UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE REGISTRADOR DE PULSOS CONECTADO À INTERNET PARA MEDIDORES DE ENERGIA ELETRÔNICOS

MARCELO DE PAULA BRAGA JÚNIOR

UBERLÂNDIA-MG
MARÇO 2022

DESENVOLVIMENTO DE REGISTRADOR DE PULSOS CONECTADO À INTERNET PARA MEDIDORES DE ENERGIA ELETRÔNICOS

Trabalho Final de Curso apresentado a Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Área de conhecimento: Engenharia Elétrica

Orientador: Prof. Dr. José Rubens Macedo Júnior

UBERLÂNDIA-MG

FACULDADE DE ENGENHARIA ELÉTRICA

MARÇO 2022

DESENVOLVIMENTO DE REGISTRADOR DE PULSOS CONECTADO À INTERNET PARA MEDIDORES DE ENERGIA ELETRÔNICOS

MARCELO DE PAULA BRAGA JÚNIOR

Trabalho Final de Curso apresentado a Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia como requisito parcial à obtenção do título de Bacharel em Engenharia Elétrica.

Aprovado em: 25/03/2022

Banca de avaliação:

Orientador: Prof. Dr. José Rubens Macedo Júnior

Avaliador: Prof. Dr. Isaque Nogueira Gondim

Avaliador: Prof. M. Sc. Eduardo Tavares Silvério

AGRADECIMENTOS

Aos docentes da Faculdade de Engenharia Elétrica por todos os ensinamentos ao longo da graduação. Em especial ao meu orientador Prof. José Rubens pelo apoio para a realização deste trabalho.

Aos meus pais, Marcelo e Sueli, e as minhas irmãs, Marcela e Bianca, por acreditarem em mim e por sempre terem incentivado meus estudos.

Aos meus amigos e colegas de curso por todos os momentos de estudos, pelos conselhos e pelas experiências que vivemos juntos. Obrigado por terem tornado a graduação mais leve.

RESUMO

O presente trabalho tem como objetivo apresentar o desenvolvimento de um sistema para obter a curva de carga e realizar o teste de calibração de medidores de energia eletrônicos. O sistema é composto por um dispositivo conectado ao medidor, o qual é responsável pela leitura dos pulsos e envio dos dados via rede Wi-Fi para um banco de dados na nuvem. O dispositivo foi construído utilizando a placa de desenvolvimento NodeMCU e um sensor de cor e movimento para a detecção dos pulsos. Para o controle do registrador e para a visualização dos dados foi desenvolvido um aplicativo móvel. O aplicativo exibe a curva de carga em tempo real e oferece ferramentas para estimar o custo da energia consumida em diferentes modalidades tarifárias. Diante do exposto, o sistema foi testado em laboratório em três diferentes medidores de energia eletrônicos. Para a análise da precisão do teste de calibração e da obtenção da curva de carga, também foi coletado dados com outros equipamentos que possuem funcionalidades parecidas. Por meio dos testes realizados, foi possível comprovar a eficiência do sistema em coletar dados relevantes para os consumidores utilizando a constante Kh.

Palavras-chave: constante Kh; medidores de energia; curva de carga; teste de calibração; Internet das Coisas;

ABSTRACT

The present work aims to present the development of a system for getting the load curve and testing the calibration of electronic meters. The system is composed of a connected device on a meter for which it is responsible for the pulse reading and sending the data with Wi-Fi for a cloud database. The device was built using the NodeMCU, a hardware development environment, and a sensor of color and gesture sensor for detecting the pulses. To control the recorder and visualize the data was developed a mobile app. The app displays the real-time load curve and provides tools for estimating the cost of the energy consumed. Based on the above, the system was tested in the laboratory with three electronic meters distinct. To analyze the precision of the calibration test and the getting load curve, was collected too data with others equipment with similar features. Through the tests, was proven the efficiency of the system in collecting relevant data for the consumers using the Kh constant.

Key-words: Kh constant; energy meters; load curve; calibration test; Internet of things;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PRORET	*Procedimentos de RegulaçãoTarifária*
LED	*Light Emitting Diode*
ANEEL	*Agência Nacional de Energia Elétrica*
IoT	*Internet das Coisas*
APK	*Android Application Pack*
JSON	*JavaScript Object Notation*
SoC	*System On Chip*
IDE	*Integrated Development Environment*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
API	*Application Programming Interface*
UV	*Ultravioleta*
IR	*Infravermelho*
I2C	*Inter-Integrated Circuit*
VL	*Load Voltage*
VCC	*Voltage Common Collector*
INT	*Interrupção*
GND	*Ground*
SPI	*Serial Peripheral Interface*
OLED	*Organic Light Emitting Diode*
SCL	*Serial Clock*
SDA	*Serial Data*
LADEE	*Laboratório de Distribuição de Energia Elétrica*

SUMÁRIO

1 INTRODUÇÃO

O fornecimento de energia elétrica é um serviço essencial à população. Para garantir isso, o sistema elétrico brasileiro é responsável pela geração, transmissão e a distribuição da energia elétrica até o consumidor, seja uma residência, comércio ou indústria. Para manter a qualidade e a eficiência desse sistema, os consumidores devem pagar uma tarifa mensal de acordo com a quantidade de energia consumida. A quantificação mensal da energia consumida é feita por meio dos medidores de energia elétrica, esses devem estar presentes em todas as unidades consumidoras.

Os medidores de energia podem ser classificados de acordo com sua tecnologia, sendo eles os medidores eletromecânicos e eletrônicos. O tipo mais comum atualmente no setor elétrico brasileiro é o eletromecânico. No entanto, a tendência é que o medidor eletromecânico seja gradualmente substituído pelo eletrônico por apresentar melhor custo benefício.

O desenvolvimento tecnológico e a substituição dos medidores de energia têm como objetivo não só o melhor custo benefício, mas também a maior precisão no registro da quantidade de energia consumida. A quantificação precisa nos medidores é de fundamental importância para garantir a cobrança da tarifa justa dos consumidores e evitar prejuízos no faturamento das concessionárias de energia. Dessa forma, para evitar erros na quantificação da energia consumida, os medidores de energia presente nas unidades consumidoras devem ser devidamente calibrados. Para garantir isso, esses equipamentos de medição possuem a constante Kh, também conhecida como constante de calibração ou verificação. Essa constante fornece a quantidade de energia que o medidor está registrando no momento.

O conhecimento da quantidade de energia consumida é de grande importância tanto para a concessionária tanto para os consumidores. Nos medidores de energia eletromecânicos é registrado somente uma grandeza, a quantidade do consumo de energia ativa acumulado. Já nos medidores eletrônicos, eles possuem a memória de massa, essa é responsável por armazenar a energia consumida ao longo do tempo em intervalos definidos. Para ter acesso a esses dados, é necessário a solicitação para a concessionária de energia responsável. Dessa forma, para a maioria dos medidores de energia atuais, é necessário o uso de ferramentas adicionas para obter os dados da quantidade de energia consumida ao longo do tempo de forma independente.

O registro da demanda de energia de um consumidor em função do tempo em período diário é denominado curva de carga. Para as concessionárias de energia, a curva de carga fornece dados importantes para o planejamento da rede. Para os consumidores o conhecimento do seu perfil de consumo também é importante, pois fornece informações que

auxiliam na execução de projetos elétricos e a adoção de medidas de eficiência energética. Além disso, também pode ser fundamental para auxiliar na escolha da modalidade tarifária para econômica de acordo com o perfil de consumo.

Dessa forma, dentro do contexto apresentado anteriormente, é importante ressaltar que por meio do uso de ferramentas adequadas, a constante Kh pode auxiliar não só na calibração dos medidores, mas também fornecer informações do consumo de energia ao longo do tempo. Neste sentido, o objetivo deste trabalho é apresentar o desenvolvimento de um dispositivo para auxiliar na calibração e obtenção a curva de carga utilizando a constante Kh em medidores de energia eletrônicos.

Por meio do dispositivo desenvolvido, será possível obter a curva de carga de um consumidor em tempo real. Dessa forma, para que o sistema seja conectado a internet e tenha uma característica responsiva, foi necessário a aplicação de conceitos de internet das coisas. Para a leitura da constante Kh dos medidores de energia eletrônicos, foi utilizado um sensor conectado a uma placa de desenvolvimento. Esta placa, além de fornecer o microcontrolador do dispositivo, também é responsável por enviar os dados para um banco de dados na nuvem. Para o controle do registrador e para a visualização dos dados foi desenvolvido um aplicativo móvel. O aplicativo exibe a curva de carga em tempo real e oferece ferramentas para estimar o custo da energia consumida em diferentes modalidades tarifárias.

2 FUNDAMENTOS TEÓRICOS

2.1 MEDIDORES DE ENERGIA ELÉTRICA

No setor elétrico brasileiro os medidores de energia eletromecânicos surgiram entre as décadas de 1980 e 1990. Atualmente, eles ainda estão presentes na maioria das unidades consumidas de energia. O seu funcionamento é baseado nos princípios da indução eletromagnética. As interações de fluxos magnéticos fazem o disco presente do medidor se movimentar com as correntes elétricas, cada giro do disco representa certa quantidade de energia consumida.

O desenvolvimento de novas tecnologias permitiu a criação dos medidores de energia eletrônicos. Esse tipo de medidor permite que a quantificação da energia seja mais precisa. Além disso, oferece a possibilidade de conferir a energia consumida de forma mais rápida e fácil. Nos medidores eletrônicos a medição da energia é feita por meio de transdutores de tensão e de corrente.

De acordo com [1], a medição é definida como o processo realizado por um equipamento que possibilite a quantificação e o registro de grandezas elétricas associadas à geração ou consumo de energia elétrica, assim como à potência ativa ou reativa. No sistema elétrico essa medição é de responsabilidade dos medidores de energia elétrica que devem estar presentes em todas as unidades consumidoras. Ele é fundamental para a realização da cobrança da fatura de acordo com a quantidade de energia consumida.

O faturamento do consumo de energia ocorre através da quantificação realizada pelo medidor durante o período de 30 dias. As tarifas aplicadas aos consumidores de energia possuem diferentes grupos tarifários. Estes grupos são: o Grupo A e o Grupo B, essa classificação é definida de acordo com o Módulo 7 dos Procedimentos de Regulação Tarifária – PRORET [2].

O Grupo A são as unidades consumidoras da Alta Tensão (Subgrupos A1, A2 e A3), Média Tensão (Subgrupos A3a e A4), e de sistemas subterrâneos (Subgrupo AS). Já o Grupo B, são as unidades consumidoras da Baixa Tensão, das Classes Residencial (Subgrupo B1), Rural (B2), Demais Classes (B3) e Iluminação Pública (B4). O foco deste trabalho será nos medidores de energia utilizados pelo Grupo B.

Dentro do Grupo B, existe a modalidade de Tarifa Convencional e a Tarifa Branca. A modalidade Convencional possui valor único de tarifa independente do horário do consumo da energia elétrica. Já na Tarifa Branca, o valor pago na tarifa depende do horário em que a energia foi consumida. Para a adesão da Tarifa Branca, o consumidor deve utilizar o medidor eletrônico, pois ele permite o registro da energia consumida em

função do tempo.

É importante destacar que os medidores de energia possuem papel fundamental na aplicação de tarifação no sistema elétrico. Além disso, para garantirem o faturamento e a cobrança correta, são responsáveis por realizarem a medição precisa do consumo.

Neste contexto, os medidores de energia são construídos para operarem em condições senoidais e equilibradas. No entanto, o avanço do uso de componentes eletrônicos no sistema elétrico brasileiro provocou um aumento nos níveis de distorção harmônica na rede elétrica. Por consequência disso, os medidores de energia podem apresentar desvios na medição e consequentemente erros em cobranças de fatura.

Em estudos realizados por [3], foi comprovado a existência de desvios de medição em condições distorcidas e desequilibradas para medidores eletromecânicos e eletrônicos. Nesse estudo, para uma mesma forma de onda aplicada em diferentes medidores foi apresentado desvios significativos. Assim, foi constatado a existência da falta isonomia para a medição de energia elétrica ativa.

Dessa forma, a calibração correta dos medidores de energia é extrema importância. Pois garante a cobrança justa para os consumidores e o faturamento correto para as concessionárias de energia, de forma que nenhuma das partes tenha prejuízos. Assim, destaca-se a relevância de equipamentos para a realização e verificação da calibração nos medidores de energia.

2.1.1 Constante Kh

A realização dos testes de calibração nos medidores de energia elétrica é feita por meio da constante Kh. Ela está presente em todos os tipos de medidores e também pode ser chamada de constante de verificação ou calibração. Nos medidores eletromecânicos a constante Kh é representada pela rotação completa do disco do medidor, sendo expressa em Wh/rotação ou varh/rotação. Já nos medidores eletrônicos, a constante de verificação é representada por pulsos de LED (*Light Emitting Diode*) emitidos pelo medidor, sendo expressa em Wh/pulso ou varh/pulso.

