CENTRO UNIVERSITÁRIO FEI

ARTHUR MARTINS DA SILVA
DANIEL YEIDI UEHARA
GABRIEL ALMEIDA MACHADO
JONATHAS SOARES SOUZA

SISTEMA INTELIGENTE DE MEDIÇÃO DE ENERGIA ELÉTRICA

São Bernardo do Campo
2019

ARTHUR MARTINS DA SILVA
DANIEL YEIDI UEHARA
GABRIEL ALMEIDA MACHADO
JONATHAS SOARES SOUZA

SISTEMA INTELIGENTE DE MEDIÇÃO DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica. Orientado pelo Prof. Dr. Silvio Xavier.

São Bernardo do Campo
2019

Sistema inteligente de medição de energia elétrica / Arthur Martins da Silva...[et al.]. São Bernardo do Campo, 2019.
117 p. : il.

Trabalho de Conclusão de Curso - Centro Universitário FEI.
Orientador: Prof. Dr. Silvio Xavier Duarte.

1. Energia elétrica. 2. Medidor de energia. 3. Transmissão de dados. 4. Aplicativo para usuário. I. Silva, Arthur Martins da. II. Uehara, Daniel Yeidi. III. Machado, Gabriel Almeida. IV. Souza, Jonathas Soares . V. Duarte, Silvio Xavier, orient. VI. Título.

Elaborada pelo sistema de geração automática de ficha catalográfica da FEI com os dados fornecidos pelo(a) autor(a).

ARTHUR MARTINS DA SILVA
DANIEL YEIDI UEHARA
GABRIEL ALMEIDA MACHADO
JONATHAS SOARES SOUZA

SISTEMA INTELIGENTE DE MEDIÇÃO DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso, apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Comissão julgadora

_____

Orientador e presidente

_____

Examinador (1)

_____

Examinador (2)

São Bernardo do Campo
04 de novembro de 2019

Dedicamos este trabalho aos nossos familiares e amigos que contribuíram na nossa formação pessoal e acadêmica durante todo o curso e neste trabalho de conclusão de curso.

AGRADECIMENTOS

Agradecemos a Prof. Dra. Michele Rodrigues, ao Prof. Dr. Silvio Xavier e ao Prof. Dr. Marcelo Parada pela orientação ao longo do processo deste projeto.

Aos professores do Centro Universitário da FEI que contribuíram para a elaboração de todo o conteúdo deste trabalho de conclusão de curso.

A todos os familiares e amigos pelo apoio durante a jornada acadêmica, e durante a conclusão deste trabalho.

Agradecemos também ao Centro Universitário da FEI por nos apoiar com todos os materiais e infraestrutura.

Por fim, a Deus que nos acompanhou em mais uma etapa de nossas vidas nos concedendo força para concluir este ciclo acadêmico.

“Eu gosto do impossível porque lá a concorrência é menor”

(DISNEY, 1965)

RESUMO

Este trabalho apresenta o projeto e o desenvolvimento de um protótipo de medição de grandezas elétricas que auxilia o usuário final a gerenciar o consumo de energia elétrica de sua residência. O sistema desenvolvido é um medidor de tensão elétrica e corrente elétrica alternada, cujo núcleo é um módulo microcontrolado por um ESP32, que aproveita os recursos nativos à plataforma Arduino IDE.

As funcionalidades do protótipo são as medições das grandezas elétricas fundamentais, cálculo da potência ativa em tempo real, cálculo do consumo de energia mensal e o custo mensal da residência, a fim de apresentar valores confiáveis, os resultados são apurados através de sistemas de redundância de dados. Outro ponto é a transmissão dos dados em longas distâncias, utilizando um protocolo de comunicação que permite a transferência de arquivos, tal como LoraWan, uma tecnologia que usa rádio frequência que permite comunicação a longas distâncias, com consumo mínimo de energia.

Além disso, têm-se como objetivo a utilização de um aplicativo que roda em sistemas iOS e Android com comunicação ao medidor de energia, onde a empresa responsável pelo fornecimento de energia tem o acesso para realizar as devidas análises técnicas de fornecimento e o usuário final para avaliar as despesas com consumo de energia em diferentes períodos. Todo este sistema é interativo, pois permite ao consumidor definir metas de despesas e receber mensagens a respeito das metas batidas ou caso detecte falha de energia elétrica na rede.

Palavras-chave: Energia elétrica. Medidor de energia. Transmissão de dados. Aplicativo para usuário.

ABSTRACT

This paper presents the design and development of a prototype for measuring electrical quantities in which assists the customer in managing the electricity consumption of their home. The system developed is a voltage and alternating current meter, whose core is an ESP32 module, which uses advantage of features native to the Arduino IDE platform.

The functionalities of the prototype are the measurements of the fundamental electrical quantities, the calculation of the active power in real time, the calculation of the monthly energy consumption and the monthly cost of the residence, in order to present reliable values, the results are determined through data redundancy systems. Furthermore, another analysis is the transmission of data over long distances using a communication protocol that allows files transfer, such as LoraWan, a technology which uses radio frequency and allows long distance communication with minimal power consumption.

In addition, the goal is to develop an application, on iOS and Android systems, with communication to the power meter, where the company responsible for the power supply has the access to perform the appropriate technical analysis of energy supply and for the costumer evaluate energy consumption expenses at different periods. This entire system is interactive as it allows the consumer to set expense goals and receive messages about missed goals or if power failure detection in the grid.

Keywords: Electrical energy. Power meter. Data transmission. User app.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O consumo de energia elétrica é o motor da nossa sociedade, onde diversas atividades, das mais simples às mais sofisticadas, exige um consumo de energia elétrica, que por sua vez demanda uma crescente necessidade de produção de mais energia através da construção de usinas hidrelétricas, eólicas, solares, entre outras.

De acordo com o Ministério de Minas e Energia (MME, 2019) e da Agência Nacional de Energia Elétrica (ANEEL, 2019), o consumo residencial de energia elétrica no Brasil era aproximadamente 30% do total de energia gerada, e apesar da queda neste percentual em 2015, o consumo por pequenos consumidores era próximo a 22%. Conforme a Empresa de Pesquisa Energética (EPE, 2019) existe um crescimento médio próximo a 2,5% ao ano.

Com esta constante necessidade de aumentar a sua matriz energética e uma sucessão de erros na política energética do Brasil nas últimas décadas resultaram em uma das tarifas de energia mais caras do mundo, com tendência de agravamento para os próximos anos.

Além disso, a criação de diferentes taxas e impostos, aonde chega até 26 taxas, cobrada direta ou indiretamente, como IPTU, IPVA, PIS e carvão, são incluídos na conta de luz e por consequência, aumentam o valor a ser pago e dificulta o entendimento do consumidor final para identificar os seus gastos. (MAX, 2018). A empresa de energia Eletropaulo, atualmente comprada pela empresa ENEL, alega que a cada R$ 100 pagos pelo cliente, só R$ 15,40 são destinados às atividades da distribuidora, o restante é direcionado para pagamento de custos de energia, tributos e impostos (ELETROPAULO, 2018).

Outro custo adicional ao consumidor pode ser devido ao erro ou demora na aferição do medidor de energia residencial, como ainda existem muitos medidores de energia analógicos, existe a possibilidade que o profissional que realizar estas medições se enganar e escrever a potência consumida errada ou no caso da utilização de um medidor que realiza esta aferição, trocar os relatórios obtidos com outra residência, resultando no aumento da fatura.

A conta de luz por estimativa, quando a concessionária deixa de fazer a leitura do relógio marcador e emite uma conta ponderando uma média de consumo dos 12 últimos meses, é uma prática que as empresas utilizam principalmente para zonas rurais e locais de difícil acesso, é uma prática que pode ser feita no máximo durante 3 meses seguidos, e quando a leitura finalmente ocorre, costuma trazer, acumulada, as diferenças de cada mês em que não houve a leitura, podendo aumentar a conta exponencialmente (BENEVIDES, 2018).

2 PROPOSTA DE PROJETO

A partir dos problemas apresentados acima, o cliente final recebe uma conta, com a aplicação de diferentes taxas e tributos, que não possui confiança em sua leitura, tampouco o conhecimento de seus gastos diários e locais que geram um maior custo, o resultado disto é a impossibilidade de realizar o controle de gastos e a previsão do valor final de sua conta de luz.

Logo, a proposta deste projeto é o desenvolvimento de um sistema que auxilie o usuário final a gerenciar o consumo de energia elétrica. O sistema a ser apresentado consta de um medidor de tensão e corrente elétrica alternada, cujo núcleo é um módulo microcontrolado por um ESP32, que aproveita os recursos nativos à plataforma Arduino IDE. As funcionalidades do protótipo são as medições já citadas, o cálculo da potência ativa em tempo real, apuração dos dados através de sistemas de redundância de dados. Outro ponto é a transmissão dos dados em longas distâncias, utilizando um protocolo de comunicação que permita uma transferência de arquivos, tal como LoraWan, uma tecnologia que usa rádio frequência que permite comunicação a longas distâncias, com consumo mínimo de energia,

Para o desenvolvimento do aplicativo, serão utilizadas plataformas previamente desenvolvidas, tais como Blynk, uma plataforma de aplicativos que rodam em sistemas iOS e Android para a interação com a empresa responsável pelo fornecimento de energia que terá o acesso para realizar as devidas análises técnicas de fornecimento, e para o usuário, têm-se como objetivo monitorar o custo de energia e pois permite ao consumidor definir metas de despesas e receber mensagens a respeito das metas batidas ou caso detecte falha de energia elétrica na rede.

2.1 OBJETIVOS

Gerais:

a) Sistema de fácil visualização da conta de luz através de um aplicativo para o usuário;
b) Medição em tempo real do consumo de energia elétrica de uma residência;
c) Armazenamento personalizado do histórico do consumo elétrico na nuvem;
d) Detalhamento do consumo de energia por períodos fixos;
e) Transmissão de dados em longa distância para zonas de difícil acesso.

Específicos:

a) Conferir os dados e possíveis falhas na rede elétrica através do sistema;
b) Aplicativo com informações detalhada da potência consumida, valores envolvidos na conta de luz, previsão da conta a ser paga.

3 REVISÃO BIBLIOGRÁFICA

Na fundamentação teórica será abordado o sistema de energia elétrica no Brasil, o conceito de internet das coisas (IoT) e a conectividade entre os equipamentos, assim como os medidores de energia utilizados atualmente. Além disso, serão abordados os conceitos de medidores de energia, microcontroladores e os módulos auxiliares para o desenvolvimento deste projeto.

Ademais, será apresentado um aplicativo voltado ao monitoramento e registro dos dados na nuvem, os quais poderão ser analisados pelo consumidor ou fornecedor de energia elétrica.

3.1 CONCEITO DE ENERGIA ELÉTRICA NO BRASIL

O conceito de energia elétrica no Brasil inicia-se desde a geração da energia, até o momento em que chega à residência do consumidor final. Podendo assim, ser classificado em quatro pilares: Geração, Transmissão, Distribuição e Comercialização de energia. É de responsabilidade da (ANEEL, 2019) regulamentar e fiscalizar estes pilares, conforme a (BRASIL, 1996).

No Brasil, a principal fonte de geração é a hidrelétrica (água corrente dos rios), que responde por 62% da capacidade instalada em operação no país, seguida das termelétricas (gás natural, carvão mineral, combustíveis fósseis, biomassa e nuclear), com 28%. O restante é proveniente de usinas eólicas (energia dos ventos) e importação da energia de outros países (ANEEL, 2019).

