva > maxMedidas.PotenciaAtiva){
        maxMedidas.TimestampPotenciaAtiva = data.Timestamp;
        maxMedidas.PotenciaAtiva = data.PotenciaAtiva;
        maxPotenciaAtiva.innerHTML = "<h5>Potência Ativa (máxima):
"+maxMedidas.PotenciaAtiva+"W</h5><p>em "+moment(new
Date(maxMedidas.TimestampPotenciaAtiva*1000)).format("D/MM/YYYY -
HH:mm:ss")+"<p>";
    }
    if (data.PotenciaReativa > maxMedidas.PotenciaReativa){
        maxMedidas.TimestampPotenciaReativa = data.Timestamp;
        maxMedidas.PotenciaReativa = data.PotenciaReativa;
        maxPotenciaReativa.innerHTML = "<h5>Potência Reativa (máxima):
"+maxMedidas.PotenciaReativa+"VAr</h5><p>em "+moment(new
Date(maxMedidas.TimestampPotenciaReativa*1000)).format("D/MM/YYYY -
HH:mm:ss")+"<p>";
    }
    if (data.PotenciaAparente > maxMedidas.PotenciaAparente){
        maxMedidas.TimestampPotenciaAparente = data.Timestamp;
        maxMedidas.PotenciaAparente = data.PotenciaAparente;
        maxPotenciaAparente.innerHTML = "<h5>Potência Aparente (máxima):
"+maxMedidas.PotenciaReativa+"VA</h5><p>em "+moment(new
Date(maxMedidas.TimestampPotenciaAparente*1000)).format("D/MM/YYYY -
HH:mm:ss")+"<p>";
    }
    if (data.FP > maxMedidas.FP){
        maxMedidas.TimestampFP = data.Timestamp;
        maxMedidas.FP = data.FP;
        minFP.innerHTML = "<h5>Fator de Potência (mínimo):
"+maxMedidas.FP+"</h5><p>em "+moment(new
Date(maxMedidas.TimestampFP*1000)).format("D/MM/YYYY - HH:mm:ss")+"<p>";
    }
});

window.setInterval(()=>{
    gPotencias.update();
},2000);

function filtroChartX(chart,padrao,input){
    if (padrao === "min"){
        config.scales.x.min = input;
    }else{
        config.scales.x.max = input;
    }
    config.scales.x.type = "time";
    chart.update();
}

aplicarFiltro.addEventListener("click",(event)=>{
    if (dataInicialFiltro.value != ""){
        let dataInicial = new Date(dataInicialFiltro.value).getTime();
        filtroChartX(gPotencias,"min",dataInicial);
    }
    if (dataFinalFiltro.value !=""){
        let dataFinal = new Date(dataFinalFiltro.value).getTime()
```

```
        filtroChartX(gPotencias,"max",dataFinal);
    }
    config.plugins.streaming.duration = 10000000;
    gPotencias.update();
});