O conhecimento do pulso emitido pelo medidor de energia eletrônico, responsável por representar a constante Kh, é de fundamental importância para este trabalho. A partir da leitura dos pulsos, será possível obter a quantidade de energia elétrica medida para desenvolver as funcionalidades do sistema proposto. Esse dado será utilizado para a verificação da precisão de leitura do medidor e para a obtenção da curva de carga.

2.1.2 Medidor Eletrônico

A quantificação da energia consumida nos medidores eletrônicos ocorre através da utilização de circuitos integrados. Com o desenvolvimento da tecnologia e da eletrônica, a tendência é que os medidores eletrônicos sejam cada vez mais utilizados no setor elétrico por apresentarem melhor custo benefício.

Os medidores de energia eletrônicos, são regulamentados pela Agência Nacional de Energia Elétrica (ANEEL) para as unidades consumidoras do Grupo B desde 2012. Segundo a ANEEL, em [4], a medição eletrônica veio com a proposta de trazer mais benefícios para os consumidores, como consumo mais eficiente de energia, possibilidade de atendimento remoto pela concessionária, o melhor monitoramento da rede pela distribuidora e a oferta de novos serviços aos consumidores.

Um desses novos serviços que podem ser oferecidos é a Tarifa Branca. Nesta modalidade, o valor da tarifa é diferente de acordo com os períodos postos. Estes períodos são a Ponta em que possui a tarifa mais elevada, o Intermediário, em que possui a tarifa de valor intermediário e a Fora Ponta com a tarifa de menor valor. Nos finais de semana e feriados nacionais, o valor aplicado é sempre o da tarifa Fora de Ponta. Este tipo de tarifa visa reduzir os gastos com energia elétrica e, ao mesmo tempo, melhorar o fator de utilização das redes [5].

Além disso, o desenvolvimento dos medidores de energia eletrônicos permitiu dar início as discussões para a implantação dos chamados medidores de energia inteligentes. De acordo com [6], um sistema é inteligente quando possui sensoriamento, comunicação, aplicação de inteligência artificial e controle da base de dados obtidos. No sistema elétrico, isso permite aumentar a confiabilidade, gerenciamento de ativos e redução de custos operacionais.

No entanto, de acordo com [7], desde o início da implantação dos medidores eletrônicos no sistema elétrico brasileiro esses não eram dotados de inteligência adicional. O avanço tecnológico para os medidores eletrônicos se deu majoritariamente devido ao custo ter-se tornado inferior ao medidor eletromecânico. Assim, isso mostra que o aumento do uso de medidores eletrônicos não indica necessariamente que está havendo uma evolução na implantação dos medidores inteligentes.

Recentemente, tem-se notado a implantação de alguns medidores inteligentes por meio de programas de modernização da rede, como foi noticiado em [8]. Esses novos medidores permitem a leitura do consumo de forma remota e os consumidores podem acompanhar a quantidade de energia consumida diariamente. No entanto, é importante ressaltar que essas tecnologias estão presentes em apenas uma pequena parcela do sistema elétrico.

A utilização de medidores inteligentes é uma peça fundamental para a implantação das chamadas redes elétricas inteligentes. As redes inteligentes são caracterizadas por ser uma sobreposição de sistemas unificados de comunicação e controle em uma infraestrutura de energia elétrica existente [9]. Por fim, é importante destacar que o avanço da Internet das Coisas é responsável por prover dispositivos necessários para implementação de redes inteligentes.

2.1.3 Curva de Carga

A curva de carga representa o comportamento de uma carga ou o consumo de potência de uma carga variável suprida por uma fonte durante um período de análise [10]. A coleta desses valores registrados ocorre em intervalos de tempo pré-determinados, normalmente 15 minutos. Usualmente o período de registro é diário, mas em alguns casos pode ser para um período semanal, mensal ou anual. Na Figura 1, é apresentado a curva de carga de uma residência em que o consumo anual é de cerca de 330kWh por mês.

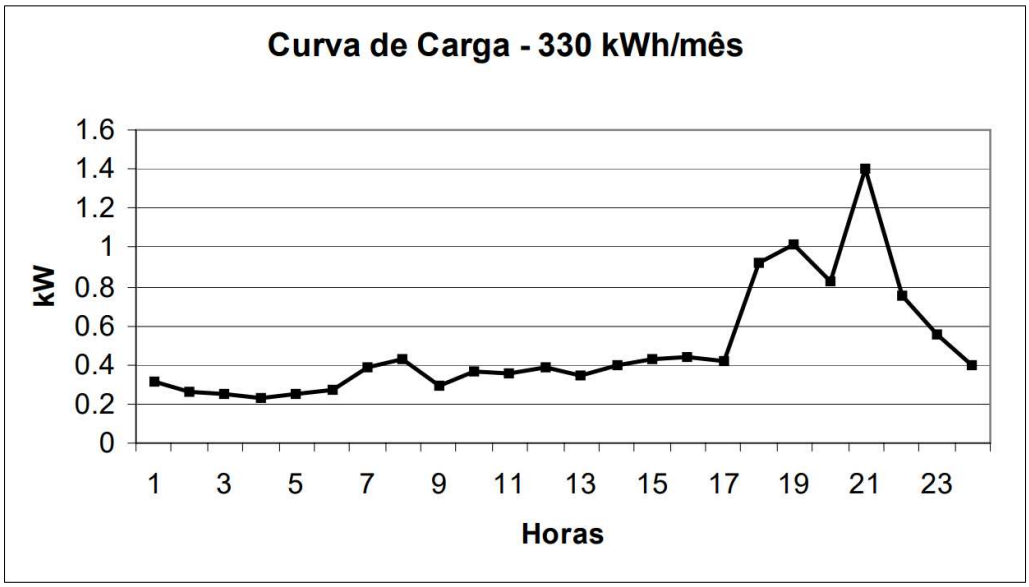


Figura 1 – Curva de Carga de um consumidor residencial [11].

Para o consumidor o conhecimento da curva de carga permite a análise mais detalhada do seu perfil de consumo. Essa informação permite melhor enquadramento tarifário, bem como auxiliar na aplicação de ações de eficiência energética por meio do gerenciamento de demanda. Dessa forma, a curva de carga é importante para o consumidor, pois colabora para a redução de tarifas e consumo [12].

2.2 INTERNET DAS COISAS

O avanço da Internet das Coisas (IoT) se deu devido a evolução tecnológica nas áreas de sistemas embarcados, microeletrônica, comunicação e sensoriamento. Isso permitiu uma gama de novas possibilidades de aplicações, como cidades inteligentes, saúde e casas inteligentes [13].

A Internet das Coisas pode ser definida, de acordo com [14], como a evolução da internet atual. Os objetos são conectados através de uma rede para coletar informações e interagir com o mundo físico. Além disso, usa os padrões existentes da internet para fornecer serviços de transferência de informação, análise, aplicações e comunicações.

A arquitetura de um sistema de Internet das Coisas pode ser descrita através dos seus componentes. Um dos principais componentes é o dispositivo físico, responsável por realizar uma coleta contínua de informações através de sensores. Os outros componentes podem ser classificados como a porta de comunicação, o local de armazenamento de dados na nuvem, a análise dos dados e a interface para o usuário. A organização destes componentes em um sistema é apresentada na Figura 2 a seguir.

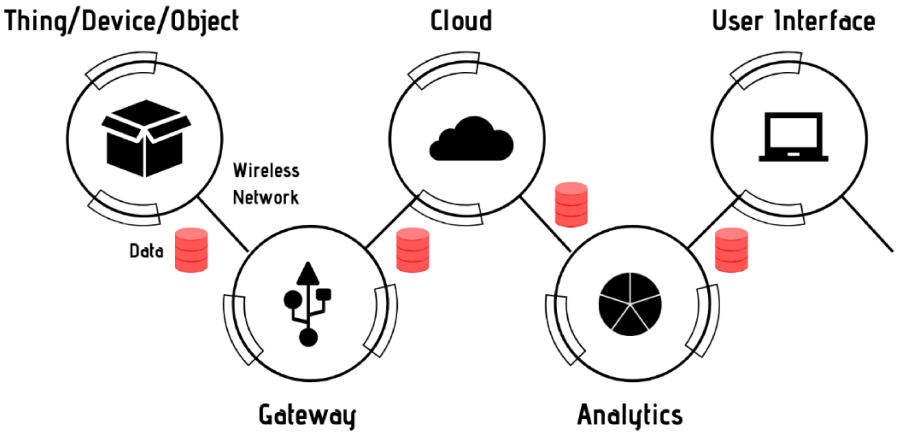


Figura 2 – Principais componentes de um sistema de Internet das Coisas [15].

É importante destacar também o papel fundamental da interface, é por meio dela que o usuário irá controlar o dispositivo físico que realiza a coleta das informações. Além disso, também exibe as informações tratadas pelo componente responsável pela análise dos dados.

3 DESENVOLVIMENTO

3.1 O APLICATIVO MOVÉL

No sistema proposto é necessário a utilização de uma interface para comunicar com o dispositivo físico acoplado no medidor de energia. A interface é responsável por enviar e receber dados de modo que facilite a interação do usuário com todo o sistema. Além disso, ela deve ser intuitiva, fácil de entender e navegar.

Dessa forma, para este projeto, é utilizado uma interface móvel devido a portabilidade e praticidade que oferece ao usuário. Uma interface móvel ou aplicativo, como é popularmente conhecida, é um software desenvolvido para smartphones ou tablets que podem ser instalados em sistemas Android ou iOS.

Atualmente, existem diversas linguagens de programação, *frameworks* e plataformas para o desenvolvimento de aplicações móveis. A seguir será apresentado todas as tecnologias utilizadas para o desenvolvimento do aplicativo do sistema.

3.1.1 Tecnologias utilizadas

- **React Native**

No processo de desenvolvimento de interfaces existem *frameworks* que oferecem ferramentas para ajudar na criação e na forma como os dados são exibidos. O *framework* é responsável por fornecer uma estrutura base para a criação da aplicação por meio de um conjunto de bibliotecas. Dessa forma, simplifica o processo de desenvolvimento e otimiza a utilização dos recursos.

Para o desenvolvimento da aplicação deste projeto foi escolhido o React Native. Um *framework* criado pelo Facebook que possibilita desenvolvimento de aplicativos móveis multiplataforma utilizando a linguagem Javascript. O React Native funciona como um facilitador para utilização da linguagem Javascript no desenvolvimento de aplicações móveis.

Dessa forma, a ferramenta é responsável por converter o código para a linguagem nativa do sistema operacional, o que torna a aplicativo mais fluido. Além do Javascript, também é possível utilizar na construção dos códigos a linguagem Typescript, um superconjunto sintático estrito de Javascript.

O React Native oferece a possibilidade de utilizar componentes para a criação de interfaces, como botões, caixas de entrada, componentes de listagem e entre outras coisas. Além disso, o *framework* é uma tecnologia de código aberto e permite que os desenvolvedores usem suas bibliotecas e estruturas gratuitamente, o que também incentiva

o desenvolvimento de mais ferramentas e bibliotecas para a comunidade. A principal biblioteca utilizada para o desenvolvimento da interface deste projeto é a React Native Paper. Ela oferece uma coleção de componentes seguindo as diretrizes do Material Design do Google.

- **Typescript**

O Typescript é uma linguagem de programação desenvolvida pela Microsoft com o objetivo de trazer melhorias no sistema de tipos para a linguagem Javascript. A linguagem representa um superconjunto do Javascript, ou seja, oferece todos os recursos do Javascript com uma camada adicional sobre eles.

O diferencial do TypeScript é o sistema de tipagem estática que a linguagem possui, assim, os tipos das variáveis devem ser definidos explicitamente no código. Dessa forma, para o desenvolvimento deste aplicativo foi escolhido o uso do Typescript, pois oferece a possibilidade de descobrir e corrigir erros em tempo real durante o desenvolvimento, o que oferece maior segurança para a aplicação.

- **Expo**

Além das tecnologias citadas, também é utilizado a plataforma Expo para auxiliar no desenvolvimento e na implantação. O Expo fornece ferramentas e serviços para facilitar a utilização do *framework* React Native. Com a utilização desta plataforma, ao começar o desenvolvimento do software, é possível ter acesso a uma estrutura inicial para o aplicativo com telas bases e navegação entre elas.

A plataforma também fornece ferramentas para acessar a aplicação no smartphone durante o desenvolvimento, possibilitando a realização de testes sem a necessidade de emulador. Além disso, também auxilia no processo de construção do aplicativo para o formato APK (*Android Application Pack*), um conjunto de pacotes para distribuir e instalar um aplicativo Android.

3.1.2 Interface

Com as ferramentas e tecnologias apresentadas, foi possível realizar o desenvolvimento da interface para o sistema proposto. A interface é o principal meio de operação do sistema pelo usuário. Por isso, deve fornecer de forma prática e acessível as funcionalidades de teste de calibração de obtenção de curva de carga.