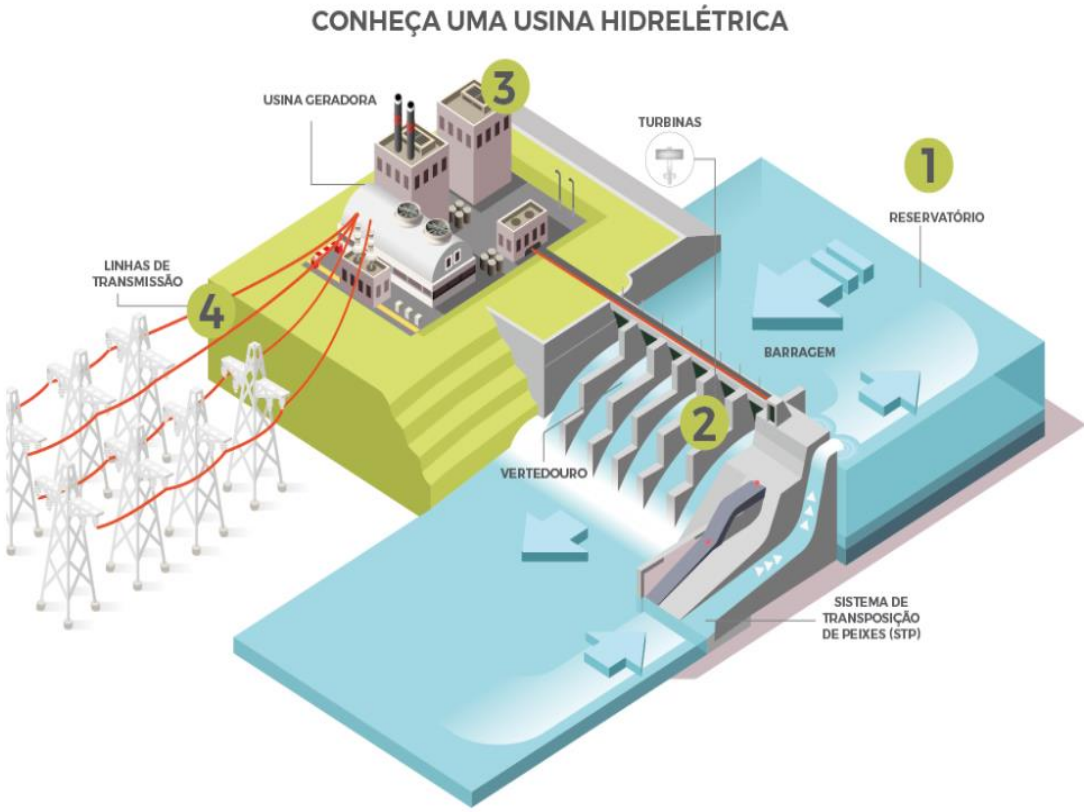
3.1.1 Geração de energia elétrica

O motivo de novas construções de hidrelétricas no Brasil se deve ao fato do país ser banhado por muitos rios com grandes desníveis, sendo uma das soluções mais econômicas, pois aproveita a força das águas para realizar a movimentação de turbinas (ELETROBRAS, 2017).

São construídas barreiras para o direcionamento da água até as turbinas, onde há um gerador em cada turbina, formando um conjunto de geradores que a partir do movimento das pás da turbina, converte energia mecânica em energia elétrica.

As Usinas de Aimorés e Funil da empresa (ALIANÇA, 2017) possuem um sistema de reservatório chamado fio d'água, o qual aproveita o fluxo de água do rio, ou seja, sua vazão determina a quantidade de energia gerada. O volume de chuvas tem impacto direto na geração de energia nas usinas hidrelétricas, aumentando sua produção. Toda a água que chega pelo rio é utilizada para a geração, por isso, não há acúmulo nos períodos de cheia, tampouco existe desperdício de energia acumulada. Todo esse processo é apresentado na Figura 1.

Figura 1 - Exemplo de usina hidrelétrica



Fonte: (ALIANÇA, 2017)

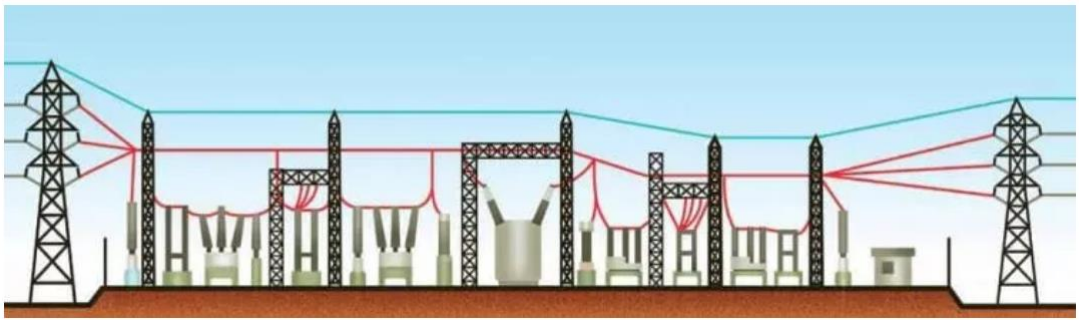
3.1.2 Transmissão de energia

Os sistemas elétricos são conectados através do Sistema Interligado Nacional (SIN, 2018), formando uma malha de transmissão, possibilitando que a energia produzida em uma determinada região do país seja transportada e utilizada em outro.

Conforme descreve (BARROS, 2009), o papel desse sistema é interligar a geração de energia até a carga e interligar as diferentes regiões do País, de forma a permitir a exploração racional dos recursos disponíveis em todas as regiões.

O Sistema Interligado Nacional (SIN, 2018) interliga as regiões do Brasil, porém ainda há áreas isoladas, principalmente na região norte. Com o passar dos anos, estas áreas vêm sendo interligadas gradativamente ao sistema, a Figura 2 exemplifica o conceito de transmissão de energia sobre uma cidade.

Figura 2 – Exemplo de uma transmissão de energia



Fonte: (ABRADEE, 2018)

3.1.3 Distribuição de energia

Atualmente, as distribuidoras de energia são independentes e caracterizam a ligação entre a população e o setor elétrico através do suprimento de energia elétrica que recebem das companhias de transmissão do país.

A energia elétrica no Brasil é compartilhada por diferentes empresas de distribuição, que varia conforme a sua localização. Esta rede elétrica pode ser de tipo aéreo, sustentada por postes, ou subterrânea, sustentada por dutos instalados no subsolo (PEVEDUTO, 2017).

O sistema de distribuição de energia é feito através de cabos elétricos, que podem ser de alta, média e baixa tensão. A linha de média tensão é a mais comum, pois pode ser vista nas ruas e avenidas das grandes cidades.

Existem inúmeras vantagens na construção e instalação da rede elétrica de forma subterrânea, dentre elas, a diminuição da poluição visual causada pela exposição dos cabos e postes nas ruas. Desta forma, normalmente as construtoras que optam por esse tipo de instalação, maximizam o valor dos imóveis da região (TRISUL, 2019).

A distribuição de energia elétrica no Brasil é feita por meio da integração da produção, transmissão e distribuição ao consumidor final. Essa integração é motivada pela industrialização e urbanização, pelo aumento da demanda e pela origem das hidrelétricas.

3.1.4 Comercialização de energia

Em 2014, foi criada a Câmara de Comercialização de Energia Elétrica (CCEE, 2019) que tem como objetivo viabilizar as negociações de compra e de venda de energia no Brasil. É a responsável pela contabilização e pela liquidação financeira no mercado de curto prazo de energia. A instituição é incumbida do cálculo e da divulgação do preço de liquidação das diferenças (PLD), utilizado para valorar as operações de compra e venda de energia.

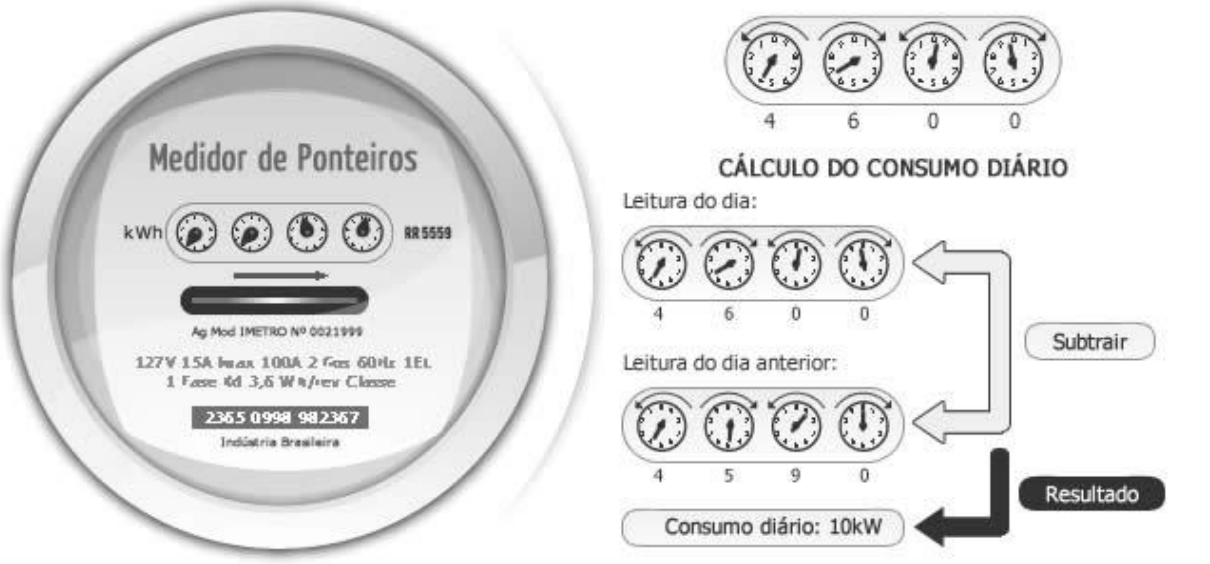
3.1.5 Medição de energia

No medidor, o leiturista da concessionária verifica o consumo (kWh) indicado no relógio do medidor e subtrai pelo valor da medição obtida no mês anterior. Logo, esta diferença indica o consumo de energia daquele local no mês, a Figura 3 descreve tal operação.

Exemplo: Se o medidor indicar o consumo de 1100 kWh e no mês anterior indicava 1000 kWh. O consumo no mês atual é de 100 kWh.

Caso ele não consiga fazer a leitura, o consumo é estimado, denominado custo por estimativa, calculado pela média dos últimos 12 meses.

Figura 3 – Leitura do consumo diário através do medidor de energia



Fonte: (COELBA, 2018)

3.1.6 Conta de energia

Muitos consumidores desconhecem a composição da conta de energia paga mensalmente, devido à aplicação de diferentes taxas e impostos implicados ao valor final. Além disso, as tarifas podem variar de acordo com o consumo, horário, região e até com as condições climáticas.

A tarifa total é composta pela soma de:

a) Tarifa de energia (TE): Custo da geração de energia;

   Exemplo: em dez/18, a TE custava R$ 0,37 por kWh consumidos (ENEL - SP).

b) Tarifa de uso do sistema de distribuição (TUSD);

   Exemplo: em dez/18, a TUSD custava R$ 0,29 por kWh consumidos (ENEL – dez/18).

c) Tarifa por bandeira: Sinaliza as condições de geração de energia pelas hidrelétricas no período.

   O ano de 2018 foi um ano com déficit hídrico, então as tarifas das bandeiras foram atualizadas (junho/2019).

   - Verde (Boas condições de geração): Não há cobrança extra. A geração de energia está dentro da normalidade
   - Amarela (Sinal de alerta): Há cobrança extra de R$ 1,50 por 100 kWh consumidos (proporcional)
   - Vermelha I (Condições ruins de geração): Há cobrança extra R$ 4,00 por 100 kWh consumidos (proporcional)
   - Vermelha II (Condições ruins de geração): Há cobrança extra de R$ 6,00 por 100 kWh consumidos (proporcional)

Com isto, a conta de energia paga pelo consumidor é a tarifa homologada (Tarifa de Energia + Tarifa do Uso de Distribuição + Tarifa por Bandeira). Acrescida dos impostos que são aplicados de acordo com a região e podem sofrer variações de acordo com o consumo. O valor cobrado ao usuário no final é estipulado conforme a fórmula abaixo (BARROS, 2009).

$$\text{Valor cobrado} = \frac{Tarifa\ homologada\ pela"\ ANEEL"}{1 - [(Alíquota\ efetiva\ do\ "PIS"\ e\ "COFINS") + "ICMS"]}$$

3.2 MEDIDORES DE ENERGIA

O medidor de energia é popularmente conhecido como “relógio de luz”, e tem a função de medir o consumo de energia em quilowatt-hora (kWh).

Neste tópico serão apresentados os medidores utilizados no Brasil que são instalados pelas concessionárias (eletromecânico e eletrônico) e suas principais características.