excluirFiltros.addEventListener("click",(event)=>{
    dataInicialFiltro.value ="";
    dataFinalFiltro.value = "";
    config.scales.x.type = "realtime";
    gPotencias.update();
    config.plugins.streaming.duration = 100000;
});
```

C. Página Medições

a. Código HTML

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Analisador TCC</title>
    <link rel="stylesheet" href="medicoes.css"/>
</head>

<body>
    <!-- HEADER -->
    <header class="topo">
    <!-- LOGOMARCA -->
        <h4>Analisador de Energia</h4>
    </header>
    <!-- FIM DO HEADER -->
    <section class="main">
        <div class="sidebar">
            <div class="menu">
                <h3>Painel de Controle</h3>
                <hr>
                <a href="/index.html">Página Inicial</a>
                <a href="/consumo/consumo.html">Consumo</a>
                <a href="/medicoes/medicoes.html">Medições</a>
                <a href="/potencias/potencias.html">Potências</a>
            </div>
            <div class="menu">
                <h3>Configurações</h3>
                <hr>
                <a href="/instrucoes/instrucoes.html">Instruções</a>
                <a href="/parametros/parametros.html">Parâmetros</a>
                <a href="/suporte/suporte.html">Suporte</a>

            </div>
        </div>
        <!-- INICIO DO CONTENT (CONTEÚDO DO DASHBOARD) -->
        <div class="content">
            <!-- INICIO DO FILTRO CALENDARIO -->
            <div class="filtros">
                <h5>Período de observação</h5>
                <div class="calendario">
                    <p>Início:</p><input type="datetime-local"
class="filtrodata" id="dataInicialFiltro"></input>
```

```
                    <p>Fim:</p><input type="datetime-local"
class="filtrodata" id="dataFinalFiltro"></input>
                    <button class="filtrodata btnfiltro"
id="aplicarFetch">Atualizar Medições</button>
                    <button class="filtrodata btnfiltro"
id="excluirFiltros">Excluir Filtros</button>
                </div>
            </div>

            <!-- INICIO DO PAINEL MEDIÇÕES MÁXIMAS -->
            <div class="resumomedicoes">
                <div class="boxmedmax"><h3>Medições</h3></div>
            </div>
            <!-- INCIO DOS GRÁFICOS -->
            <div>
                <!-- INCIO DO DIV DO GRÁFICO TENSÃO -->
                <table id="tabelaDados">
                    <tr><td>Tempo</td><td>Tensão(V)</td><td>Tensão
Eficaz(Vef)</td><td>Corrente(A)</td><td>Corrente Eficaz(Aef)</td><td>Potência
Ativa(kW)</td><td>Potência Reativa(kVar)</td><td>Potência
Aparente(kVA)</td><td>Fator de Potência</td></tr>
                </table>
            </div>
        </div>
    </section>

    <!-- IMPORTAÇÕES DE JS -->
    <script src="https://kit.fontawesome.com/9ade8f3144.js"
crossorigin="anonymous"></script> <!-- PACK DE ICONS -->

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <!--
BIBLIOTECA CHART JS (GRÁFICOS) -->

    <!-- ADAPTADOR DE DATAS PRO CHART JS -->
    <script src="https://cdn.jsdelivr.net/npm/luxon@^2"></script>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-luxon@^1"></script>


    <!-- MANIPULAÇÃO DE DATA E HORA -->
    <script src="https://momentjs.com/downloads/moment.min.js"></script>
    <script
src="https://momentjs.com/downloads/moment-timezone-with-data.min.js"></scrip
t>
    <script
src="https://cdn.jsdelivr.net/npm/chartjs-adapter-date-fns/dist/chartjs-adapt
er-date-fns.bundle.min.js"></script>

    <!-- ARQUIVO JAVASCRIPT CONTENDO O FIREBASE E CONFIGURAÇÕES -->
    <script type="module" src="medicoes.js"></script>
</body>
</html>
```

b. Código CSS

```
*{
    margin:0;
    padding: 0;
    box-sizing: border-box;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
    width: 100%;
}
html,body{
```

```
    height: 100%;
    background: #F5F5F5;
}
.topo{
    display: flex;
    flex-wrap: nowrap;
    width: 100%;
    height: 7%;
    background: #0e8e87;
    padding: 10px;
    text-align: center;
    color: white;
}

.main{
    display: flex;
    flex-wrap: wrap;
    width: 100%;
    height: 100%;
}
.sidebar{
    background: #0e8e87;
    color: white;
    width: 15%;
    height: 100%;
    padding: 2%;
}

.menu{
    margin-top: 50px;
}
.menu h3{
    font-size: 15px;
}
.menu a{
    display: block;
    text-decoration: none;
    color: white;
    margin-top: 10px;
}
.content{
    display: flex;
    flex-wrap: wrap;
    flex-direction:column;
    width: 85%;
    padding: 10px;

}
.filtros{
    background: #ffffff;
    padding: 10px;
    border: 3px;
    border-radius: 10px 0px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
.calendario{
    display: flex;
    width: 100%;
    height: fit-content;
    justify-content: center;
```

```
}
.calendario p{
    width: fit-content;
    height: fit-content;
    font-size: small;
}
.filtrodata{
    width: fit-content;
    margin-bottom: 0;
    margin-inline: 20px;
}
.btnfiltro{
    padding-inline: 10px;
    height: 25px;
    border-radius: 8px;
    border: 0px;
}
.btnfiltro:hover{
    background: #054d49;
    color: white;
}
#tabelaDados {
    margin-top: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    border-radius: 6px;
    padding: 1%;
    border: 1px solid #333;
}
#tabelaDados td{
    text-align: center;
    width: fit-content;
    border: 1px solid #333;
}
.resumomedicoes{
    display: flex;
    justify-content: center;
    width: 100%;
    height: 60px;
    margin-top: 10px;
}
.boxmedmax{
    background-color: bisque;
    width: 100%;
    height: 100%;
    padding: 10px;
    border-radius: 6px;
    color: white;
    background: linear-gradient(45deg,#054d49,#14A098);
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
    text-align: center;
}
.boxmedmax p{
    font-size: 10px;
}
.resumoMedicoes{
    display: flex;
    flex-wrap: wrap;
    text-align: center;
    width: 100%;
    background: #ffffff;
    border-radius: 10px;
```

```
    padding: 10px;
    box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
}
```