Para ajudar na especificação da interface foi construído um diagrama de estados de navegação, o mesmo é apresentada na Figura 3. Essa ferramenta apresenta as telas que compõe o aplicativo e as ações realizadas pelo usuário para navegar entre elas. As transições de telas ocorrem por meio de componentes de ação, como o botão.

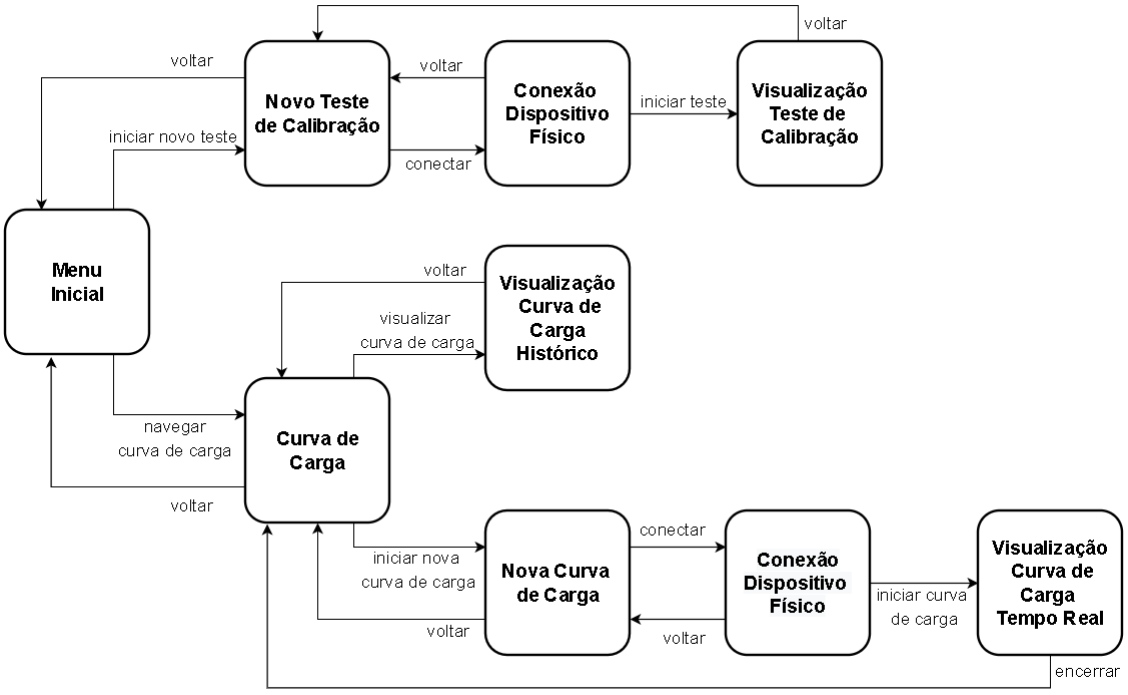


Figura 3 – Diagrama de estados de navegação.

3.2 BANCO DE DADOS

O banco de dados em projetos de Internet das Coisas é responsável por armazenar os dados de forma estruturada para que diferentes sistemas possam acessá-los. No projeto desenvolvido, o banco de dados recebe os dados do dispositivo físico conectado no medidor e também compartilhar com a interface móvel. Para que o sistema tenha uma característica responsiva, é importante a escolha de um banco de dados que ofereça sincronização em tempo real.

Dessa forma, para o desenvolvimento deste projeto, foi utilizado o Firebase Realtime Database. Uma plataforma desenvolvida pelo Google que fornece armazenamento de banco de dados não relacional na nuvem. Os dados são armazenados no formato JSON (*JavaScript Object Notation*), uma notação para estruturas de dados em formato de texto para serem utilizados em diferentes tipos de sistemas.

O Firebase utiliza um sistema de sincronização de dados. Assim, sempre que os dados são alterados, todos os dispositivos conectados recebem a atualização em tempo real. Além disso, a plataforma possui planos gratuitos para aplicações com até 1 GB de armazenamento.

Na Figura 4, é apresentado a estrutura do banco de dados no Firebase desenvolvido para o projeto. A estrutura segue o modelo de banco de dados não relacional, os dados são armazenados em pares de chave e valor, como em documentos JSON.

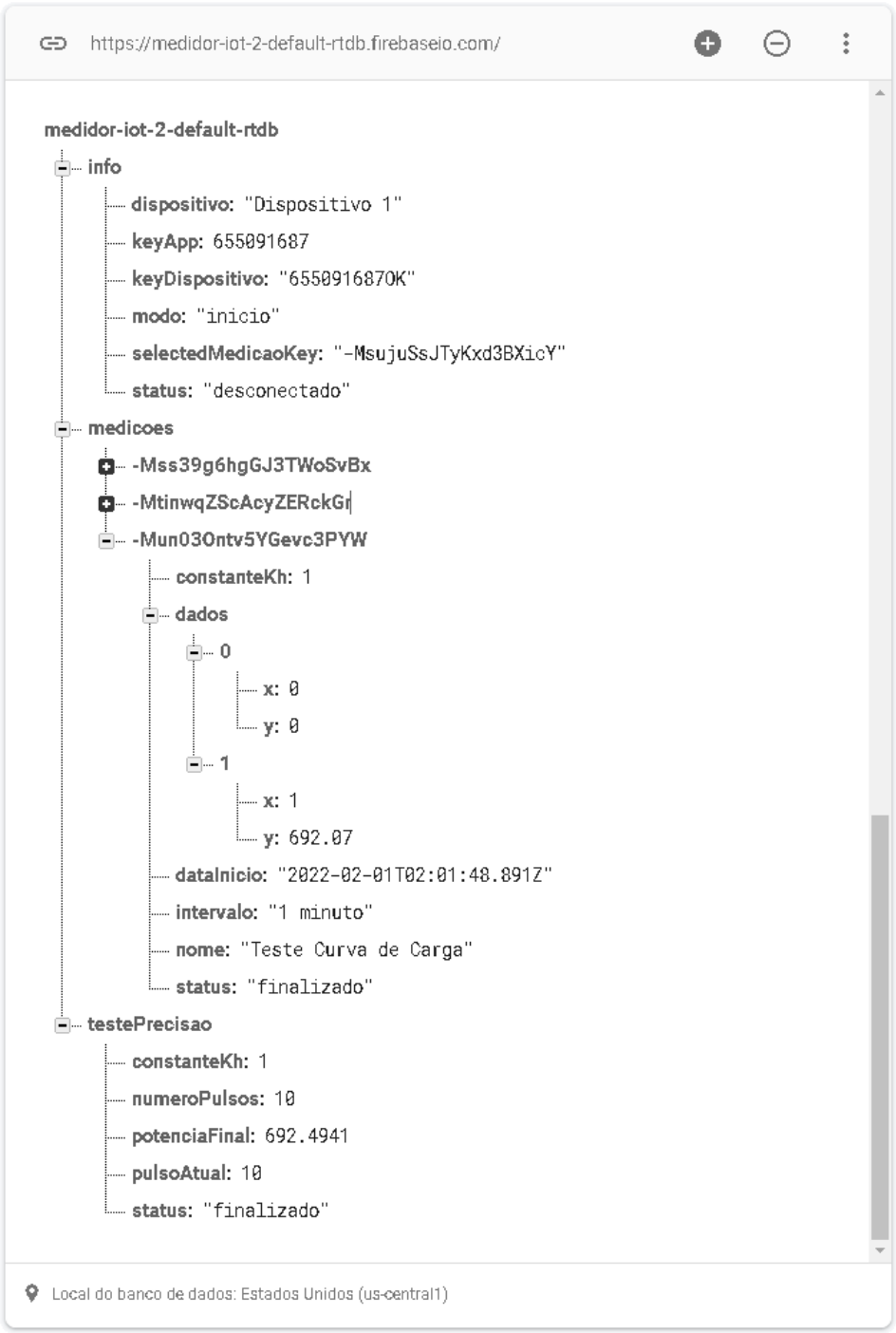


Figura 4 – Estrutura do banco de dados no Firebase.

A estrutura desenvolvida é utilizada para armazenar as informações das duas principais funcionalidades do sistema: o teste de calibração e a obtenção de curva de carga. Além disso, para a comunicação em tempo real da interface com o dispositivo físico foi necessário a adição de elementos para suportar esta funcionalidade, como a adição de campos que indicam o estado da interface ou do dispositivo físico.

3.3 DISPOSITIVO FÍSICO

A coleta dos dados é realizada por meio de um dispositivo físico acoplado na parte frontal do medidor de energia. O dispositivo será responsável por fazer a leitura dos pulsos de LED do medidor de energia e enviar os dados coletados para o banco de dados na nuvem.

A construção do registrador de pulsos foi realizada utilizando uma placa de desenvolvimento eletrônica que permite a conexão de sensores periféricos e módulos de conexão Wi-Fi. Além disso, todo o hardware foi pensado para possuir uma estrutura compacta e portátil para fácil acoplamento com o medidor de energia. Dessa forma, a seguir será apresentado todos os recursos utilizados para a construção da parte física do sistema.

3.3.1 Recursos utilizados

- **NodeMCU**

A placa de desenvolvimento utilizada no projeto foi a NodeMCU, uma plataforma que permite desenvolvimento de projetos de Internet das Coisas com baixo custo, simplicidade e flexibilidade. A placa combina o chip ESP8266, um SoC (*System On Chip*) para a conexão com a rede Wi-Fi. Também possui uma interface USB Serial e um regulador de tensão de 3,3 V.

Para a conexão de periféricos, a placa possui disponível 11 pinos de I/O e conversor analógico-digital, como mostra a Figura 5. Além disso, possui antena embutida e conector Micro-USB para comunicação ao computador e alimentação do sistema.

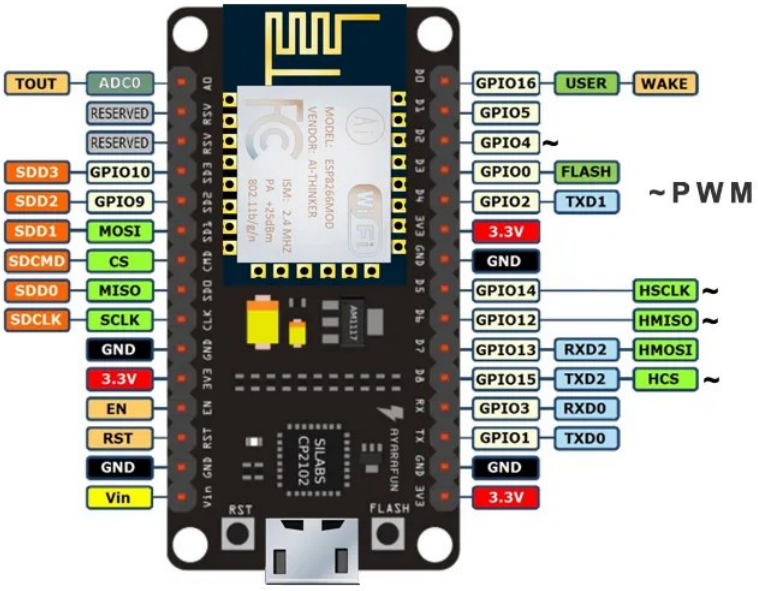


Figura 5 – Esquemático dos pinos da placa NodeMCU [16].

O desenvolvimento do software embarcado do NodeMCU foi utilizando a Arduino IDE (*Integrated Development Environment*). Uma ferramenta de código aberto que permite escrever e fazer *upload* de código dentro de um ambiente de trabalho utilizando a linguagem C++. O sistema é compatível com qualquer placa de software do Arduino. Mas também pode ser utilizada com a NodeMCU através da adição de pacotes específicos para esta finalidade. Dessa forma, foi necessário a instalação do pacote ESP8266 versão 2.4.2 por meio do Gerenciador de Placas do Arduino IDE.

- **ESP8266**

O módulo ESP8266 é um SoC, possui o protocolo de comunicação em rede TCP/IP sem fio, Wi-Fi de 2,4 GHz. A partir dele é possível realizar as funções necessárias para o desenvolvimento de aplicações conectadas ao Wi-Fi. Dessa forma, no projeto apresentado, o módulo é responsável pelo compartilhamento de dados para o banco de dados do Firebase. Para a programação do módulo no Arduino e envio dos dados para o banco de dados é necessário a utilização de algumas bibliotecas.

A biblioteca ESP8266WiFi é responsável para fazer a conexão com a rede Wi-Fi e manter a conexão ativa no sistema. Para enviar e receber os dados para o Firebase é utilizado a biblioteca FirebaseArduino responsável por fornecer a API (*Application Programming Interface*) para a comunicação entre as plataformas. Além disso, os dados enviados para o banco de dados devem estar no formato JSON, por isso é utilizado a biblioteca responsável pela formatação correta dos dados. O código desenvolvido para conectar o hardware ao banco de dados está disponível no Apêndice A.