3.2.1 Medidor eletromecânico

O medidor eletromecânico é o mais antigo, mas continua como o mais utilizado no Brasil. Seu funcionamento basicamente consiste na indução eletromagnética, provocada pela eletricidade percorrida nas bobinas. O disco gira, movimentando as engrenagens e consequentemente, os ponteiros do medidor, na Figura 4 é possível observar o medidor instalado no Brasil.

Nansen, uma das últimas fabricantes deste modelo de medidor atuando no Brasil, relata que “a medição com tecnologia eletromecânica, robusta e confiável, dá lugar à necessidade de melhorar a gestão pela análise da demanda, através de medidores inteligentes, capazes de dialogar com uma rede que se moderniza em todos os níveis” (NANSEN, 2019).

Por isso, existe atualmente um investimento muito grande para a transição de medidores analógicos para medidores digitais, a fim de modernizar e melhorar o sistema de energia com os conceitos de *smart grids* e medidores inteligentes.

Figura 4 - Medidor de energia eletromecânico

Fonte: (ANEEL, 2009)

3.2.2 Medidor eletrônico

Segundo (BARROS, 2009), “a classe de precisão mais apurada dos medidores eletrônicos tem feito o mercado optar cada vez mais por esta tecnologia”.

A medição deste equipamento é realizada de forma digital, possui a capacidade de enviar as informações de consumo de energia para a concessionária, dispensando a leitura presencial no medidor. Outras vantagens com relação ao eletromecânico: registros, precisão, segurança contra furto de energia, estabelecimento de tarifa horária, a Figura 5 apresenta o medidor eletrônico da empresa *Nansen* (NANSEN, 2019).

Figura 5 - Medidor de energia eletrônico

Fonte: (NANSEN, 2019)

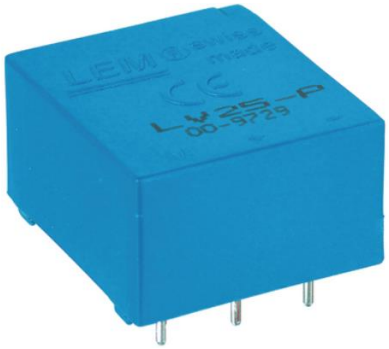
3.3 MEDIDOR DE TENSÃO

Após a contextualização da matriz energética no Brasil e dos medidores utilizados. Foram analisados possíveis medidores e sensores das grandezas elétricas aplicadas a um medidor de energia elétrica. Desta forma, na fase de planejamento do projeto, foi utilizado sensor de tensão como proposta para realização dos testes e até mesmo, incluir no projeto final.

Trata-se do sensor de tensão LV 25-P, conforme apresentando na Figura 6, utilizada para medição eletrônica de corrente contínua e alternada.

Para medição de tensão, uma corrente proporcional à tensão medida deve passar através de um resistor externo, instalado em série com o circuito primário do transdutor. Este sensor é indicado para medir cargas com tensões nominais de 10 V até 500 V (LEM, 2019).

Figura 6 - Sensor de tensão LV 25-P



Fonte: (DISTRELEC, 2019)

3.4 MEDIDOR DE CORRENTE

Da mesma maneira que o sensor de tensão foi incluído no planejamento do projeto, foi realizado um estudo para a aplicação dos sensores de corrente invasivos e não-invasivos.

3.4.1 Sensor de corrente LA 125-P

O sensor LA 125-P foi desenvolvido para medição de corrente contínua e alternada, o mesmo realiza medições na faixa de 0 a 200A. Normalmente é utilizado para o monitoramento da corrente de acionamentos de velocidade variável CA/servomotor, conversores estáticos para inversores de motores CC, suprimento de energia, entre outros, a Figura 7 apresenta o seu formato (LEM, 2019).

Este é um sensor de corrente invasivo, logo é necessário realizar a abertura do circuito para a implementação do mesmo.

Figura 7 - Sensor de corrente LA 125-P

Fonte: (RSDELIVERS, 2019)

3.4.2 Sensor de corrente não invasivo SCT-013

Para a aplicação de um medidor de corrente não invasivo, ou seja, não há necessidade de contato elétrico com a carga a ser monitorada e ligá-lo em série para medição da corrente. Para realizar a medição, basta envolver o condutor entre o sensor. Pois, o mesmo possui a capacidade de medir correntes alternadas de até 20A RMS ou 100A RMS, dependendo do modelo, conforme Figura 8. O sinal de saída é através do plug P2, composto pelos dois fios que saem do sensor (ELECTRONICS, 2019).

Figura 8 - Sensor de corrente 20A SCT-013

Fonte: (ELETRÔNICA, 2019)

3.5 INTERNET DAS COISAS (IOT)

A *Internet of Things* (IoT) ou Internet das Coisas é um sistema de interconexão de dispositivos capazes de medir, filtrar, manipular, receber e transferir dados, muitas vezes entre os próprios dispositivos, classificando-os em uma comunicação M2M, *machine to machine* ou "de máquina para máquina", através de uma rede sem ou com pouca interação humana (TYAGI e JAIN, 2019).

Figura 9 – Conceito de Internet das coisas

Fonte: (GIPSON, 2019)

A "coisa" dentro da IoT pode ser classificada como qualquer objeto manufaturado, o qual pode ser atribuído a um endereço IP (ou *Internet Protocol Address*), que é uma identificação única para cada dispositivo conectado a uma rede, ou seja, o sistema é apto a enviar e receber dados através dessa rede. Em alguns casos, de acordo com a complexidade da "coisa" envolvida, é possível predizer o comportamento do objeto ou ambiente de atuação conforme a análise dos dados recebidos e tratados. Alguns exemplos básicos são um monitor de batimentos cardíacos, um chip de geolocalização de um animal de fazenda, um sensor automobilístico que informa ao piloto quando o carro precisa de uma manutenção corretiva ou medidores que disponibilizam dados de consumo de água ou energia de uma casa (ROUSE, 2019).

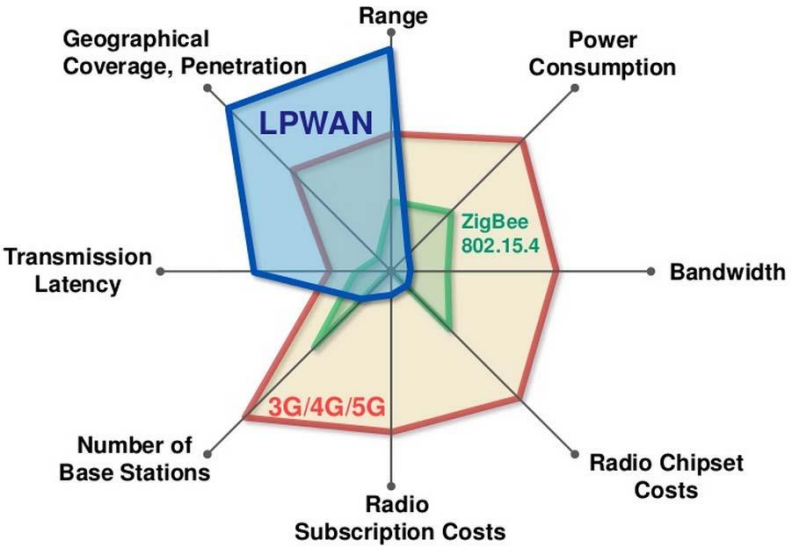
3.6 LPWAN

Conforme o número de dispositivos IoT foi crescendo ao decorrer dos anos, foi necessário a criação de uma nova tecnologia que abrigasse uma quantidade significativamente grande desses aparelhos sem onerar ou sobrecarregar a rede de internet atual.

Para isso foram criadas as LPWAN (*Low Power Wide Area Network* ou rede de área ampla e baixo consumo), que possuem como propósito a conexão de dispositivos IoT, os quais necessitam de um tamanho de mensagem e uma periodicidade muito menor de envio de dados do que uma pessoa ou uma plataforma de vídeo, por exemplo, e por consequência uma durabilidade de bateria muito maior (WANG e ZHAI, 2019).

Realizando a comparação, através da Figura 10, entre as tecnologias de LPWAN com outras redes celulares clássicas ou redes *mesh*, como *ZigBee*, é possível retirar algumas conclusões.

Figura 10 – Aplicação do LPWAN



Fonte: (LORA, 2019)

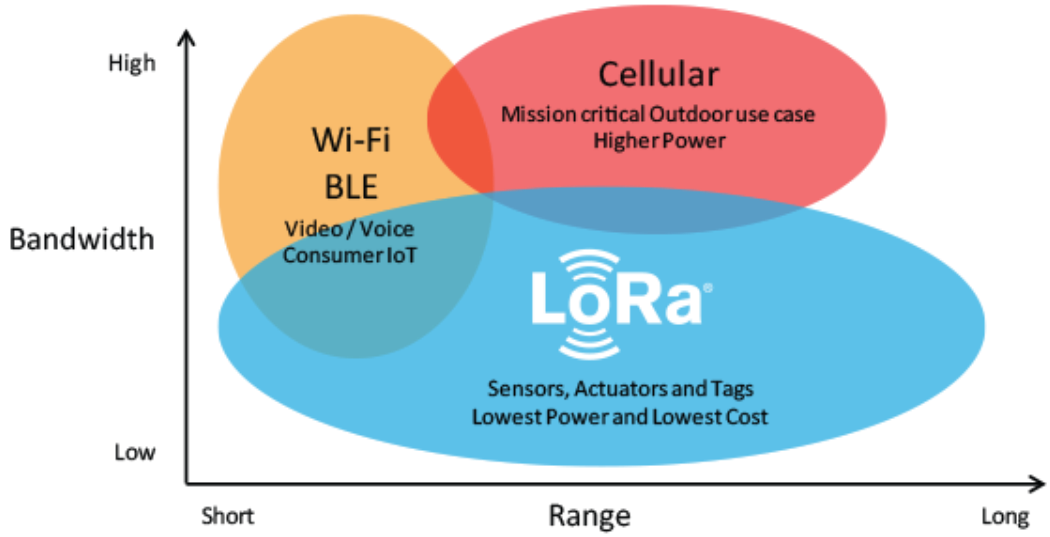
O LPWAN possui uma cobertura, distância e penetração maiores do que as tecnologias celulares convencionais. Contudo, há um aumento considerável de latência na transmissão, que para as aplicações em que esse sistema é utilizado não se torna impeditivo.

3.7 LORA

Uma das redes LPWAN é a LoRa (*Long Range* ou longa distância), que é uma das mais populares dentre as redes de conectividade, ao comparar esta rede com as demais, é identificar que esse sistema atinge longas distâncias, conforme apresentado na Figura 11.

Os módulos enviam e recebem dados de Gateways específicos (similar às redes Wifi, mas com alcance muito maior), que os encaminham via conexão IP para servidores locais ou remotos. Suas principais aplicações são sistema de IoT como sensores (pressão, luz, on-off, temperatura), sobretudo aqueles operados a bateria, de mensagens curtas e em alguns casos utilizados em locais de difícil acesso (SEMTECH, 2019).