c. Código JS

```
// --------------- CONFIGURAÇÃO DO FIREBASE --------------- //

// Import the functions you need from the SDKs you need
import { initializeApp } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-app.js";
import { getAnalytics } from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-analytics.js";
// TODO: Add SDKs for Firebase products that you want to use
// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyAWPDemS-kFBcBJb3e02EypnKD3z9vVn48",
  authDomain: "analisador-de-energia-89d60.firebaseapp.com",
  projectId: "analisador-de-energia-89d60",
  storageBucket: "analisador-de-energia-89d60.appspot.com",
  messagingSenderId: "806459782406",
  appId: "1:806459782406:web:04f0af6e6d3bef909a5a87",
  measurementId: "G-0BXMQXPDQF"
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);

// --------------- FIM DA CONFIGURAÇÃO DO FIREBASE --------------- //

// --------------- INICIO DA ITERAÇÃO COM O BD --------------- //

import {getDatabase, ref,get} from
"https://www.gstatic.com/firebasejs/10.1.0/firebase-database.js";
const db = getDatabase();

const banco = ref(db);

const dataInicialFiltro = document.getElementById('dataInicialFiltro');
const dataFinalFiltro = document.getElementById('dataFinalFiltro');
const excluirFiltros = document.getElementById('excluirFiltros');
const atualizarMedicoes = document.getElementById('aplicarFetch');
const tabelaDados = document.getElementById('tabelaDados');


atualizarMedicoes.addEventListener("click",(event)=>{
    console.log(new Date())
    var tabela="";
    get(banco).then((snapshot)=>{
        const snap = snapshot.val();
        tabelaDados.innerHTML="<tr><td>Tempo</td><td>Tensão(V)</td><td>Tensão
Eficaz(Vef)</td><td>Corrente(A)</td><td>Corrente Eficaz(Aef)</td><td>Potência
Ativa(kW)</td><td>Potência Reativa(kVar)</td><td>Potência
Aparente(kVA)</td><td>Fator de Potência</td></tr>";
        for (const data in snap){
            if (dataInicialFiltro.value !="" && snap[data].Timestamp*1000 <
new Date(dataInicialFiltro.value).getTime()){
```

```
                console.log("Deletado: "+new
Date(dataInicialFiltro.value).getTime());
                continue
            }
            if (dataFinalFiltro.value !="" && snap[data].Timestamp*1000 > new
Date(dataFinalFiltro.value).getTime()){
                console.log("Deletado: "+new
Date(dataFinalFiltro.value).getTime());
                continue
            }
            tabela += "<tr><td>"+moment(new
Date(snap[data].Timestamp*1000)).format("D/MM/YYYY -
HH:mm:ss")+"</td><td>"+snap[data].Tensao+"</td><td>"+snap[data].Vef+"</td><td
>"+snap[data].Corrente+"</td><td>"+snap[data].Aef+"</td><td>"+snap[data].Pote
nciaAtiva+"</td><td>"+snap[data].PotenciaReativa+"</td><td>"+snap[data].Poten
ciaAparente+"</td><td>"+snap[data].FP+"</td></tr>";
        }
        tabelaDados.innerHTML += tabela;
    })
});

function filtroChartX(padrao,input){
    if (padrao === "min"){
        //config.scales.x.min = input;
        console.log("Escala Mínima: "+input)
    }else{
        //config.scales.x.max = input;
        console.log("Escala Máxima: "+input)

    }
}

dataInicialFiltro.addEventListener("change",(event)=>{
    let dataInicial = new Date(dataInicialFiltro.value).getTime()
    filtroChartX("min",dataInicial);
    filtroChartX("min",dataInicial);
});

dataFinalFiltro.addEventListener("change",(event)=>{
    let dataFinal = new Date(dataFinalFiltro.value).getTime()
    filtroChartX("max",dataFinal);
    filtroChartX("max",dataFinal);
});

excluirFiltros.addEventListener("click",(event)=>{
    dataInicialFiltro.value ="";
    dataFinalFiltro.value = "";
    filtroChartX("min","");
    filtroChartX("min","");
    filtroChartX("max","");
    filtroChartX("max","");

});
```