- **Sensor APDS-9960**

A leitura dos pulsos de LED da constante Kh do medidor de energia é feita utilizando um sensor que permita a detecção de luz e cor no ambiente. Dessa forma, foi escolhido para este projeto o sensor APDS-9960, um módulo eletrônico capaz de detectar toque sem contato, detectar proximidade e fazer medições de luz e cor ambiente. O módulo é amplamente utilizado para a identificação de gestos no desenvolvimento de smartphones, como no Samsung Galaxy S5.

O sensor é composto por quatro diodos separados sensíveis a diferentes direções e filtros bloqueadores de UV e IR, o que permite leituras mais precisas e eficientes. Para conectar o periférico ao sistema embarcado, está disponível uma interface compatível com I2C. Assim, como mostrados na Figura 6, o módulo possui os pinos SLA e SDA para a comunicação I2C, os pinos VL e VCC para alimentação de 3,3 V, o pino INT que permite a utilização de interrupção e o GND.

Figura 6 – Sensor de cor APDS-9960.

A programação do sensor é feita com a utilização de algumas bibliotecas. Para a comunicação é utilizado a biblioteca Wire, já disponível no Arduino IDE, ela permite o protocolo de comunicação I2C para transferência de dados com o dispositivo. Além disso, também é instalado a biblioteca APDS9960 RGB and Gesture Sensor desenvolvida pela SparkFun, ela fornece funções para auxiliar na leitura dos valores de luz e cor ambiente. Dessa forma, o código desenvolvido para a utilização do sensor para a identificação dos pulsos de LED está disponível no Apêndice B.

- **Display OLED**

Para facilitar a utilização do dispositivo e a visualização das informações, será conectado no sistema um display. Foi escolhido o display OLED de 0,96 polegadas de 128x64 pixels. O display permite a exibição de informações com alto nível de nitidez, contraste e excelente resolução com um baixo consumo de energia. Além disso, o display utiliza o microcontrolador SSD1306, que possibilita a conexão com outro microcontrolador via interface I2C. O periférico possui 4 pinos, o SLA e SDA para a comunicação I2C, o VCC para uma alimentação de 3,3 V e o GND, como mostrado na Figura 7.

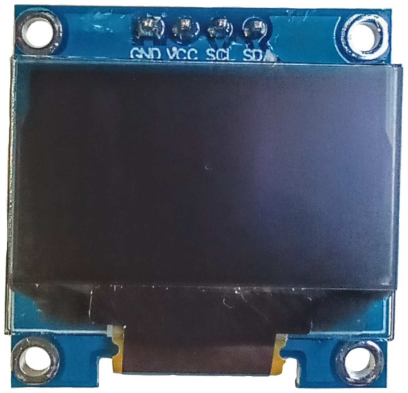


Figura 7 – Display OLED 0,96".

A programação do display é feita com a utilização de algumas bibliotecas. Para a comunicação com o microcontrolador, é utilizado a biblioteca Wire para permite a comunicação com o dispositivo via protocolo I2C. Além disso, também é instalado outras duas bibliotecas, a Adafruit SSD1306 para auxiliar na comunicação do SSD1306 e a Adafruit GFX Library para fornecer um conjunto de recursos gráficos (pontos, linhas, círculos, etc.) para exibição no display. O código desenvolvido para a configuração desse periférico e todas a funções utilizadas para a exibição de informações no display está disponível no Apêndice C.

3.3.2 Montagem

O display OLED e o sensor de cor APS-9960 se comunicam com o NodeMCU através do protocolo I2C. A placa de desenvolvimento possui apenas um barramento I2C disponível.Por isso, os dois principais periféricos do sistema irão compartilhar os mesmos pinos para se conectarem com a placa. O I2C é um barramento de comunicação serial *multiserve*, ou seja, permite a conexão de vários periféricos utilizando apenas dois fios. Na NodeMCU o pino D1 está o SCL, responsável pelo *clock* do barramento, e no pino D2 está o SDA responsável pela transmissão de dados. Dessa forma, como mostrado na Figura 8 foi realizado a conexão dos periféricos.

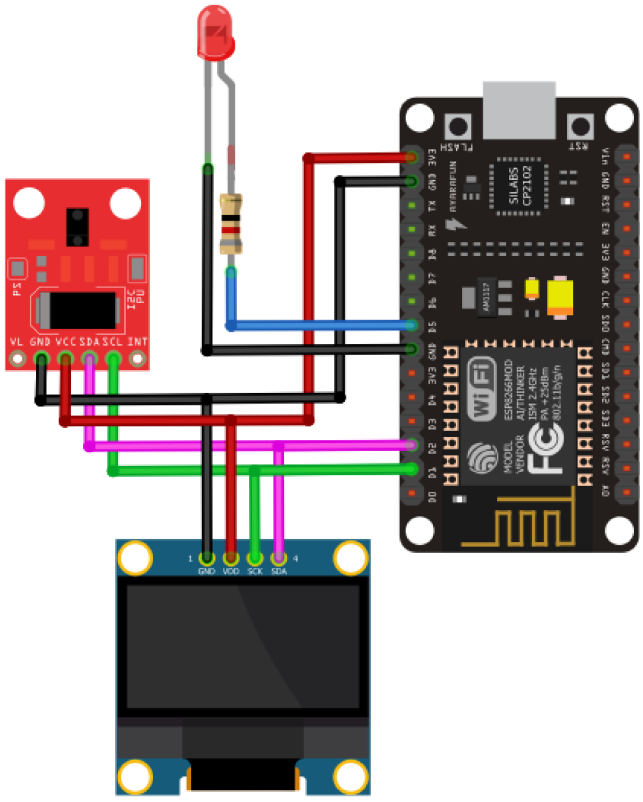
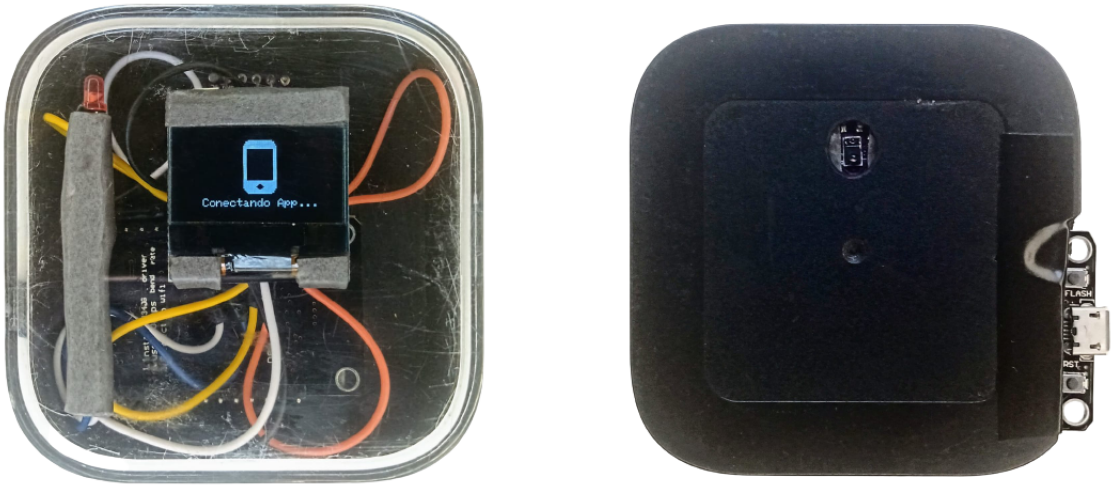


Figura 8 – Esquemático de conexão da placa com os periféricos.

Além dos periféricos descritos, também foi conectado um LED vermelho de alto brilho no pino D5. O LED é responsável por replicar o pulso do LED coletado no medidor, de forma que ainda fique visível para a leitura.

Foi utilizado uma caixa de plástico e acrílico para a montagem da parte física, para que toda a estrutura ficasse compacta e portátil, como mostrado na Figura 9.



(a) (b)

Figura 9 – Montagem do dispositivo físico. (a) Visão parte frontal e (b) Visão parte traseira.

A alimentação do dispositivo é por meio da entrada Micro-USB disponível na placa. A entrada é de 5 V que são convertidos para 3,3 V pelo regulador de tensão interno para ser utilizado na alimentação dos periféricos.

4 FUNCIONAMENTO DO SISTEMA

A interface móvel é responsável por enviar ao dispositivo físico quais ações devem ser realizadas. Por meio dela, o usuário deve ter acesso e controle das funcionalidades do sistema. Ao iniciar o aplicativo, o usuário tem a possibilidade de acessar a funcionalidade teste de calibração ou obter curva de carga, como mostrado na Figura 10.

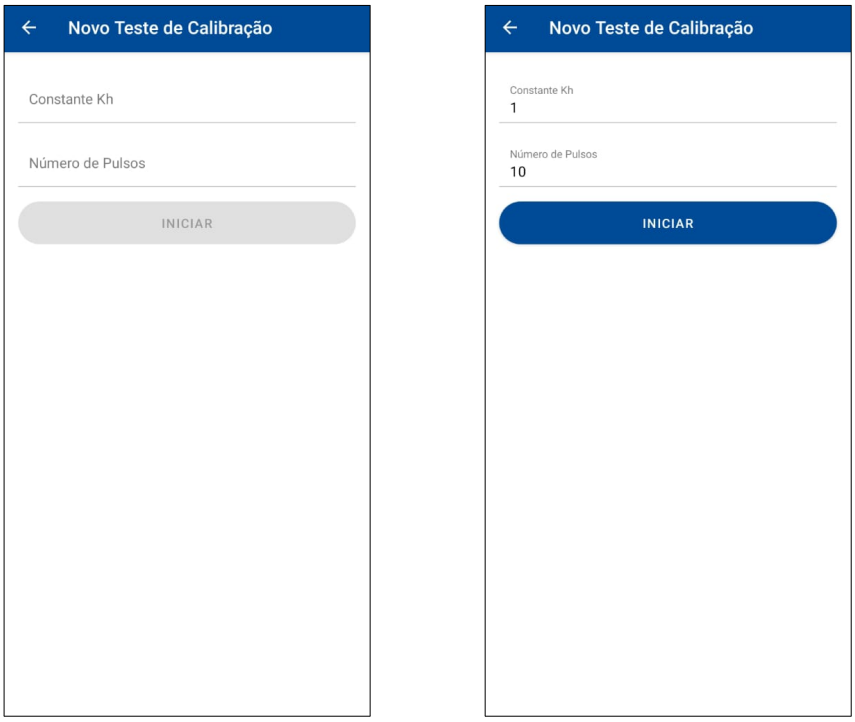
Figura 10 – Menu inicial do aplicativo.

Para iniciar o sistema, o dispositivo físico deve ser ligado e devidamente conectado no medidor. Assim, ocorre todas as inicializações necessárias para que o registrador esteja pronto para receber os comandos do aplicativo. O código principal do dispositivo físico responsável pelo controle das funcionalidades do sistema, está disponível no Apêndice D

Nos tópicos a seguir, serão apresentados os detalhes do funcionamento de cada uma das funcionalidades: o Teste de Calibração e obtenção da Curva de Carga.

4.1 TESTE DE CALIBRAÇÃO

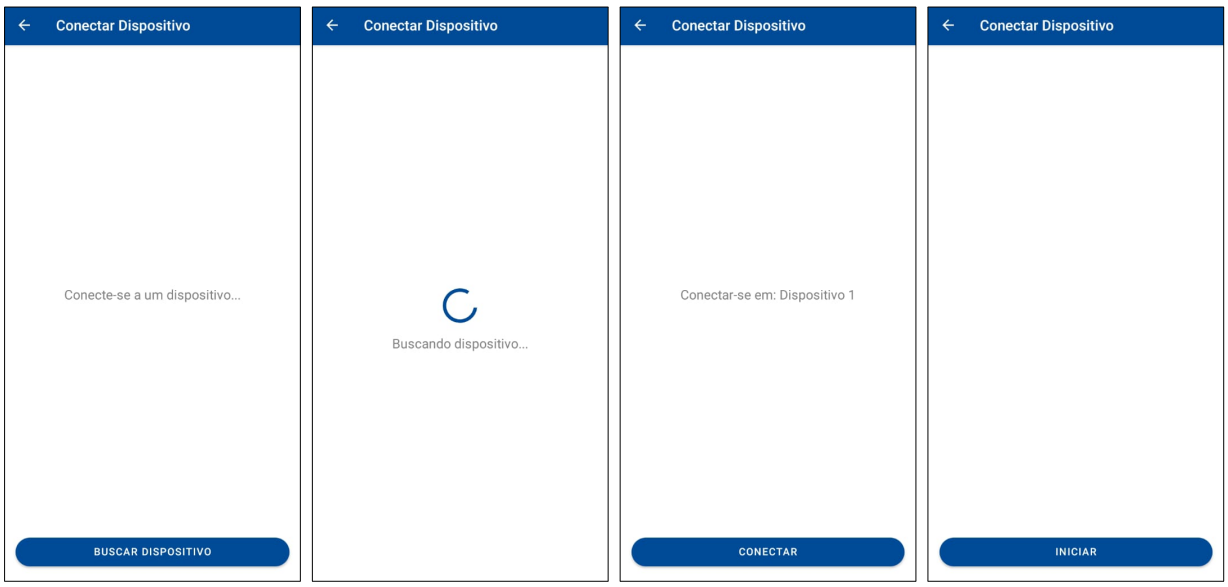
Para iniciar um Teste de Calibração é necessário a entrada das informações básicas para o teste. O primeiro campo de entrada é a constante Kh, uma informação específica do medidor que será realizado o teste. O campo permite a entrada de valores inteiros e decimais. O segundo campo de entrada é a quantidade de pulsos que se deseja ler para a realização do cálculo. O botão de iniciar fica desabilitado até a entrada de valores validos. A tela descrita é representada pela Figura 11.