Figura 11 – Comparação da tecnologia LoRa com as demais



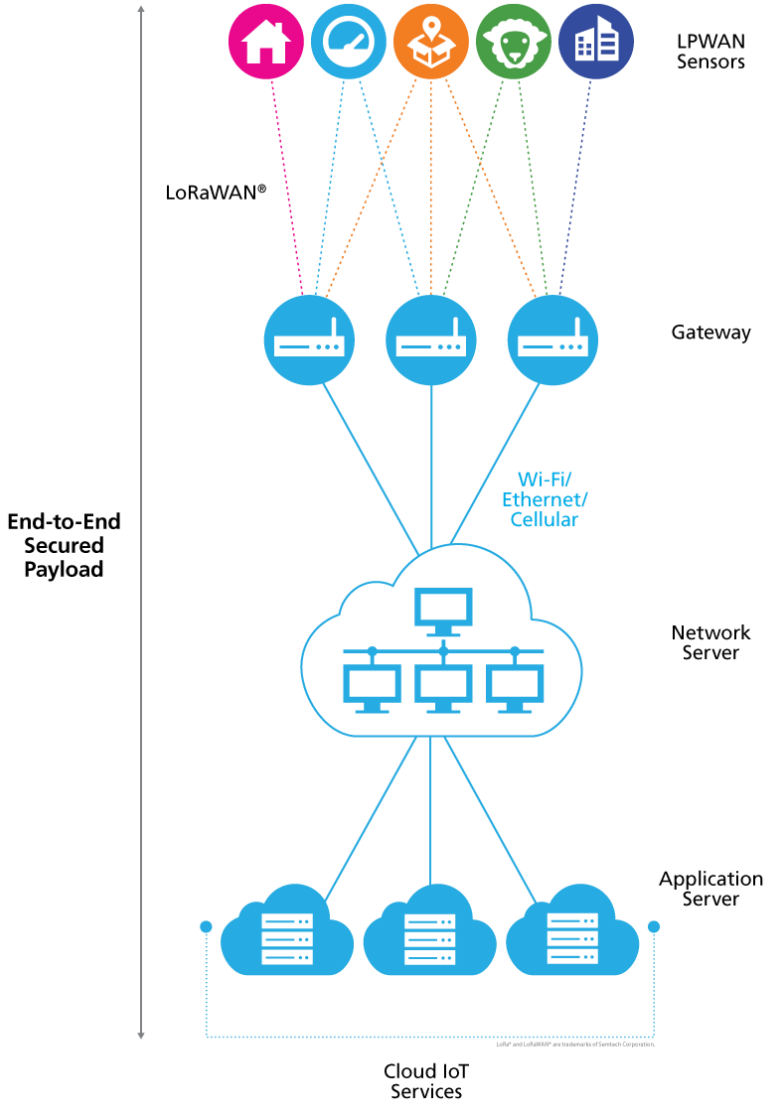
Fonte: (SEMTECH, 2019)

A tecnologia permite comunicações em longas distâncias, tipicamente 3 a 4 quilômetros em área urbana, com um pequeno consumo de energia. Em áreas rurais, com uma menor interferência no sinal, essa distância pode chegar até 15 quilômetros.

Além disso, a potência aplicada é muito baixa, da ordem de 20 dbm ou 100 mW. Exemplificando, um rádio LoRa configurado em potência máxima tem picos de 0,12 Ampères, este valor bem menor que a corrente de 2 Ampères de um modem que utiliza a tecnologia GSM.

Para que esse sistema funcione corretamente, são aplicadas diferentes camadas de aplicação, descritas na Figura 12.

Figura 12 – Descrição da rede LoRa



Fonte: (SEMTECH, 2019)

A tecnologia LoRa é constituída por diversas camadas, a primeira é a camada física (PHY) de silício, ou modulação sem fio, usada para criar o link de comunicação de longo alcance. Assim como, os transceptores configurados são incorporados aos nós finais ou sensores, projetados para uma infinidade de aplicações do setor.

Os sensores capturam e transmitem dados para gateways a distâncias próximas e distantes, internas e externas, com um mínimo de energia.

Os gateways enviam informações via Wi-Fi, Ethernet ou celular para o servidor de rede, responsável pelas funções de gerenciamento de rede, como ativação sem fio, tratamento de dados, roteamento dinâmico de quadros, controle de taxa adaptável, gerenciamento de tráfego, e administração (SEMTECH, 2019).

3.8 ESP-32

O ESP-32 é um microcontrolador de alto desempenho para aplicações envolvendo Wifi e Bluetooth, fabricado pela empresa *Espressif Systems*, apresentando um baixo consumo de energia. O mesmo possui 4 Mb de memória flash, 520 Kbytes de memória RAM e 448 Kbytes de memória ROM. Além disso, utiliza a CPU Tensilica Xtensa X6 de 32 bits, produzida pela empresa *Cadence,* com dois núcleos de processamento e um clock que pode variar entre 80 MHz até 240 MHz (BERTOLETI, 2019).

Esta placa permite o desenvolvimento de diferentes aplicações para projetos de IoT, tais como acesso remoto, *webservers* e *dataloggers*, entre outros. O ESP possui outras funcionalidades, entre elas, dois conversores digital-analógico (DAC), diferentes interfaces de comunicação e o protocolo de comunicação CAN (comunicação serial síncrona), onde realizando uma comparação com outros microcontroladores no mercado, os mesmos não possuem tais especificações, além do mais, o ESP-32 possui mais pinos de entrada e saída (GPIOs) do que os demais, tornando-o um dos melhores microcontroladores da atualidade. Na Figura 13 é demonstrada tal comparação.

Figura 13 – Comparação entre microcontroladores

	ESP32	ESP8266	ARDUINO UNO R3
Cores	2	1	1
Arquitetura	32 bits	32 bits	8 bits
Clock	160MHz	80MHz	16MHz
WiFi	Sim	Sim	Não
Bluetooth	Sim	Não	Não
RAM	512KB	160KB	2KB
FLASH	16Mb	16Mb	32KB
GPIO	36	17	14
Interfaces	SPI / I2C / UART / I2S / CAN	SPI / I2C / UART / I2S	SPI / I2C / UART
ADC	18	1	6
DAC	2	0	0

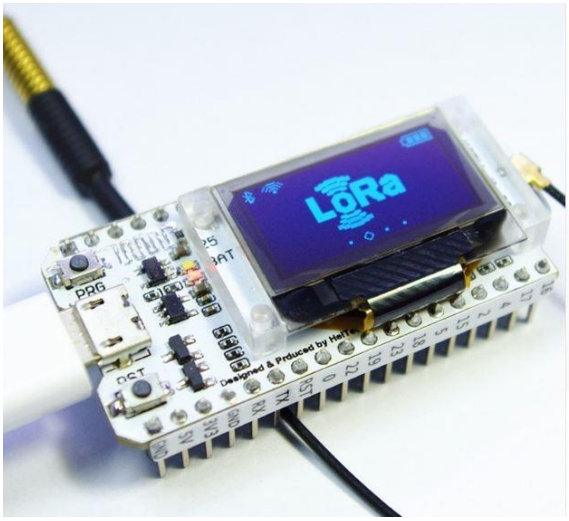
Fonte: (FERNANDO K, 2019)

3.8.1 ESP-32 LORA

A partir da decisão de utilizar o microcontrolador ESP-32 pelas diversas vantagens apresentadas, iniciou-se uma pesquisa com o objetivo de encontrar um sistema que envie as medições realizadas pelo sensor e receba tais dados em uma distância razoável. Com isso, o microcontrolador ESP-32 com o sistema LoRa integrado, desenvolvida pela empresa *Heltec Automation*, proporcionou todas as funcionalidades solicitadas para este projeto.

Essa placa de desenvolvimento integra três formas distintas de comunicação: Wifi, Bluetooth e a rede LPWAN de comunicação de longo alcance, denominada LoRa, permitindo a aquisição e envio de todas valores medidos ao usuário final por longas distâncias com baixa potência consumida (HELTEC, 2019). A Figura 14 apresenta o microcontrolador descrito.

Figura 14 – ESP-32 LORA da empresa Heltec Automation



Fonte: (HELTEC, 2019)

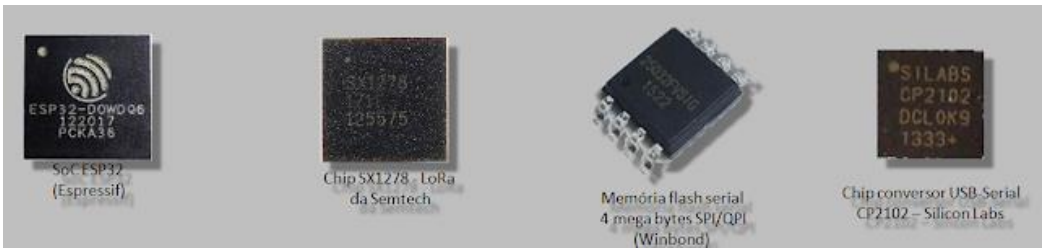
3.8.2 Descrição do ESP-32 LORA

A ESP32 LoRa é uma placa de desenvolvimento que tem como microprocessador o Xtensa 32-Bits LX6 que apresenta baixo consumo de energia, além de se destacar por ser um processador dual core que integra 4 chips:

a) O SoC ESP32-D0WDQ, que tem como principal diferencial a possibilidade de conexão Wi-Fi e Bluetooth.
b) O W25Q32FV referente a Memória Flash Serial (SPI/QPI), com aproximadamente 32 megabits (4 megabytes) de armazenamento.

c) O SX1276 LoRa, controlado pelo ESP32 que habilita a placa a trabalhar com a tecnologia LoRa comentada no tópico passado. De acordo com a tabela da Semtech (empresa fabricante do LoRa), temos 4 tipos de chips LoRa, onde possuem como principal diferença o alcance da frequência de transmissão dos dados.
d) O CP2102, da Silicon Labs, que tem como função possibilitar a comunicação serial através do USB.

Figura 15 – Configuração dos chips do ESP-32

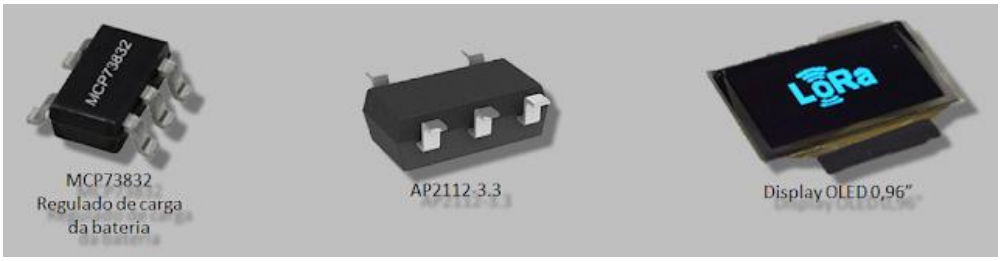


Fonte: (FERNANDO K, 2019)

Além destes circuitos integrados, há alguns periféricos que configuram outros benefícios. Entre os principais estão:

a) O MCP73831/2, controlador de carga de bateria de Íon de Lítio (Li-Ion) ou Polímero de Lítio (Li-Po) que possibilita o módulo ser alimentado por uma bateria.
b) O regulador AP2112-3.3 para fornecer 3,3V e um mínimo de 600mA quando a placa é alimentada através do USB que fornece 5v.
c) Dois cristais osciladores, sendo um de 26 MHz para o ESP32 e outro de 32 MHz para o SX1278 referente ao LoRa. Os dois servem como base da frequência de trabalho em ambos os integrados.
d) Um display “OLED 0.96”, com resolução de 128x64 pixels, este também controlado pelo ESP32.

Figura 16 – Composição do ESP-32



Fonte: (FERNANDO K, 2019)

3.9 PZEM-004T

O módulo PZEM é um medidor de tensão e corrente alternada (RMS) que realiza o cálculo do valor de potência ativa em Watt e energia consumida em Watt-hora. Alguma das vantagens que este módulo proporciona, consiste basicamente na alta isolação galvânica a partir dos opto acopladores (PC817), a comunicação TTL isolando cada parâmetro e por fim, a aquisição e armazenamento de dados com uma memória EEPROM. O sensor possui uma bobina TC (transformador de corrente) de 33 mm de diâmetro (SAOKAEW, CHIEOCHAN e BOONCHIENG, 2018).

Algumas informações técnicas deste módulo são descritas na Tabela 1.

Tabela 1 – Especificações do PZEM-004T

Especificações	**Descrições**
Tensão de trabalho	80 – 260 VAC
Medicação de corrente	0 - 100A
Potência nominal	22 kW
Frequência de operação	45 - 65 Hz
Precisão de medição	1,0 %

Fonte: Autor – "Adaptado de (INNOVATORS, 2019)".

Com estas especificações, este módulo pode ser aplicado para diferentes aplicações, pois possui um range de tensão e corrente elevado, podendo-se trabalhar até com circuitos trifásicos. Outro ponto importante é a precisão que o mesmo fornece, agregando uma alta confiabilidade nos resultados obtidos.

Para que o sistema funcione corretamente, o chip SD3004 da *SDIC Microeletronics Co. Ltd* realiza a conversão dos sinais lidos em 16 bits para pacotes de 8 bits (separando para cada valor de V, I, P e E), alocando-os em posições do mais significativo até o menos significativo, a UART irá receber estes pacotes de 8 bits.