(a) (b)

Figura 11 – Tela para iniciar teste de precisão. (a) Campos vazios e (b) Campos preenchidos.

Ao pressionar o botão de iniciar, os dados inseridos são enviados para o banco de dados. O usuário navega para a tela seguinte para realizar a conexão com o hardware. O processo de conexão entre duas plataformas realizado pelo aplicativo é mostrado na Figura 12.

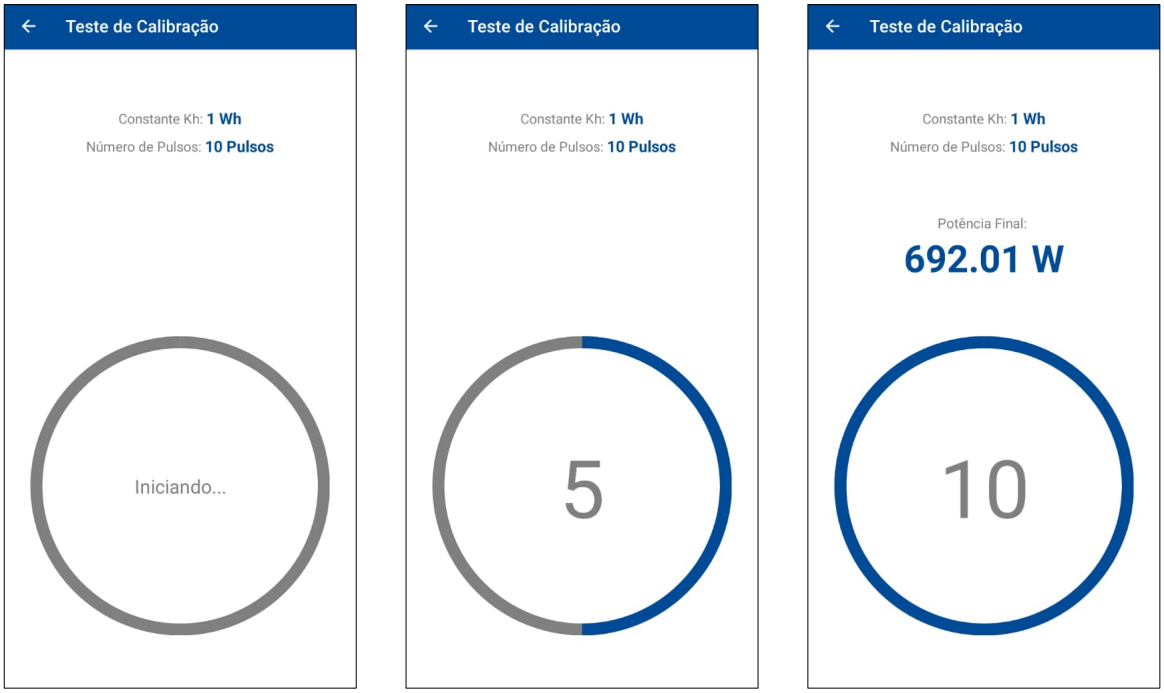


(a) (b) (c) (d)

Figura 12 – Tela conexão com dispositivo. (a) Buscar, (b) Buscando, (c) Conectar e (d) Iniciar.

Com todo o sistema devidamente conectado, o usuário navega para a tela em que pode acompanhar a realização do Teste de Calibração. Assim, o primeiro estágio do teste é a inicialização do próprio sistema, neste momento o registrador descarta a leitura dos três primeiros pulsos lidos. O descarte dos três primeiros pulsos é necessário para evitar a leitura de qualquer pulso incompleto ou erros iniciais.

Após a inicialização, o dispositivo começa a registrar os valores de cada pulso até atingir a quantidade de pulsos informada. Finalizado a contagem, é exibido o valor da potência final calculada. A tela exibida pela interface durante a realização do teste é apresentada na Figura 13.



(a) (b) (c)

Figura 13 – Tela teste de calibração. (a) Iniciando, (b) Obtendo pulsos e (c) Finalizado.

O cálculo responsável pela obtenção do valor da potência coletada do medidor é apresentado pela Equação 4.1:

$$P_{med} = \frac{N_p * Kh * 3600000}{T_T} \tag{4.1}$$

Onde:

Kh é a constante de verificação Kh, expressa em Wh/pulso;

N_p é o número de pulsos;

T_T é o tempo total para registrar número de pulsos inteiros, expresso milisegundos;e

P_{med} é a potência medida pelo medidor, expressa em W.

O cálculo é realizado pelo dispositivo físico, sendo responsável por enviar para o banco de dados somente o valor final da potência. Todo código da função executada pelo registrador, é apresentado com detalhes no Apêndice E

4.2 CURVA DE CARGA

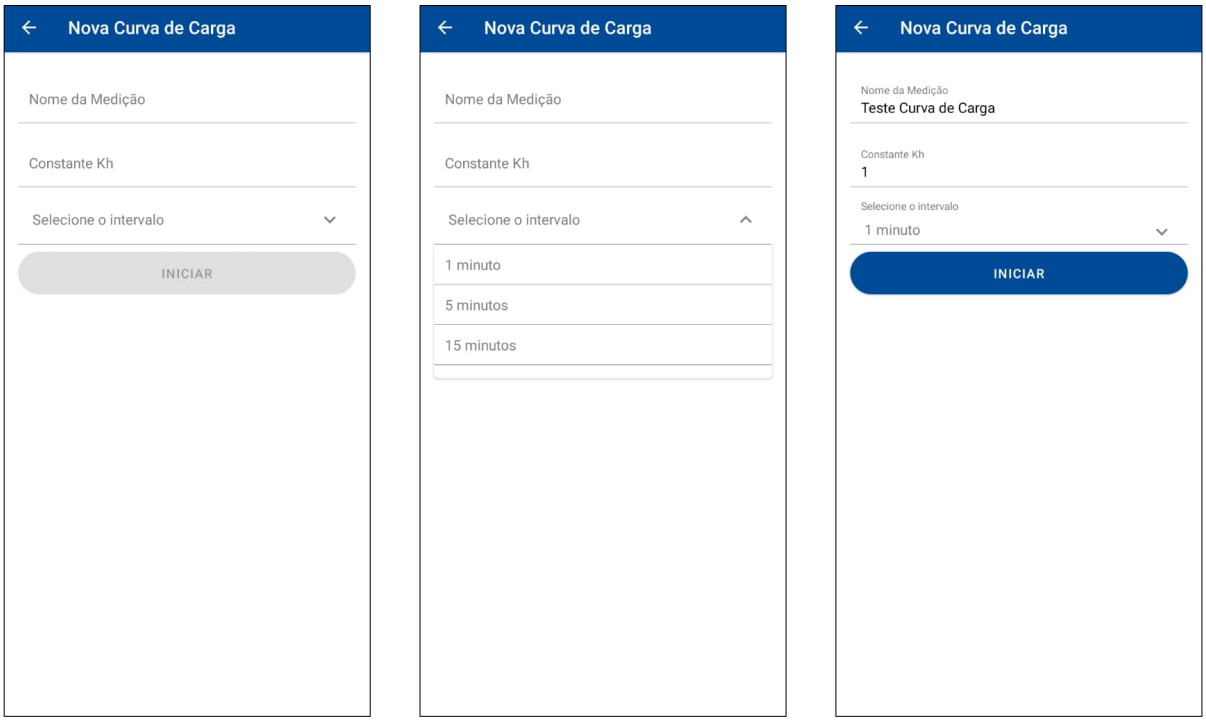
A funcionalidade de obter a curva de carga do medidor também faz o armazenamento das medições já realizadas. Assim, ao navegar do Menu Inicial para a Curva de Carga é possível visualizar a lista de todas as curvas de cargas já obtidas ordenadas de acordo com a data de realização. Ao selecionar uma curva de carga, o usuário navega para a tela em que visualiza a curva com todos os dados da medição já realizada. Além disso, na parte inferior da tela possui o botão em que permite iniciar novas medições de curva de carga. A tela descrita é apresentada na Figura 14.

Figura 14 – Tela inicial Curva de Carga.

Para iniciar uma nova coleta de dados em tempo real para a construção da curva de carga, o usuário navega para a tela de entrada dos dados necessários para iniciar a medição. No primeiro campo de entrada, deve ser informado o nome da medição a ser

realizada. O nome é utilizado como uma forma de identificação entre outras medições já realizadas para facilitar o armazenamento. No campo seguinte, deve ser inserido o valor da constante Kh do medidor em que será realizado a coleta dos dados.

Por fim, deve ser selecionado através de um menu de opções o intervalo de tempo em que os valores de potência serão enviados para o banco de dados. É possível escolher entre 5 valores de intervalo: 1 minuto, 5 minutos, 15 minutos, 30 minutos e 1 hora. Assim, ao preencher todos os campos necessários, o botão de iniciar é habilitado para iniciar a medição. A tela descrita é apresentada na Figura 15.



(a) (b) (c)

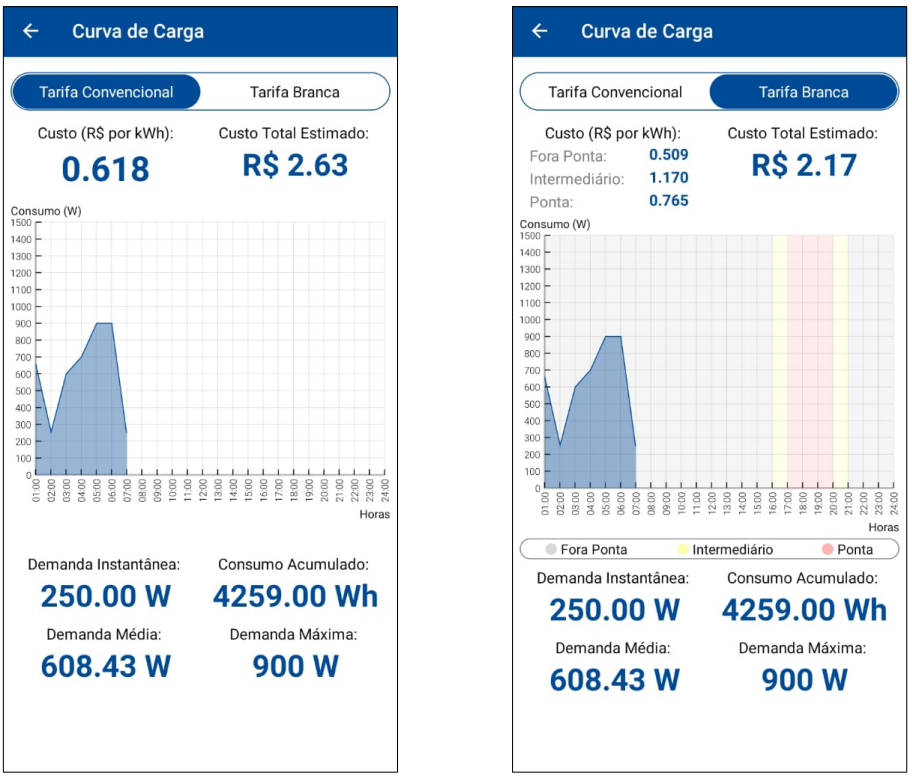
Figura 15 – Tela nova curva. (a) Campos vazios, (b) Intervalos e (c) Campos preenchidos.

Antes de iniciar a coleta dos dados, a interface deve se conectar com o dispositivo físico como mostrado nas telas da Figura 12.

Após a conexão do hardware com a interface, o usuário pode navegar para a tela de visualização da curva de carga. Nesta tela, o usuário acompanha a obtenção da curva de carga em tempo real. Além da curva de carga, a tela também exibe informações como a demanda média, demanda instantânea, demanda máxima, consumo acumulado e custo total estimado.

O valor do custo total estimado pode ser visualizado considerando a Tarifa Branca e a Tarifa Convencional. Essa funcionalidade podendo ser alternada a qualquer momento durante a medição, o que permite comparações entre os tipos de tarifa em tempo real. A

tela descrita é apresentada na Figura 16.



(a) (b)

Figura 16 – Tela curva de carga. (a) Tarifa Convencional e (b) Tarifa Branca.

A cada intervalo de tempo o registrador conectado no medidor envia o valor de potência para o banco de dados. Dessa forma, o Firebase comunica para a interface a adição de um novo valor. A interface é responsável por acrescentar o valor da potência recebido e atualizar os outros valores exibidos de acordo com o novo valor. Todo código da função executada pelo registrador para construir a curva de carga, é apresentado com detalhes no Apêndice F

5 RESULTADOS E DISCUSSÕES

Para a validação das funcionalidades desenvolvidas, foi realizado testes em três diferentes tipos de medidores de energia eletrônicos, apresentados na Figura 17. Os medidores escolhidos estão disponíveis no mercado e são utilizados por concessionárias de energia no Brasil.

Os testes foram realizados no Laboratório de Distribuição de Energia Elétrica (LADEE) utilizando um equipamento para aplicar uma corrente controlada no medidor. O equipamento é uma fonte de potência programável CMC 256 Plus, fabricada pela Omicron Electronics Corp, utilizado para aplicações que necessitam de alta precisão.

(a) (b) (c)

Figura 17 – Medidores utilizados nos testes. (a) Medidor 1, (b) Medidor 2 e (c) Medidor 3.

5.1 TESTE DE CALIBRAÇÃO

A verificação da funcionalidade de Teste de Calibração de medidores de energia foi realizada utilizando os três medidores citados anteriormente. Foi aplicado uma potência fixa de 1200 W pelo aparelho CMC 256 Plus em cada medidor. Todos os testes foram realizados utilizando 10 pulsos de precisão. Após realizar o teste em cada medidor, foi construído a Tabela 1 com os valores obtidos e os respectivos erros em relação à potência de 1200 W aplicada no medidor.

Tabela 1 – Testes utilizando o registrador IoT

Tipo Medidor	Potência Registrada (W)	Erro (%)
Medidor 1	1195,85	0,35
Medidor 2	1197,05	0,25
Medidor 3	1198,68	0,11

Para fins de comparação, também foi realizado testes em um registrador de pulsos desenvolvido pelo LADEE, apresentado na Figura 18. Ele possui a mesma funcionalidade para a realização de testes de calibração, sua construção foi utilizando um leitor de pulsos fabricado pela Nansen S.A e um microcontrolador Arduino para o tratamento das informações. Além disso, também é importante destacar, que o aparelho foi certificado por um laboratório de metrologia.

Figura 18 – Registrados de pulsos desenvolvido no LADEE.

Para a realização dos testes, foi aplicado a mesma potência fixa de 1200 W nos três medidores e configurado o contador para realizar a leitura de 10 pulsos. Dessa forma, foi construído a Tabela 2 com os valores obtidos e os respectivos erros.

Tabela 2 – Testes utilizando o registrador do LADEE

Tipo Medidor	Potência Registrada (W)	Erro (%)
Medidor 1	1196,59	0,28
Medidor 2	1197,03	0,25
Medidor 3	1199,02	0,08

Ao analisar os resultados obtidos nos dois testes é possível observar que registrador IoT e o registrador do LADEE sempre apresentaram erros menores que 0,5 %. Para cada medidor testado os erros analisados seguiram bem próximos nos dois dispositivos, isso indica que o erro encontrado é relativo à própria precisão do medidor.

Além disso, é importante notar que o registrador LADEE apresentou erros menores que o IoT. Isso pode ter ocorrido devido o sensor utilizado para a detecção de pulsos ser de maior qualidade e ter oferecido maior precisão.

5.2 CURVA DE CARGA

Para validar a funcionalidade do sistema de obter a curva de carga foi utilizado a fonte de potência programável para simular uma potência variável durante o tempo. Foi

aplicado uma potência de período de 15 minutos com uma variação de potência a cada 3 minutos. A curva de carga simulada é apresentada na Figura 19 com a indicação dos valores de potência.

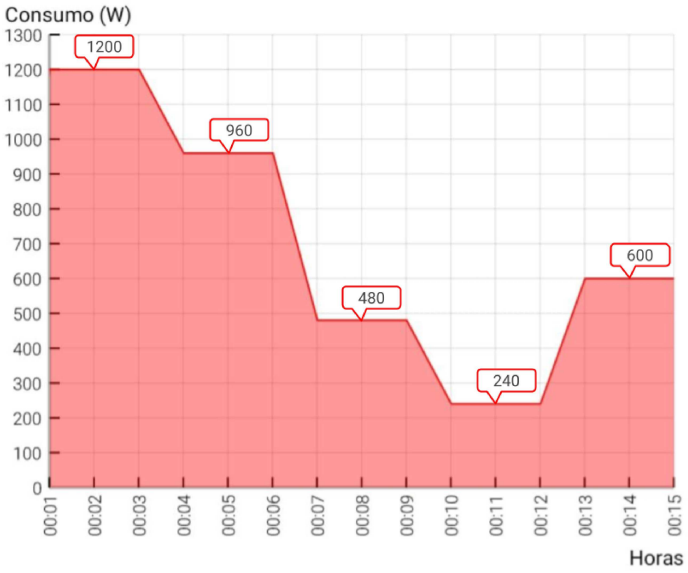


Figura 19 – Curva Aplicada.

O registrador de pulsos foi configurado para enviar os valores de potência em intervalos de 1 minuto. Dessa forma, foi realizado a medição para os três medidores. Nas Figuras 20, 21 e 22 é apresentado as curvas de carga de cada medidor. A linha vermelha indica os valores inseridos no medidor pela fonte de potência e a linha azul a curva de carga lida pelo registrador IoT.

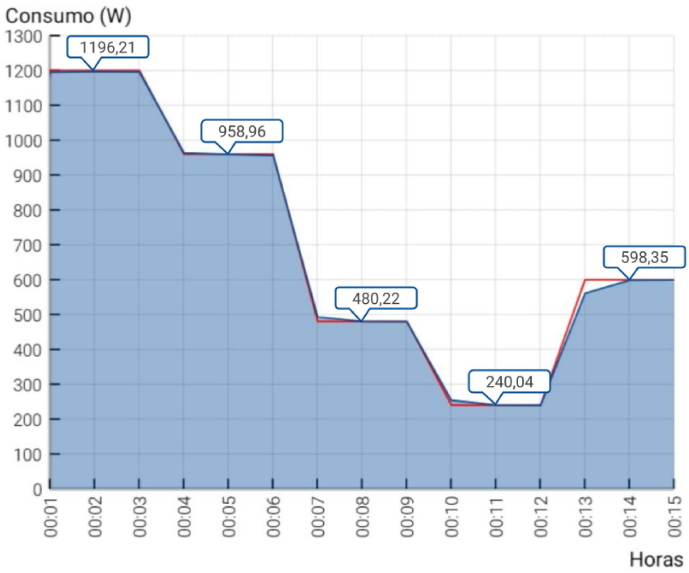


Figura 20 – Medidor 1.

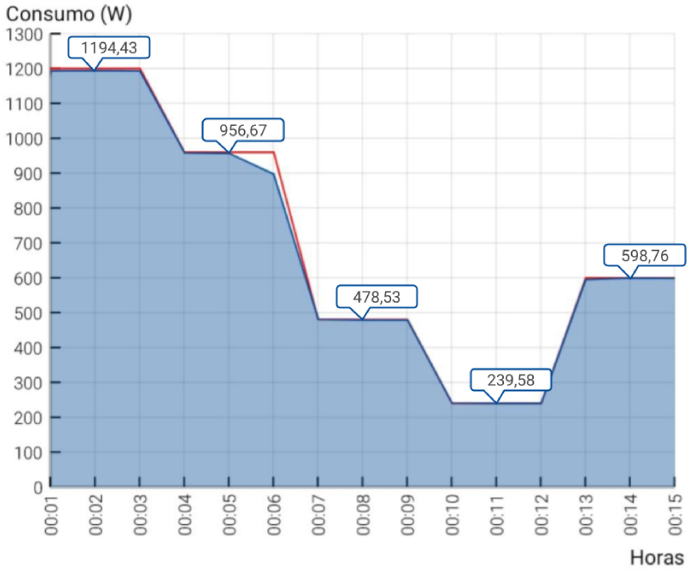


Figura 21 – Medidor 2.

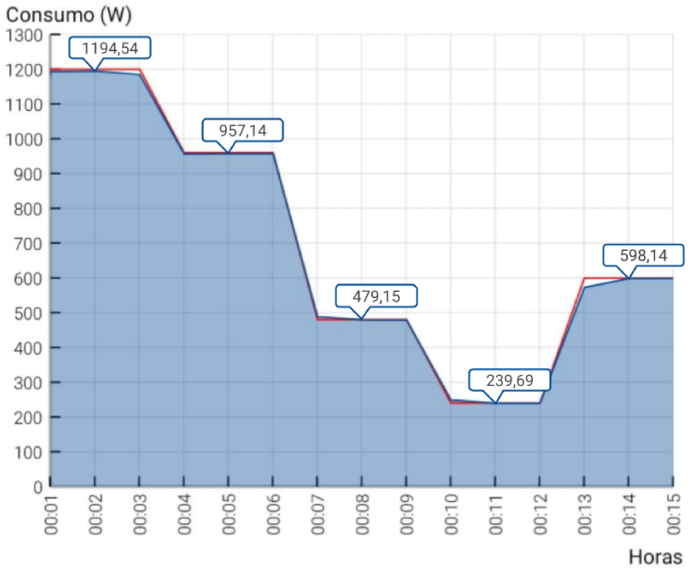


Figura 22 – Medidor 3.

Ao analisar os resultados obtidos, é observado nas três medições erros maiores nos valores anteriores ou posteriores a uma mudança de potência. Isso ocorreu devido uma dissincronia no início da aplicação da potência no medidor e no início da medição no registrador. Como o registrador foi configurado para realizar o envio do valor da potência em intervalos de 1 minuto, em alguns casos a dissincronia no início da medição causou a leitura de diferentes valores de potência em um mesmo intervalo.

Cada variação de potência teve uma duração de 3 minutos, ou seja, três intervalos de 1 minuto. Dessa forma, melhor valor para analisar o erro da potência medida pelo registrador é o valor intermediário, pois mesmo em casos de dissincronia a leitura não é

afetada pelos valores anteriores ou posteriores.

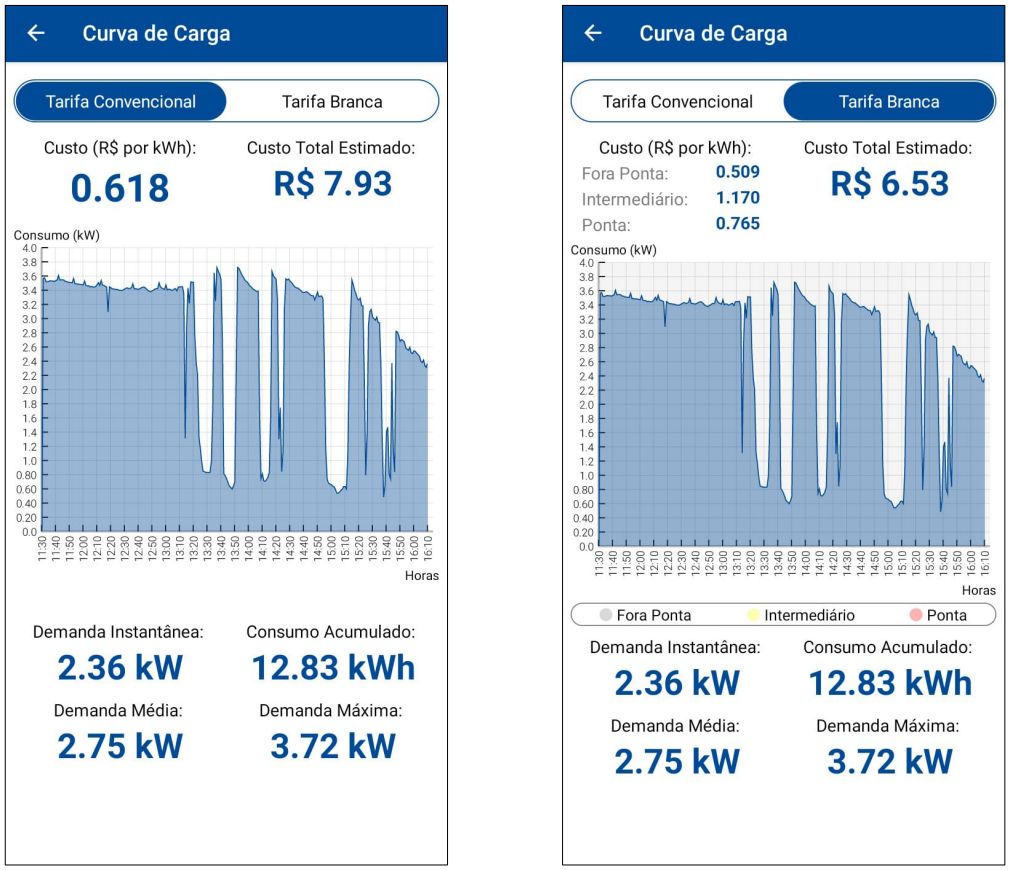
Considerando assim, os valores da curva de carga medida apresentaram valores bem próximos do esperando, apresentando erro máximo de 0,46 %. De modo geral, a medição da curva de carga obteve um resultado próximo do esperado. Mesmo em condições de dissincronia a curva manteve resultados aceitáveis.