O módulo também dispõe de um chip AT24C02N EEPROM (*electrically erasable programmable read-only memory*), formato serial com 2 Kbits da empresa *Microchip*, que permite um range de tensão de alimentação entre 4,5V até 5,5V. O chip comporta mais de 1 milhão de ciclos de escritas e correções, além de permitir aproximadamente 200 anos de retenção de dados, a Figura 17 apresenta os pinos que o PZEM possui.

Figura 17 – Pinagem do PZEM-004T

Fonte: (INNOVATORS, 2019)

A interface de comunicação de dados via porta serial TTL, é realizada através do protocolo UART (*Universal Asynchronous Receiver-Transmitter*), ou seja, possui uma comunicação assíncrona, não dependendo de nenhum clock. Com velocidade de transmissão de 9600 bps (bits por segundo), sendo 8 bits de dados, nenhum de paridade e 1 de bit de parada.

A comunicação com este módulo se concebe através de comandos AT (atenção), e conforme analisado, é uma linguagem de comandos que começou a ser utilizada em modems e atualmente é muito aceita na família dos microcontroladores ESPs (SENA, 2018).

A Tabela 2 apresenta a variação das grandezas elétricas medidas e suas devidas sensibilidades, informações importantes durante o desenvolvimento do projeto.

Tabela 2 – Range das medições do PZEM

Grandeza	Range do PZEM	Sensibilidade
Tensão (V)	80,0 – 260 V	± 0,1V
Corrente (A)	0,00 - 99,99V	± 0,01A
Potência (W)	0 – 22 kW	± 1 W (de 0 a 9999 W), ou ± 10 W(de 10000 a 22000 W)
Energia (kWh)	0 - 9999 kWh	± 1 Wh

Fonte: Autor – "Adaptado de (INNOVATORS, 2019)"

Na Tabela 3 é descrito o formato dos códigos do protocolo de comunicação, utilizando exemplos para facilitar o entendimento.

Tabela 3 – Protocolo de comunicação do PZEM

Função	Cabeçalho (Head)	Dados Enviados / Recebidos	Soma (sum)
Tensão (V)	B0	C0 A8 01 01 00 (Envio do valor de leitura da tensão)	1A
	A0	00 E6 02 00 00 (Exemplo de resposta que o valor da tensão é 230,2V)	88
Corrente (A)	B1	C0 A8 01 01 00 (Requisição leitura corrente)	1B
	A1	00 11 20 00 00 (Exemplo, responde 17,32A)	D2
Potência Ativa (W)	B2	C0 A8 01 01 00 (Requisição potência ativa)	1C
	A2	08 98 00 00 00 (Exemplo, 2200W)	42
Energia Consumida (kWh)	B3	C0 A8 01 01 00 (Requisição energia consumida)	1D
	A3	01 86 9f 00 00 (Exemplo, 99999Wh)	C9
Comunicação (TCP/IP)	B4	C0 A8 01 01 00 (Corresponde ao endereço 192.168.1.1)	1E
	A4	00 00 00 00 00 (Responde que a conexão foi estabelecida)	A4

Fonte: Autor – "Adaptado de (INNOVATORS, 2019)"

Analisando a Tabela 3, para cada par de letras e número, há um valor em hexadecimal que dependendo da posição equivale a uma parcela do valor lido. Por exemplo:

Para leitura da corrente, o µC envia o seguinte comando: B1 C0 A8 01 01 00 1B e o PZEM envia com a seguinte resposta: A1 00 11 20 00 00 D2. Analisando o início dos 3 pares da resposta, a mesma começa com 11 e o próximo par é 20, onde o número 11 corresponde ao número inteiro 17 em número decimal e o número 20 corresponde ao restante da corrente depois da vírgula, ou seja, o número inteiro 32 em número decimal. Totalizando assim 17.32 amperes.

Nota-se que todos os comandos de envio só mudam o cabeçalho (B1, B2, B3…). Outro ponto a se destacar é o último par enviado, ou seja, o sétimo par que corresponde à soma dos pares anteriores. Há a necessidade de fazer isso, pois o sistema não faz a conversão para verificar o valor, ele apenas compara o último par com a soma dos restantes em hexadecimal. Para maiores informações sobre o envio e transmissões de dados do PZEM, o *Datasheet* deste módulo encontra-se no ANEXO A.

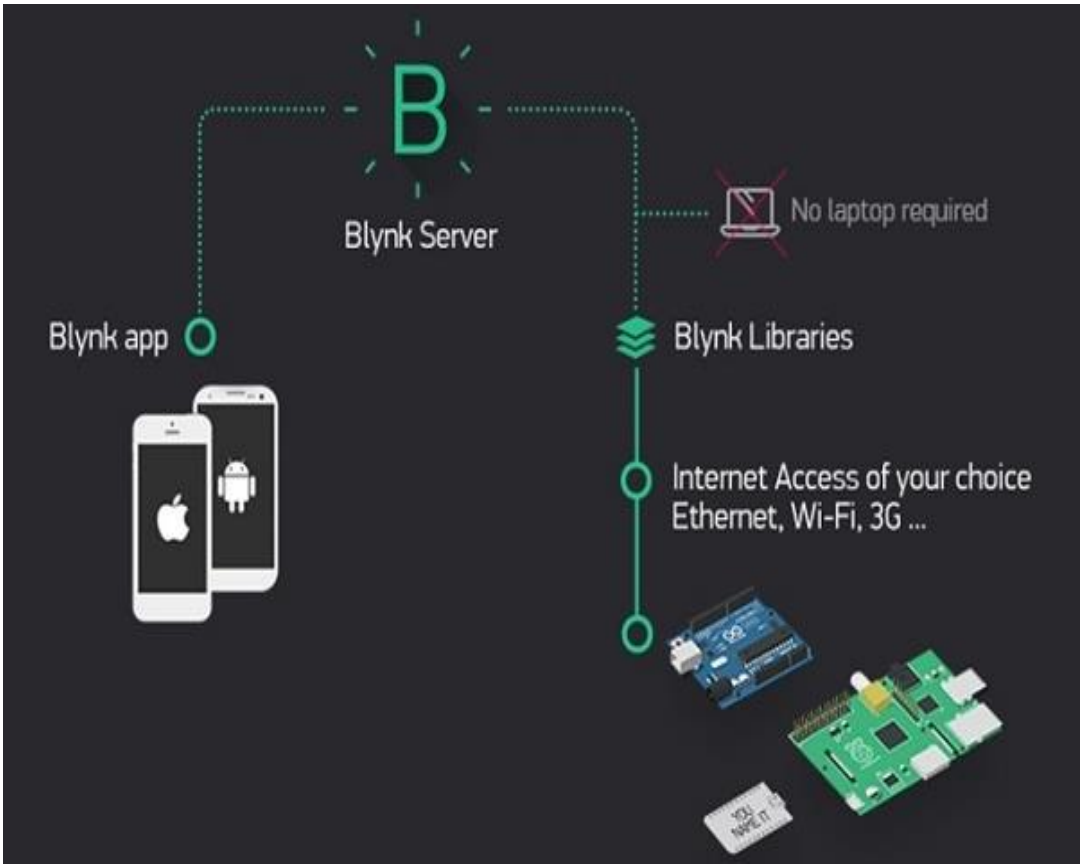
.

3.10 BLYNK

O Blynk é uma plataforma IoT (Internet of Things), ou seja, consiste em uma plataforma desenvolvida para dispositivos móveis para sistemas operacionais iOS e Android, permitindo a interação com diversos dispositivos embarcados, os quais se comunicam com um servidor através de bibliotecas pela internet, assim descartando o uso de notebooks ou desktops durante sua operação, demonstrado na Figura 18.

Esta plataforma utiliza o protocolo de comunicação um pouco diferente dos demais conhecidos no mercado, como Mosquitto, que utilizam o protocolo MQTT. Ela foi criada com o intuito de ser fácil de configurar e mexer onde o usuário precisa saber o mínimo de programação (NEVES, 2017).

Figura 18 – Descrição da comunicação entre a placa e o aplicativo



Fonte: (BLYNK, 2019)

Toda mensagem de envio do Blynk é constituída de 2 partes, o cabeçalho e o corpo, onde o cabeçalho é dividido em 3 etapas:

a) Comando de protocolo (1 byte);
b) Mensagem de identificação (2 bytes):
   - Corresponde a uma mensagem curta e com valor maior que zero. O campo de identificação da mensagem é um campo de 2 bytes que define o identificador exclusivo da mensagem. Utiliza-se para distinguir como gerenciar as respostas provenientes do hardware do cliente, em sua plataforma on-line móvel. O campo de identificação da mensagem deve ser gerado no lado do cliente;
c) Comprimento do corpo da mensagem (2 bytes):
   - Corresponde também a uma mensagem curta e com valor maior que zero. Esta mensagem composta por 2 bytes define o comprimento do corpo, podendo ser equivalente a zero se o corpo estiver vazio, caso contrário, não será preenchido com valor nulo.

A etapa que corresponde ao corpo na mensagem de envio do Blynk possui somente uma parte, sendo esta uma *string*, ou seja, uma cadeia de caracteres com tamanho variável. O corpo possui um tamanho de até 2^{15} bytes.

A partir do detalhamento do cabeçalho de forma sequencial, é possível visualizar sua estrutura através da Tabela 4, onde seus valores de comando, mensagem de identificação, tamanho da mensagem e corpo serão todos transmitidos em binário no formato *big endian*, onde os bytes são armazenados por ordem decrescente do seu peso numérico, ou seja, do bit mais significativo até o menos significativo.

Tabela 4 - Estrutura da mensagem de envio do Blynk.

Comando	**Mensagem de Identificação**	**Tamanho/Estado**	**Corpo**
1 byte	*2 bytes*	*2 bytes*	*Variável*

Fonte: Autor – "Adaptado de (BLYNK, 2019)"

Toda mensagem enviada pelo aplicativo Blynk corresponde basicamente a um comando, onde o servidor tem como prioridade responder toda e qualquer mensagem, ou seja, o mesmo direciona uma resposta para qualquer comando enviado.

Para os comandos como registro e *login*, por exemplo, não é solicitado nenhum retorno de dados, onde a mensagem retornada será zero. Entretanto, para os comandos *loadProfile e getToken*, que solicitam o retorno de um dado, é enviado uma mensagem com o mesma linha de comando, ou seja, '*loadProfile'*. Esta configuração é semelhante aos comandos AT utilizado no módulo PZEM.

Quanto às mensagens recebidas, é obtida uma estrutura diferente da anterior, exemplificada na Tabela 5, onde sua segmentação foi abordada anteriormente.

Tabela 5 – Estrutura da mensagem de resposta do Blynk

Comando	**Mensagem de Identificação**	**Código de resposta**
1 byte	*2 bytes*	*2 bytes*

Fonte: Autor – "Adaptado de (BLYNK, 2019)"

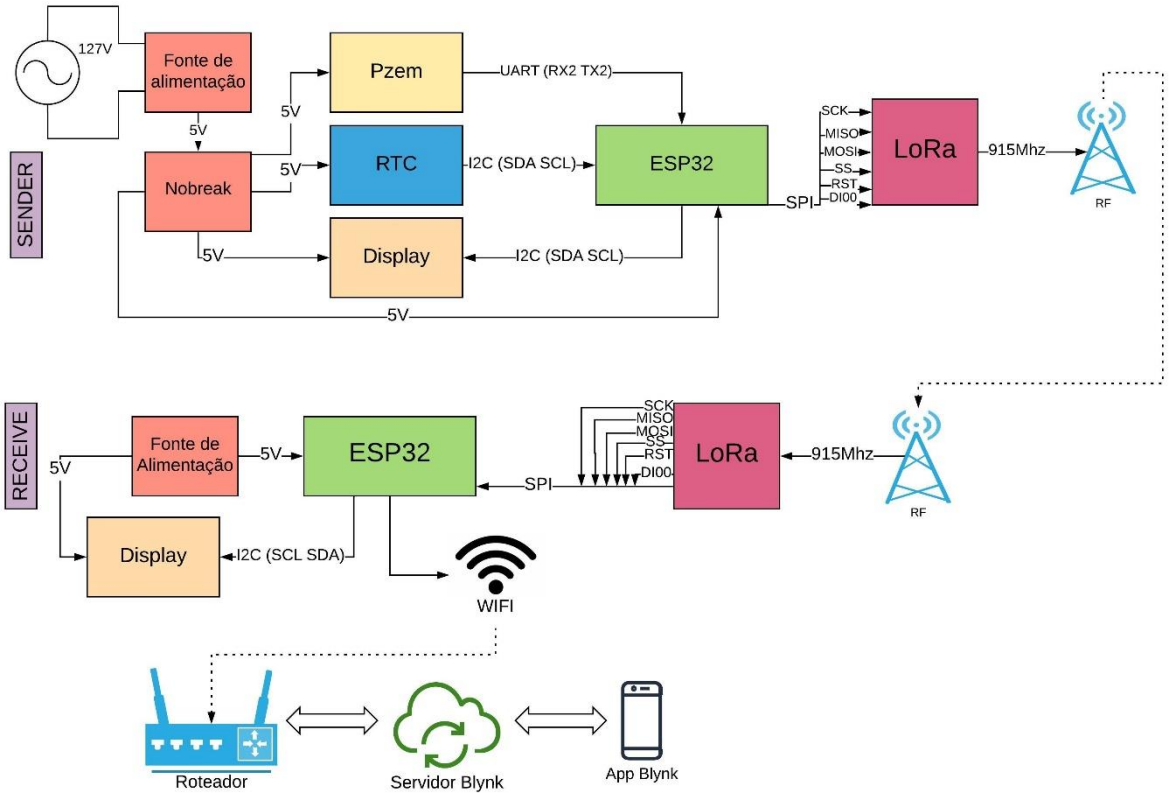
Esta plataforma *mobile* apesar de simples possui uma gama de ferramentas que outros aplicativos não fornecem, tampouco agregam uma interface com diferentes recursos quanto o Blynk. O app suporta mais de 400 sistemas embarcados e mesmo assim, os desenvolvedores fornecem a opção de compatibilizar qualquer outro hardware com a biblioteca da empresa.