5.2.1 Curva de carga de um sistema de geração

A funcionalidade de obter curva de carga também foi testada em um medidor instalado em um sistema de geração de energia fotovoltaica para obter uma curva real. O sistema está instalado no Laboratório de Distribuição de Energia Elétrica da Universidade Federal de Uberlândia, apresentado na Figura 23. O sistema possui um inversor inteligente, o Growatt MIN 2500-6000TL, em que fornece ferramentas para o monitoramento da geração de energia do sistema. Além disso, também possui um medidor eletrônico, igual ao Medidor 3 da Figura 17, em que é responsável por fazer a quantificação da energia gerada.

Figura 23 – Montagem do sistema de geração no laboratório.

O registrador de pulsos desenvolvido foi conectado no medidor para a obtenção da curva de carga. Foi definido um intervalo de 1 minuto para receber o valor de potência. O período de medição foi de 11:30 às 16:10, em que foi possível obter a curva apresentada na Figura 24.



(a) (b)

Figura 24 – Curva de carga do sistema de geração. (a) Tarifa Convencional e (b) Tarifa Branca.

Para fins de comparação, também foi coletado a memória de massa do inversor inteligente no mesmo período de medição. Ele armazena a potência gerada em intervalos de 10 minutos. Assim, os dados são apresentados na Figura 25.

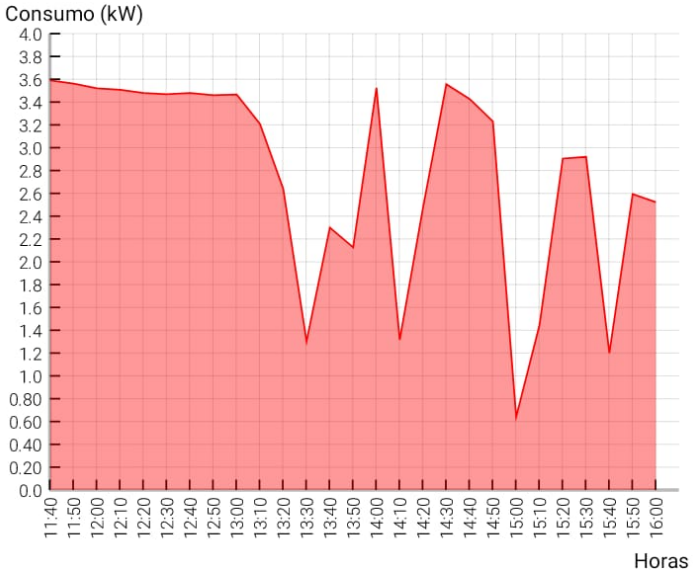


Figura 25 – Curva obtida pelo inversor inteligente.

Para realizar uma comparação mais adequada entre as curvas obtidas pelos dois dispositivos, os dados coletados pelo registrador de pulsos foram formatados para serem apresentados também em intervalos de 10 minutos. Para isso, os dados foram retirados do banco de dados e realizado a média dos valores lidos no mesmo período de tempo da medição realizada pelo inversor inteligente. Dessa forma, foi construído outra curva de carga para comparar os valores das duas medições, com apresentado na Figura 26. A linha vermelha representa os dados coletados pelo inversor e a curva em azul representa os dados coletados pelo registrador de pulsos.

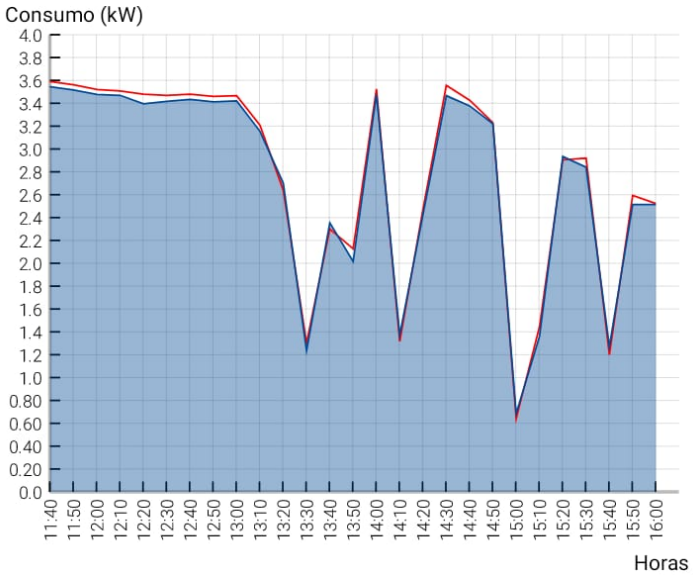


Figura 26 – Curva de carga do registrador em intervalos de 10 min.

Ao analisar as duas curvas é possível comprovar que a obtenção de curva de carga pelo registrador de pulsos desenvolvido apresentou resultados satisfatórios. A maioria dos valores de potência obtidos pelos dois dispositivos foram bem próximos e dentro dos limites aceitáveis. Além disso, o registrador de pulsos apresentou um bom funcionamento durante todo o período da medição, todos os dados coletados foram enviados corretamente para o banco de dados e nenhuma informação foi perdida.

6 CONCLUSÕES

Ao iniciar a pesquisa desse trabalho, foi constatado a importância da utilização da constante Kh na calibração de medidores para garantir a tarifação adequada para os consumidores. A constante Kh está presente em todos os tipos de medidores de energia elétrica e realizar sua leitura é possível obter a quantidade de energia que o medidor está registrando no momento. Além disso, também foi destacado a relevância do conhecimento da curva de carga e a necessidade de ferramentas que auxiliem na obtenção desses dados. A curva de carga oferece informações fundamentais para o melhor gerenciamento do consumo, aplicação de medidas de eficiência energética e escolha da melhor modalidade de tarifa.

Diante disto, esse trabalho teve como objetivo o desenvolvimento de um dispositivo para auxiliar na calibração e obtenção a curva de carga por meio da constante Kh em medidores de energia eletrônicos. Constata-se que o objetivo foi atendido, pois ao longo do trabalho foi apresentado o desenvolvimento de um sistema em que é possível coletar pulsos para realizar o teste de calibração do medidor e também construir a curva de carga a partir destes dados.

No sistema desenvolvido foi aplicado conceitos de Internet das Coisas para transmitir os dados coletados no registrador via internet e em tempo real. O registrador foi construído utilizando um sensor de luz e cor ambiente APDS-9960 responsável por fazer a detecção dos pulsos de LED do medidor eletrônico. Além disso, para se conectar com o sensor, foi utilizado a placa de desenvolvimento eletrônica NodeMCU em que possui acoplada o módulo Wi-Fi para o compartilhamento dos dados com o banco de dados no Firebase.

Para controlar o registrador e visualizar os dados coletados foi desenvolvido um aplicativo móvel. O aplicativo se conecta com o dispositivo do sistema por meio do banco de dados hospedado na nuvem. O aplicativo foi desenvolvido utilizando o *framework* React Native e a plataforma Expo em que fornecem ferramentas para auxiliarem no desenvolvimento. O aplicativo tem potencial para funcionar em sistema Android e iOS, mas todos os testes foram realizados no Android.

Nos testes realizados em laboratório foi possível obter resultados satisfatórios na utilização do sistema. Quando comparado com outros dispositivos com funcionalidades parecidas os dados coletados foram próximos e dentro dos limites aceitáveis. Dessa forma, foi possível comprovar a eficiência do dispositivo desenvolvido para a realização de testes de calibração de medidores de energia. Como também, mostrar que por meio da constante Kh presente dos medidores é possível realizar a coleta de informações importantes para os consumidores como a curva de carga

Após a realização dos testes do sistema proposto foi notado algumas limitações. A

primeira delas, foi a necessidade do uso de uma alimentação externa para o dispositivo físico. A utilização de uma alimentação própria por meio de uma bateria facilitaria bastante a utilização do sistema em campo. Além disso, também é importante notar que todos os testes foram realizados em ambiente de laboratório. Em consumidores reais é observado que alguns medidores são instalados dentro de caixas de proteção, o que dificultaria o acoplamento do registrador no medidor para a utilização do sistema apresentado.

REFERÊNCIAS

[1] ANEEL, *Resolução Normativa nº 414, de 9 de setembro de 2010*, Estabelece as Condições Gerais de Fornecimento de Energia Elétrica de forma atualizada e consolidada. Acesso em 01 de Março de 2022 em: `https://www.aneel.gov.br/documents/656877/14486448/bren2010414.pdf`

[2] ANEEL, *Módulo 7 - Estrutura Tarifária das Concessionárias de Distribuição de Energia Elétrica*, Procedimentos de Regulação Tarifária - PRORET, 2021.

[3] G. Miyasaka, *Análise do desempenho de medidores de energia elétrica ativa em condições distorcidas e desequilibradas*, Uberlândia: Dissertação de Mestrado, Universidade Federal de Uberlândia, 2020.

[4] ANEEL, *ANEEL regulamenta medidores eletrônicos.* Acesso em 20 de Fevereiro de 2022 em: `https://www.aneel.gov.br`.

[5] ANEEL, *Tarifa Branca.* Acesso em 20 de Fevereiro de 2022 em: `https://www.aneel.gov.br/tarifa-branca`.

[6] C. W. Gellings, The Smart Grid: Enabling Energy Efficiency and Demand Response, GA, Estados Unidos: Editora Fairmont Press Inc, 2009.

[7] D. R. Leite, *Medidores eletrônicos: uma análise da viabilidade econômica no contexto das redes inteligentes*, Brasília: Dissertação de Mestrado, Universidade Federal de Brasília, 2013.

[8] O Setor Elétrico, *Neoenergia leva medidores inteligentes ao interior de São Paulo* Acesso em 20 de Fevereiro de 2022 em: `https://www.osetoreletrico.com.br/neoenergia-leva-medidores-inteligentes-ao-interior-de-sao-paulo/`.

[9] O. Siddiqui, *The Green Grid: Energy Savings and Carbon Emissions Reductions Enabled by a Smart Grid*, EPRI Technical Update Report 1016905, 2008.

[10] O Setor Elétrico, *Conceitos básicos de eletrotécnica aplicada* Acesso em 01 de Março de 2022 em: `https://www.osetoreletrico.com.br/conceitos-basicos-de-eletrotecnica-aplicada-a-curva-de-carga/`.

[11] A. A. Francisquini, *Estimação de curvas de carga em pontos de consumo e em transformadores de distribuição*, Ilha Solteira: Dissertação de Mestrado, Universidade Estadual Paulista "Júlio De Mesquita Filho", 2006.

[12] P. V. S. Queiroz, *Mensuração do consumo de energia elétrica: algoritmo para detecção de potenciais usuários da termoacumulação como alternativa para deslocamento de carga*, Rio de Janeiroa: Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2011.

[13] B. P. Santos, L. A. M. Silva, *Internet das coisas: da teoriaa prática*, Belo Horizonte: Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2016.

[14] J. Buckley, *The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems* New York: Auerbach Publications, 2006.

[15] PathPartner Technology, *Sensors – The most vital element of the Internet of Things* Acesso em 01 de Março de 2022 em: https://www.pathpartnertech.de/sensors-the-most-vital-element-of-the-internet-of-things/.

[16] Teach Me Microcontrollers, *NodeMCU Pinout Reference* Acesso em 20 de Janeiro de 2022 em: https://www.pathpartnertech.de/sensors-the-most-vital-element-of-the-internet-of-things/.