A empresa Blynk também possui outro aplicativo chamado ESPproMon, com uma interface voltada mais para a área de energia, porém não foi utilizado neste projeto, por questões de incompatibilidade durante os testes de conexão com o servidor. Acredita-se que ainda é uma plataforma em desenvolvimento.

3.11 METODOLOGIA DO PROJETO

Durante os estudos para a execução do projeto, as etapas foram separadas em blocos, para que facilitasse durante a divisão e a sequência de pesquisas a serem feitas. Para isto, foi construído um diagrama de blocos com os principais módulos aplicados, conforme apresentado na Figura 19.

Figura 19 – Metodologia do projeto



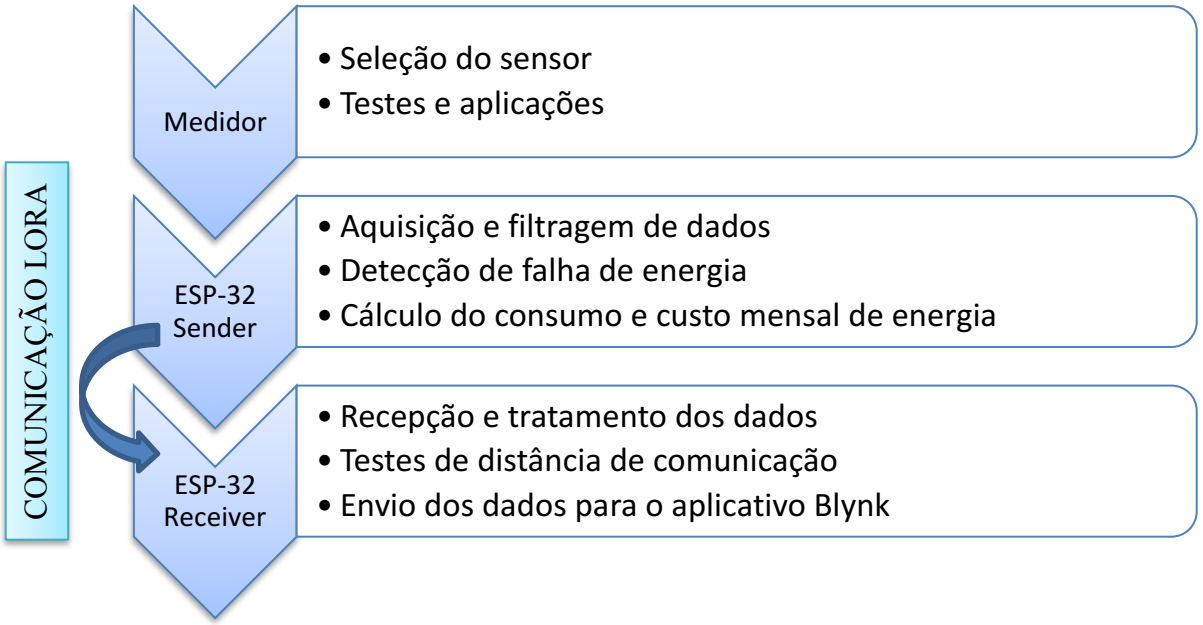
Fonte: Autor

Este diagrama foi utilizado para determinar a prioridade de execução das etapas, pois existem módulos que fornecem informações cruciais para que o seguinte possa ser avaliado com o dado correto, ou seja, antes de realizar a análise do envio dos dados via comunicação LORA, priorizou-se a definição e os testes com o sensor PZEM, pela a aquisição correta dos valores medidos. Desta forma, essa metodologia facilitou a resolução dos problemas durante os testes realizados.

4 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo serão abordados os procedimentos realizados para obter o protótipo final, conforme o fluxograma destacado abaixo na Tabela 6, desenvolvendo-o com os sensores e módulos apresentados no capítulo anterior. Além disso, são analisados aspectos importantes da plataforma de programação do ESP32, funcionalidades do sensor PZEM-004T, aquisição dos valores medidos e aplicação de um sistema de redundância sobre os dados coletados para filtragem, comunicação entre emissor e receptor conforme o protocolo LORA, recepção e tratamento dos dados, e por fim, envio e armazenamento do banco de dados no aplicativo Blynk. Ademais, será explicado como estes dados são apresentados em uma interface gráfica para que o usuário final tenha a informação da forma mais clara o possível e todo o desenvolvimento do painel elétrico e a impressão do molde do medidor.

Tabela 6 – Fluxograma do Projeto



Fonte: Autor

Conforme descrito acima, foi dividido em três etapas este projeto. Inicialmente, foi definido qual sensor iria realizar as medições propostas e a partir disto, desenvolveu-se a programação em duas partes: Sender (Transmissor) e Receiver (Receptor) para o envio e recepção dos dados à longa distância utilizando a comunicação via LORA. Por fim, foi programado o envio dos valores medidos para o registro e armazenamento dos dados para apresentar ao usuário final via interface gráfica do Blynk.

4.1 DEFINIÇÃO DO SENSOR DE MEDIÇÃO PZEM-004T

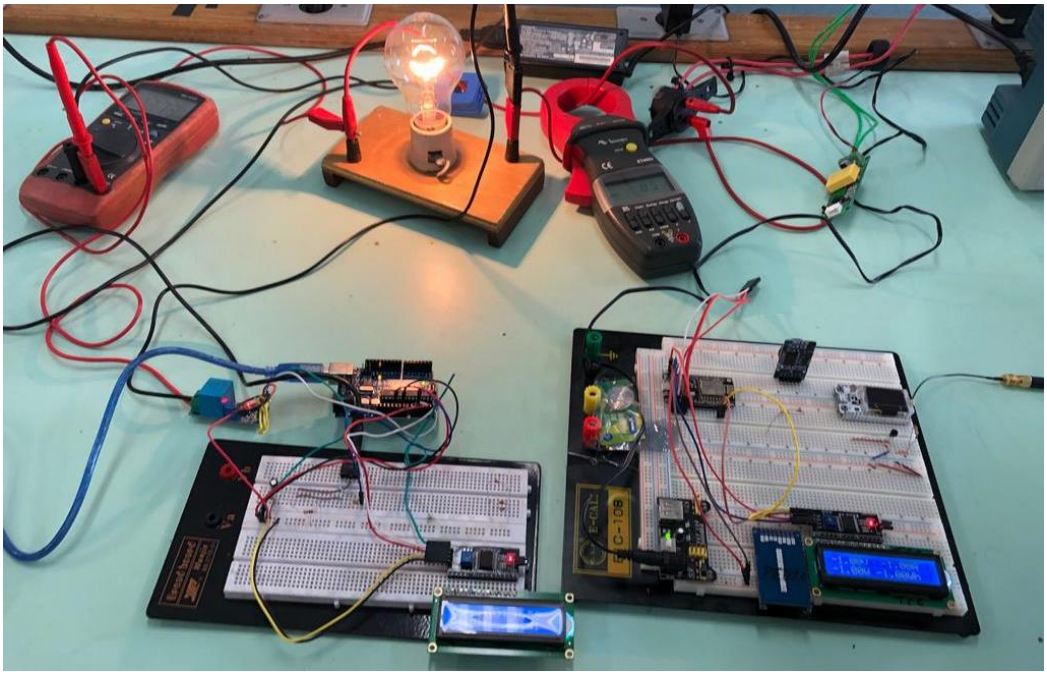
A definição do sensor é uma etapa crucial para o desenvolvimento do protótipo, pois será o núcleo do mesmo. Para isto, foram considerados como fatores de escolha: a relação de comunicação com o microcontrolador, acuracidade dos dados obtidos e a calibração do sensor. Todas as informações do PZEM estão descritas em seu Datasheet no ANEXO A.

4.1.1 Comparação entre o módulo PZEM e o sensor de tensão e corrente

Desta forma, inicialmente as primeiras análises foram com os sensores de corrente e de tensão. Contudo, os valores obtidos apresentavam uma flutuação muito alta e era necessária uma calibração específica para cada sensor. Com isto, aumentava a complexidade na detecção de possíveis problemas durante a medição da rede elétrica.

A partir disto, foi possível encontrar o sensor PZEM-004T, um módulo que realiza as medições de tensão, corrente, e calcula os valores de potência ativa e energia consumida em watt-hora. Logo, o objetivo inicial era realizar uma comparação entre os resultados obtidos entre os dois tipos de sensores, modular e analógico, e averiguar qual seria o mais apropriado para este projeto, de acordo com a Figura 20

Figura 20 – Comparação entre o sensor de corrente e tensão e o sensor PZEM-004T



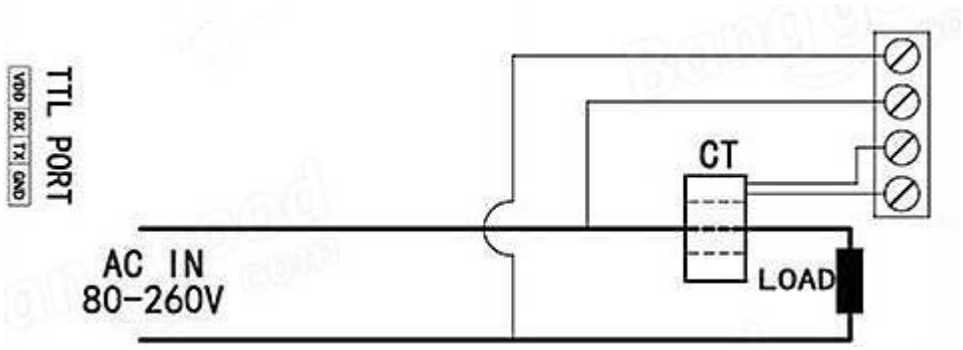
Fonte: Autor

4.1.2 Testes com o PZEM-004T

Antes de realizar a comparação, foi preciso entender o funcionamento do sensor fabricado pela empresa *Peacefair*. Contudo, as informações técnicas de sua construção interna e a configuração de hardware desse sensor não estão disponíveis na internet, que de acordo com o manual técnico, este módulo possui medições adicionais, tais como fator de potência e limite de consumo de energia, os quais não estão disponíveis, aparentemente bloqueados na programação.

Foi utilizada como fundamentação inicial, a programação em código aberto, *open-source*, de outros usuários, para a comunicação entre o módulo do ESP32 e o PZEM. Havia poucas informações sobre a utilização desse sensor com esse microcontrolador em específico. Desta forma, foi realizado diferentes testes para descobrir a configuração de hardware correta para obter tal comunicação. A ligação do PZEM, conforme descreve no seu manual, foi realizada conforme a Figura 21

Figura 21 – Ligação física do PZEM



Fonte: (INNOVATORS, 2019)

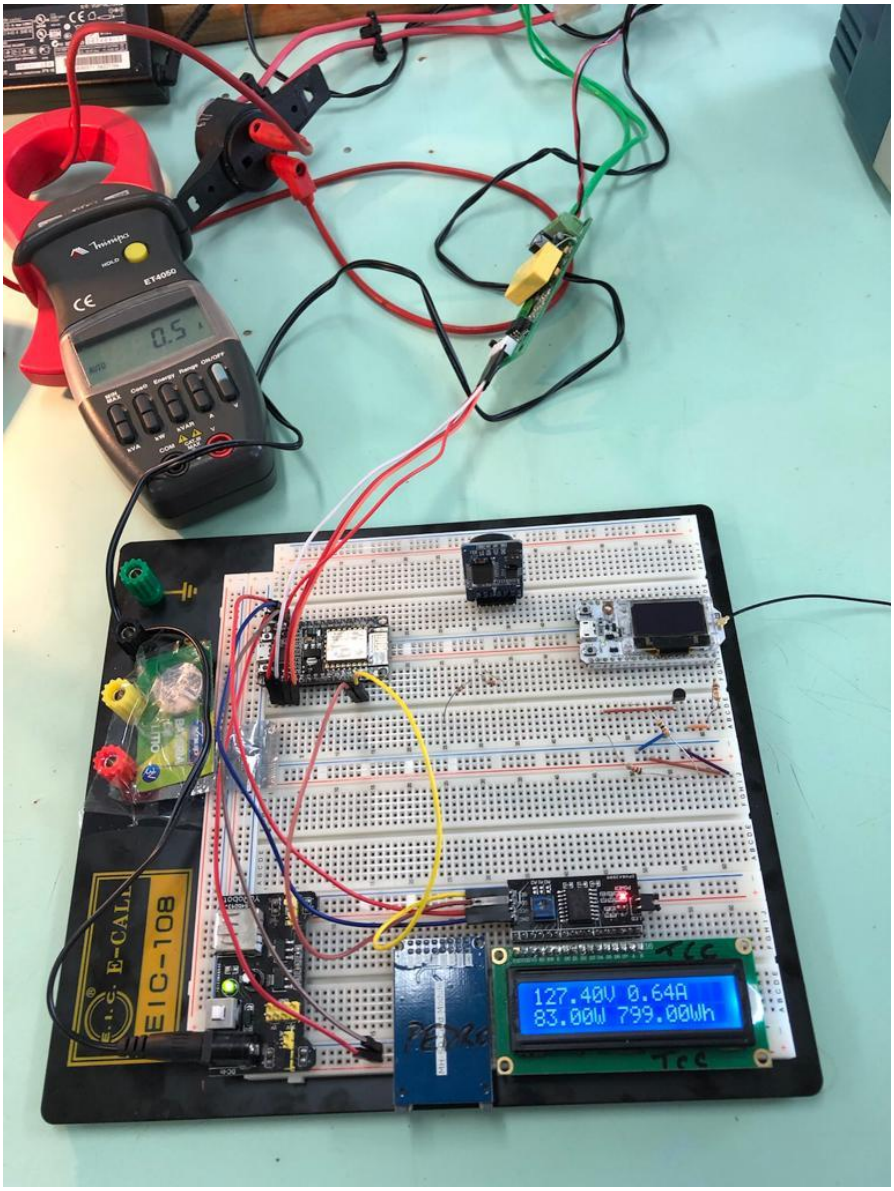
Este módulo possui dois sensores: um sensor de tensão, colocado em paralelo a carga, e o sensor de corrente. O sensor de corrente trata-se de um transformador de corrente, indicado na Figura 21 por CT. Os pinos de comunicação com o microcontrolador e a conexão com a carga situam-se em lados opostos da placa de eletrônica, com isolação por opto-acopladores.

Outro ponto importante foi em relação ao correto posicionamento dos fios fase e neutro, para que não sejam conectados aos terminais KRE na sequência errada. Portanto, deve se seguir exatamente a ordem escrita na placa.

Um ponto de atenção foi em relação ao manuseio do módulo, pois todos os locais com solda estão disponíveis ao contato físico, logo, a fim de evitar curtos-circuitos e choque elétrico,

foi inserida uma fita isolante na parte debaixo do PZEM para que não houvesse nenhuma complicação durante os testes. Foi confeccionado uma tomada com seus conectores ligados diretamente no PZEM, assim como disponíveis para inserir a carga, normalmente foi utilizado uma lâmpada ou um carregador de notebook para fins práticos, de acordo com a Figura 22. Utilizava-se um multímetro para aferição e comparação dos resultados medidos. Os resultados obtidos estavam dentro do limite aceitável, assim, foi definido a utilização do PZEM como o sensor deste projeto.

Figura 22 - Testes com o PZEM-004T e aferição dos dados com um multímetro



Fonte: Autor

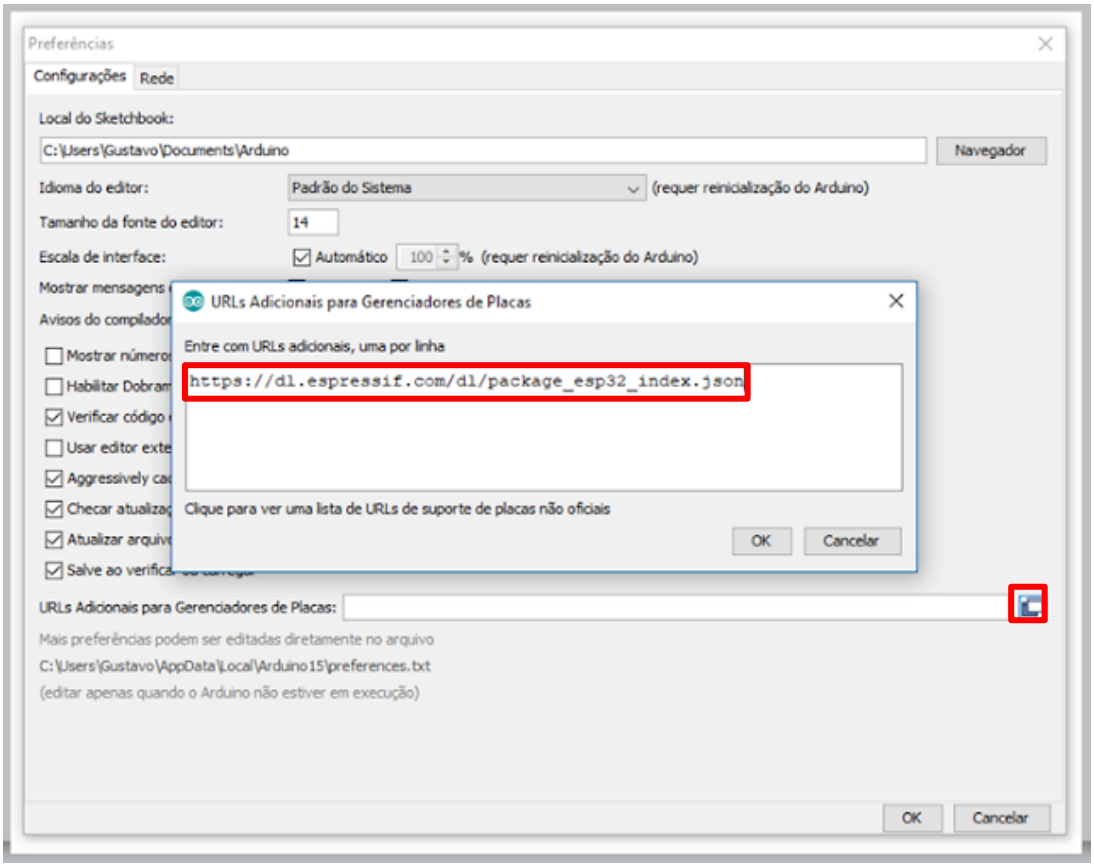
4.2 PROGRAMAÇÃO DO PROJETO

Para a programação do projeto foi realizado inicialmente toda a preparação da plataforma, assim como o download das bibliotecas a serem utilizadas pelos módulos. A partir disso, foi realizado um estudo para cada etapa do projeto, dividindo-os principalmente entre o emissor (*Sender*) e o receptor de dados (*Receiver*), a programação completa foi descrita no APÊNDICE A E APÊNDICE B

4.2.1 Preparação da plataforma do Arduino para a programação do ESP32

Para facilitar o desenvolvimento do projeto e devido a diferentes exemplos disponíveis na internet, foi utilizada a plataforma do Arduino IDE para a escrita de toda a programação aplicada ao medidor. A linguagem de programação utilizada no Arduino é a linguagem C++, com pequenas modificações, assim para programar o ESP-32, é necessária a configuração e instalação de algumas bibliotecas da empresa *Espressif*, conforme a sequência na Figura 23.

Figura 23 – Configuração da plataforma do Arduino IDE



Fonte: (FERNANDO K, 2019)

4.2.2 Descrição da programação do PZEM

Com a plataforma Arduino devidamente instalada e configurada, a partir de códigos *open-source* na internet, foi encontrada uma biblioteca que permite acesso e leitura do PZEM, cujos comandos principais serão abordados a seguir, a Figura 24 demonstra um exemplo de programação da biblioteca do PZEM.

Figura 24 – Setup do PZEM

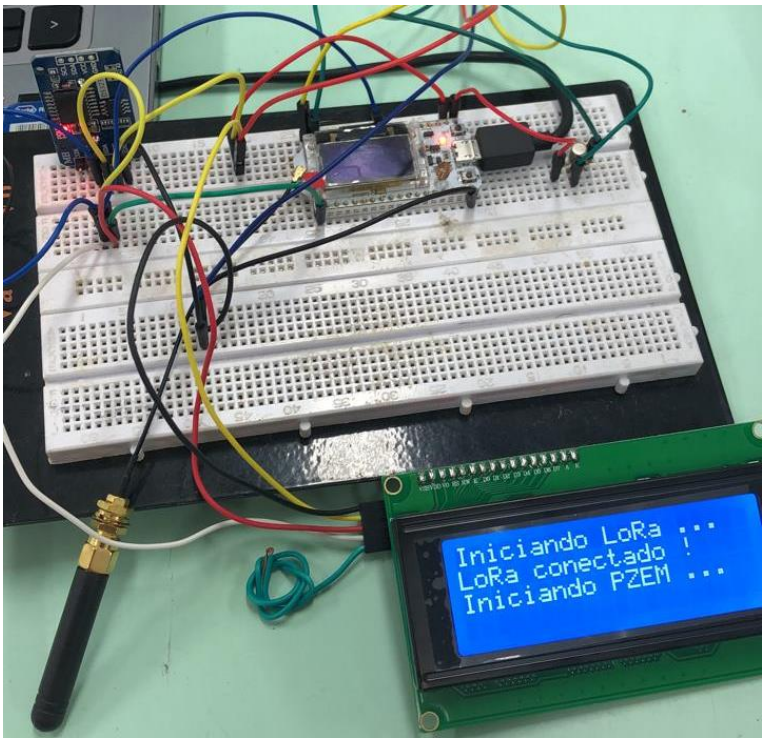
```
#include<HardwareSerial.h>
#include<PZEM004T.h>

/*
   An example on how to use ESP32 hardware serial with PZEM004T
*/

HardwareSerial PzemSerial2(2);     // Use hwserial UART2 at pins IO-16 (RX2) and
IO-17 (TX2)
PZEM004Tpzem(&PzemSerial2);
IPAddressip(192,168,1,1);

void setup() {
  Serial.begin(115200);
  while (true) {
    Serial.println("Connecting to PZEM...");
    if(pzem.setAddress(ip))
      break;
    delay(1000);
  }
}
```

Fonte: Autor

A biblioteca <*HardwareSerial.h*> é utilizada para a configuração das conexões físicas do ESP32 com o PZEM. A biblioteca <*PZEM004T.h*> permite a manipulação dos dados do PZEM com linhas de código específicas previamente definidas. Inicialmente, houve certos desafios quanto à comunicação com o PZEM, pois apenas algumas portas permitiam a comunicação UART com transmissor (TX) e receptor (RX).