APÊNDICE A – Código do ESP8266 para conectar ao banco de dados

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <ArduinoJson.h>

#define FIREBASE_HOST "medidor-iot-2-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "S2kD0fPrCvIOjvdC1kqRMM7WRf6W86VD4IH90qeY"
#define WIFI_SSID "Sueli"
#define WIFI_PASSWORD "JARVIS051170"

void iniciar_wifi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Conectando...");
  exibir_display_wifi_conectando();
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Conectando: ");
  exibir_display_wifi_conectado(WIFI_SSID);
  Serial.println(WiFi.localIP());
}

void reconectar_wifi(void) {
  if (WiFi.status() != WL_CONNECTED)
  {
    WiFi.reconnect();
  }
}

void iniciar_firebase() {
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.setString("/info/status", "desconectado");
}

String conectar_app() {
  String status_app = Firebase.getString("/info/status");
  int app_key = 0;
  String selectedMedicaoKey = "";
  String modo = "inicio";

  exibir_display_conectando_app();

  while (status_app != "conectado") {
    app_key =  Firebase.getInt("/info/keyApp");
    Firebase.setString("/info/keyDispositivo", String(app_key) + "OK");
    status_app = Firebase.getString("/info/status");
    delay(100);
  }
  modo = Firebase.getString("/info/modo");
  return modo;
}
```

APÊNDICE B – Código do sensor de cor

```
#include <SparkFun_APDS9960.h>
#include <Wire.h>
#include <SPI.h>

#define COLOR_LAG          50
#define COLOR_WIN          10
#define COLOR_THRESHOLD    600
#define PULSE_DURATION     10
#define led_pin            D5

#define COLOR_LAST_IDX(x) (x == 0 ? COLOR_LAG - 1 : x - 1)
#define COLOR_IDX(x,n)      ((x-n) < 0 ? COLOR_LAG + (x-n) : x - n)

uint16_t data;
uint16_t data_amb;
int8_t pulse;
int8_t last_pulse;
uint16_t lag;
int32_t tmf;
int32_t psi;
uint32_t pulse_time;
int32_t color_avg[COLOR_LAG];
int32_t color_data[COLOR_LAG];
int color_sense_total_pulses;
SparkFun_APDS9960 apds = SparkFun_APDS9960();

int color_sense_get_pulses(){
  return color_sense_total_pulses;
}

void color_sensor_init(){
  memset(color_avg,0,sizeof(color_avg));
  memset(color_data,0,sizeof(color_data));

  lag = 0;
  last_pulse = pulse_time = 0;
  pulse = 0;
  color_sense_total_pulses = 0; // TODO: reload from persistence

  apds.init();
  apds.enableLightSensor(false);

  pinMode(led_pin, OUTPUT);

  delay(500); // calibra  o
}

void color_sensor_tick(){
  uint16_t n;
  int32_t tm;

  apds.readAmbientLight(data_amb);
  apds.readRedLight(data);

  color_data[lag] = data;
```

```
  tm = 0;
  for(n = 0 ; n < COLOR_WIN ; n++){
    tm += color_data[COLOR_IDX(lag,n)];
  }


  tm = tm/COLOR_WIN;
  tmf = color_avg[lag] = tm;

  // y[n] = x[n-1]^2 - x[n]*x[n-2]
  int32_t f0 = (color_avg[COLOR_IDX(lag,0)] + color_avg[COLOR_IDX(lag,1)] + color_avg[
      COLOR_IDX(lag,2)])/3;
  int32_t f1 = (color_avg[COLOR_IDX(lag,3)] + color_avg[COLOR_IDX(lag,4)] + color_avg[
      COLOR_IDX(lag,5)])/3;
  int32_t f2 = (color_avg[COLOR_IDX(lag,6)] + color_avg[COLOR_IDX(lag,7)] + color_avg[
      COLOR_IDX(lag,8)])/3;

  psi = f1*f1 - f0*f2;

  if(psi > COLOR_THRESHOLD){
    pulse_time = 0;
    pulse = 1;
    digitalWrite(led_pin, HIGH);
  }

  else{
    digitalWrite(led_pin, LOW);
    if(++pulse_time > PULSE_DURATION){
      pulse = 0;
      pulse_time = 0;
    }
  }

  if(pulse == 0 && last_pulse == 1){
    color_sense_total_pulses++;
  }

  last_pulse = pulse;

  if(++lag >= COLOR_LAG){
    lag = 0;
  }
}
```

APÊNDICE C – Código do display OLED

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define NUMFLAKES 10
#define LOGO_HEIGHT 16
#define LOGO_WIDTH  16
#define SCREEN_WIDTH  128
#define SCREEN_HEIGHT 64
#define OLED_RESET      LED_BUILTIN
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

const unsigned char PROGMEM ladee_logo [] = {0x00 ... 0x00}; // imagem no formato
    hexadecimal
const unsigned char PROGMEM smartphone_48x48[] = {0x00 ... 0x00}; // imagem no formato
    hexadecimal
const unsigned char PROGMEM teste_32x32[] = {0x00 ... 0x00}; // imagem no formato
    hexadecimal
const unsigned char PROGMEM curva_32x32[] = {0x00 ... 0x00}; // imagem no formato
    hexadecimal
const unsigned char PROGMEM wifi_48x48[] = {0x00 ... 0x00}; // imagem no formato
    hexadecimal

void iniciar_display(void){
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)){
    Serial.println(F("SSD1306␣allocation␣failed"));
    for(;;);
  }
}

void exibir_display_logo_inicial(){
  display.clearDisplay();
  display.drawBitmap(32, 0, ladee_logo, 64, 64, 1);
  display.display();
  delay(2000);
}

void exibir_display_teste_precisao(){
  display.clearDisplay();
  display.drawBitmap(17, 16, teste_32x32, 32, 32, 1);
  display.setCursor(55, 22);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("TESTE␣DE");
  display.setCursor(55, 34);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("PRECISAO");
  display.display();
}

void exibir_display_curva_carga(){
  display.clearDisplay();
  display.drawBitmap(17, 16, curva_32x32, 32, 32, 1);
```

```
  display.setCursor(55, 22);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("CURVA");
  display.setCursor(55, 34);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("DE CARGA");
  display.display();
  delay(500);
}

void exibir_display_wifi_conectado(String nome_wifi){
  display.clearDisplay();
  display.drawBitmap(40, 0, wifi_48x48, 48, 48, 1);
  display.setCursor(28, 48);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Conectado em:");
  display.setCursor(40, 56);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println(nome_wifi);
  display.display();
  delay(2000);
}

void exibir_display_wifi_conectando(){
  display.clearDisplay();
  display.drawBitmap(40, 0, wifi_48x48, 48, 48, 1);
  display.setCursor(28, 52);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Conectando...");
  display.display();
}

void exibir_display_conectando_app(){
  display.clearDisplay();
  display.drawBitmap(52, 0, smartphone_48x48, 29, 48, 1);
  display.setCursor(16, 52);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Conectando App...");
  display.display();
}

void exibir_display_dados_curva_carga(String nome_curva, float constante, String
    intervalo){
  display.clearDisplay();
  display.setCursor(0, 0);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.print("Curva:");
  display.println(nome_curva);
  display.print("Constante:");
  display.print(constante);
```

```
  display.println("Wh");
  display.println("Intervalo: ");//inserir periodo
  display.println(intervalo);
  display.display();
  delay(2000);
}

void exibir_display_dados_teste( float constante, int numeroPulsos){
  display.clearDisplay();
  display.setCursor(0, 0);
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.println("Teste de Precisao");
  display.print("Constante:");
  display.print(constante);
  display.println("Wh");
  display.print("Numero Pulsos: ");
  display.println(numeroPulsos);
  display.display();
  delay(2000);
}
```

APÊNDICE D – Código principal do dipositivo

```
#include <SparkFun_APDS9960.h>
#include <ESP8266WiFi.h>
#include <Wire.h>

String modo = "inicio";

void setup() {
  Wire.begin();
  Serial.begin(9600);
  iniciar_display();
  exibir_display_logo_inicial();
  color_sensor_init();
  iniciar_wifi();
  iniciar_firebase();
}

void loop() {
  if (modo == "teste")
    executar_teste_precisao();
  else if (modo == "medicao")
    obter_curva_carga();
  else if (modo == "inicio")
    modo = conectar_app();
}
```

APÊNDICE E – Código para executar teste de precisão

```
String executar_teste_precisao() {
  String selectedMedicaoKey;
  float constanteKh;
  String nomeCurva;
  String valorIntervalo;
  String statusMedicao;
  float intervalo;
  int initTime;
  int lastPulse;
  int currentTime;
  boolean isFirstPulse = true;
  int lastPulseTime;
  int numeroPulsos = 0;
  int totalTime = 1;
  float potenciaTotal;
  int pulseDurationTime = 0;
  int numeroPulsosTeste = 0;
  String statusTeste = "calibrando";

  exibir_display_teste_precisao();
  constanteKh =  Firebase.getFloat("/testePrecisao/constanteKh");
  numeroPulsosTeste = Firebase.getInt("/testePrecisao/numeroPulsos");
  statusTeste = Firebase.getInt("/testePrecisao/status");

  Firebase.setString("/testePrecisao/status", "calibrando");
  Firebase.setInt("/testePrecisao/pulsoAtual", 0);
  Firebase.setInt("testePrecisao/potenciaFinal", 0);

  exibir_display_dados_teste(constanteKh, numeroPulsosTeste);

  while (statusTeste != "finalizado")
  {
    color_sensor_tick();
    Serial.print("color_sense_get_pulses: ");
    Serial.println(color_sense_get_pulses());
    currentTime = millis();

    if (numeroPulsos < numeroPulsosTeste + 3) {
      if (lastPulse != color_sense_get_pulses()) {
        if (numeroPulsos < 3) {
          numeroPulsos++;
          lastPulse = color_sense_get_pulses();
          lastPulseTime = currentTime;
        }
        else {
          numeroPulsos++;
          lastPulse = color_sense_get_pulses();
          pulseDurationTime = currentTime - lastPulseTime;
          lastPulseTime = currentTime;
          totalTime += pulseDurationTime;
          Firebase.setInt("testePrecisao/pulsoAtual", numeroPulsos - 3);
          statusTeste = "obtendoValores";
          Firebase.setString("testePrecisao/status", statusTeste);
        }
      }
```

```
    }
    else {
      reconectar_wifi();
      Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
      potenciaTotal = ((numeroPulsosTeste) * constanteKh * 3600000) / (totalTime);
      Firebase.setFloat("testePrecisao/potenciaFinal", potenciaTotal);
      statusTeste = "finalizado";
      Firebase.setString("testePrecisao/status", statusTeste);
      modo = "inicio";
      Firebase.setString("/info/modo", modo);
      Firebase.setString("/info/status", "desconectado");
      totalTime = 1;
      numeroPulsos = 0;
      return modo;
    }
  }
}
```

APÊNDICE F – Código para obter curva de carga

```
String obter_curva_carga() {
  String selectedMedicaoKey;
  float constanteKh;
  String nomeCurva;
  String valorIntervalo;
  String statusMedicao;
  float intervalo;
  int initTime;
  int lastPulse;
  int currentTime;
  boolean isFirstPulse = true;
  int lastPulseTime;
  int numeroPulsos = 0;
  int totalTime = 1;
  float potenciaTotal;
  int codSet = 1;
  String jsonPotencia;
  int pulseDurationTime = 0;

  exibir_display_curva_carga();

  selectedMedicaoKey = Firebase.getString("/info/selectedMedicaoKey");
  constanteKh = Firebase.getFloat("medicoes/" + selectedMedicaoKey + "/constanteKh");
  nomeCurva = Firebase.getString("medicoes/" + selectedMedicaoKey + "/nome");
  valorIntervalo = Firebase.getString("medicoes/" + selectedMedicaoKey + "/intervalo");
  statusMedicao = Firebase.getString("medicoes/" + selectedMedicaoKey + "/status");

  if (valorIntervalo == "1 minuto")
    intervalo = 1;
  else if (valorIntervalo == "5 minutos")
    intervalo = 5;
  else if (valorIntervalo == "15 minutos")
    intervalo = 15;
  else if (valorIntervalo == "30 minutos")
    intervalo = 30;
  else if (valorIntervalo == "1 hora")
    intervalo = 60;

  exibir_display_dados_curva_carga(nomeCurva, constanteKh, valorIntervalo);

  initTime = millis();
  lastPulse = color_sense_get_pulses();

  while (statusMedicao == "ObtendoDados") {
    color_sensor_tick();
    Serial.print("color_sense_get_pulses: ");
    Serial.println(color_sense_get_pulses());
    currentTime = millis();

    if ((currentTime - initTime) < intervalo * 60000) {
      if (lastPulse != color_sense_get_pulses()) {
        if (isFirstPulse) {
          lastPulseTime = currentTime;
          lastPulse = color_sense_get_pulses();
          isFirstPulse = false;
```

```
        }
        else {
          numeroPulsos++;
          lastPulse = color_sense_get_pulses();
          pulseDurationTime = currentTime - lastPulseTime;
          lastPulseTime = currentTime;
          totalTime += pulseDurationTime;
        }
      }
    }
    else {
      reconectar_wifi();
      Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
      potenciaTotal = ((numeroPulsos) * constanteKh * 3600000) / (totalTime);
      initTime = millis();
      numeroPulsos = 0;
      totalTime = 1;
      DynamicJsonBuffer jsonBuffer;
      jsonPotencia = "{\"x\":␣" + String(codSet) + ",\"y\":␣" + String(potenciaTotal) + "
          }";
      Firebase.set("/medicoes/" + selectedMedicaoKey + "/dados/" + String(codSet),
          jsonBuffer.parseObject(jsonPotencia));

      if (Firebase.failed()) {
        reconectar_wifi();
        Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
        Firebase.set("/medicoes/" + selectedMedicaoKey + "/dados/" + String(codSet),
            jsonBuffer.parseObject(jsonPotencia));
      }
      codSet++;
      statusMedicao = Firebase.getString("/medicoes/" + selectedMedicaoKey + "/status");
    }
  }

  if (statusMedicao == "finalizado") {
    modo = "inicio";
    Firebase.setString("/info/modo", modo);
    Firebase.setString("/info/status", "desconectado");
    codSet = 1;
    isFirstPulse = true;
    initTime = millis();
    numeroPulsos = 0;
    totalTime = 0;
    return modo;
  }
}
```