A fim de inicializar a comunicação com o sensor, utiliza-se o comando "*IPAddress ip(192,168,1,1)*", que posteriormente, entra em um loop aguardando a conexão ser bem-sucedida. Nesta etapa, houve alguns problemas durante a inicialização do sensor, e inicialmente, acreditava-se que teria relação com a carga utilizada ou com a alimentação do sensor. A Figura 25 exemplifica o erro apresenta, o qual fica em um loop infinito.

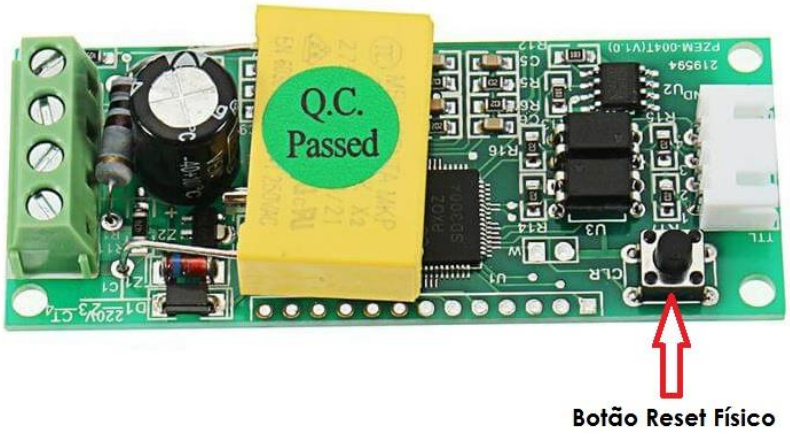
Figura 25 – Problema durante a inicialização do PZEM



Fonte: Autor

Para a resolução desse problema, foram utilizadas diferentes fontes de pesquisa e testes com o sensor, e a partir disso, foi possível concluir que seria necessário o reset físico no PZEM, apresentando na Figura 26, bem como, retirar qualquer alimentação do sistema e reconectá-lo posteriormente. Assim, o código força a conexão até sua concretização, pois sem a conexão no PZEM, o sistema não retorna qualquer informação de erro, apenas não sai do estado atual, criando um loop interminável.

Figura 26 – Identificação do botão de reset físico do PZEM



Fonte: Autor – “Adaptado de (INNOVATORS, 2019)“.

Vale ressaltar que, o módulo PZEM apenas consegue inicializar a comunicação com o microcontrolador se houver tensão em sua entrada, com um valor mínimo de 80 V. Após a conexão for bem sucedida, os valores de tensão podem variar abaixo de 80 V sem que perca esta conexão, logo, não será necessário pressionar novamente o botão de reset físico.

4.2.3 Aquisição e tratamento dos dados

Para a aquisição dos dados de forma segura, foi desenvolvido um sistema que armazena as informações até que todas as medições sejam realizadas, pois conforme analisado anteriormente, é necessário aguardar um intervalo após completar cada ciclo de leitura, para dar condições ao ESP de ler os dados corretamente. Desta forma, o sistema aguarda a aquisição de 3 medições, direcionando-os ao sistema de redundância de dados para filtragem de valores espúrios, na programação é utilizado o *while* para o loop de aquisição dos dados, de acordo com a Figura 27.

Figura 27 - Loop de aquisição dos dados do PZEM

```
// ++++++++++++++++++++++ Loop de aquisição dos dados do PZEM
++++++++++++++++++++++

// Aquisição dos dados a serem tratados - 3 resultados para
que sejam filtrados posteriormente

// ------------- Tensão ----------------------------------
  ts1=millis(); // Início da contagem do loop
  v1 = pzem.voltage(ip)+ v_offset; // Aquisição da medição de
tensão
  delay(500);
  while (v1 <= 0.0){ // Loop condicional enquanto não for
detectado nenhuma tensão
    delay(500);
    v1 = pzem.voltage(ip) + v_offset; // V_offset = Utilizado
porque o PZEM devolve um valor igual a -1 no resultado de
tensão igual a 0
    ts=millis(); // Contagem do final do loop
    if(ts-ts1 > 10000){ // Caso o loop ultrapassar 10
segundos, será considerado como falta de energia de fase
      faltadeenergia();
    }
  }
```

Fonte: Autor

Com a comunicação estabelecida com o sensor, as medições são iniciadas de forma automática. Caso a alimentação for menor que 80 V, os valores das variáveis assumem a saída igual a “-1”, onde indica que falta alimentação na entrada do sensor, neste caso, na rede elétrica. Além disso, foi possível identificar que mesmo com o sistema alimentado, algumas vezes o módulo realiza a leitura de dados espúrios, devido à flutuação natural da rede e o instante em que foi monitorada a medição.

No entanto, ao retornar a comunicação, o valor da energia consumida não é perdido, pois o módulo possui uma memória EEPROM que armazena o último valor lido durante o seu funcionamento. Caso seja necessário zerar o valor da energia consumida, é necessário pressionar o botão RESET durante 5 segundos, com o módulo devidamente alimentado, e posteriormente, pressionar o botão mais uma vez.

Além disso, foi identificado que o valor da tensão medida pelo PZEM sempre estava aproximadamente 1 Volt abaixo da tensão medida pelo voltímetro digital. A partir disso, foi possível identificar que existe uma queda de tensão do próprio circuito devido a sua construção. Com isso, foi inserido um pequeno valor de tensão de offset na programação para obter valores mais próximos do real, conforme apresentando na Figura 27.

Por fim, após a aquisição dos valores de cada variável, é realizado um tratamento dos dados para assegurar que o valor medido naquele instante seja o mais próximo da realidade. Assim, o sistema armazena 3 medições em sequência e realiza a comparação entre estes valores. Durante os testes práticos, os resultados apresentaram uma boa confiabilidade nas medições, e desta forma, foi definido o maior resultado como a medição naquele instante, evitando a presença de resultados menores devido a oscilações da rede.

Esclarecendo a situação acima, por exemplo, caso forem adquiridos os seguintes valores de tensão: 127,1 V / 128,0 V / 126,2 V, todos os valores em Volts, o resultado após a passagem no tratamento de dados será igual a 128 Volts. Foi estipulada esta configuração, pois normalmente os valores espúrios estavam mais relacionas com o problema de medição, onde retornava um resultado equivalente a “-1”. Logo, para resolver esse problema, foi desenvolvida essa comparação dos valores lidos. Para a programação, foi escrito três enlaces relacionando os dados da entrada do loop, e obtendo o maior valor em sua saída, de acordo com a Figura 28.

Figura 28 – Leitura e tratamento dos valores medidos

```
// Leitura e Tratamento dos valores de Corrente
void *espurgoi(float x, float y, float z){
  float maior = x; float menor = x;
  if (y > menor){
    menor = y;
  }
  else {
    menor = 0.0;
  }
  if (y > maior && y >= 0.0){
    maior = y;
  }
  if (z > menor){
    menor = z;
  }
  else {
    menor = 0.0;
  }
  if (z > maior && z >= 0.0){
    maior = z;
  }

  i = maior;
}
```

Fonte: Autor

4.2.4 Detecção de falta de energia na rede elétrica

Antes de ser feita a descrição da detecção de falta de energia, foi realizado um estudo para entender quais são os problemas mais comuns na rede elétrica nas casas residenciais. De acordo com as informações obtidas, existem diferentes tipos de problemas que podem surgir relacionados com uma falha do sistema de energia elétrica.

Inicialmente, a falha de energia pode ser causada por uma série de eventos, tais como: relâmpagos, linhas de energia derrubadas, super-demanda da rede elétrica, acidentes durante manutenções no poste de luz, apagões e catástrofes naturais, e os seus resultados podem ser diversos.

Além disto, tem a questão de oscilações de tensão na rede e subtensões, normalmente causadas por exigências de energia na inicialização de equipamentos elétricos, como máquinas, elevadores, motores, compressores, ar condicionado, etc. Estes equipamentos, ao serem ligados, consomem grande quantidade de energia, provocando a queda de tensão por curtos espaços de tempo.

Como não é possível abordar todo tipo de falha elétrica, foi estipulado para o projeto um tempo máximo de espera para a aquisição de valores medidos acima de 0 Volts. Pois, se por algum motivo, a rede elétrica não fornecer mais energia, o PZEM irá começar a medir os valores e apresentará como resultado o valor igual a "-1", este será descartado no loop de aquisição de dados, o qual ficará aguardando apenas os valores acima de 0 Volt.

Com os testes realizados, foi possível identificar que para realizar uma medição da tensão da rede elétrica, o sensor demora cerca de 5 a 9 segundos, desta forma, definiu-se como premissa básica o tempo de medição máximo até 1 segundo. Caso o módulo não realizar nenhuma leitura acima de 0 Volts dentro de 1 segundo a função de falta de energia será acionada, a programação deste sistema é descrita na Figura 29.

Figura 29 – Detecção de falha de energia na rede elétrica

```
// Tratamento de dados para execução do ciclo de detecção de falta de energia de fase
void *faltadeenergia(){
  LoRa.beginPacket();
  LoRa.print("Falta de energia detectada ! ");
  LoRa.endPacket();

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Falta de energia ");
  lcd.setCursor(4,1);
  lcd.print("detectada ! ");

  Serial.println("Falta de energia detectada ! ");
  delay(2000);
  a=0;
  v1 = pzem.voltage(ip)+ v_offset;
  delay(500);
  while (v1 <= 0.0){ // Loop condicional enquanto não retorna a energia ao sistema
    v1 = pzem.voltage(ip)+ v_offset;
    delay(500);
    a++;
    Serial.println("Tentativa de reconexão:  "); Serial.print(a);
    lcd.setCursor(0,2);
    lcd.print("Reconexao.... ");
    lcd.setCursor(0,3);
    lcd.print(a);
  }
}
```

Fonte: Autor

Contudo, como todo o sistema depende de uma alimentação, e esta é feita pela rede elétrica, caso a mesma apresente alguma falha, na prática, o microcontrolador também iria desligar e não apresentaria nenhuma informação. Por isso, foi utilizado um mini Nobreak para o ESP, pois se caso ocorrer à falha de energia, este sistema irá realizar a alimentação do microcontrolador durante o problema na rede elétrica. Assim, é colocada a tensão da rede elétrica na entrada do Nobreak para manter a sua bateria devidamente carregada, e a saída do mesmo conectado ao ESP via cabo de celular micro USB. Em uma falha de energia, a bateria irá manter o microcontrolador ligado e as luzes de indicação do Nobreak irão salientar tal informação, com os detalhes de sua arquitetura na Figura 30.

Figura 30 - Nobreak para o ESP-32

Fonte: Autor

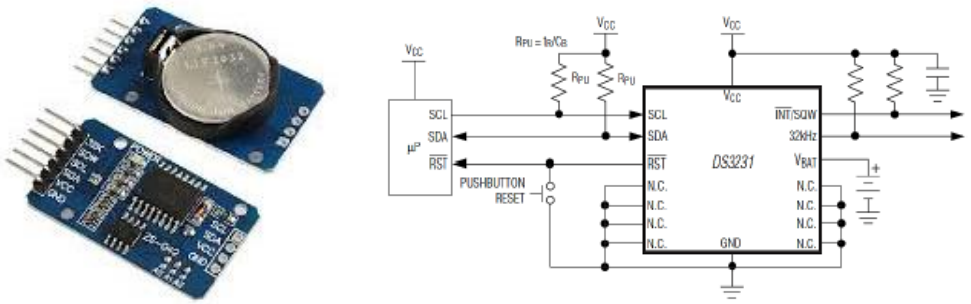
Além do mais, se o sistema entrar na função de falta de energia, será apresentado display do medidor, a informação sobre a possível falha identificada. Em paralelo a isso, o programa tenta a reconexão com o PZEM para retornar as medições previamente realizadas. Para sair deste loop, é necessário que retorne a alimentação da rede ou realizar o reset na programação.

4.2.5 Tratamento do consumo e custo mensal

A fim de obter um sistema interativo e que seja o mais prático para o usuário final, foi desenvolvido as informações de consumo mensal e custo da energia elétrica mensal, pois o objetivo principal é realizar um monitoramento do consumo de energia residencial e desta forma o usuário consegue acompanhar o seu consumo, e consequentemente o gasto diário.

Para que isso seja possível, foi utilizado um módulo RTC, *Real Time Clock*, apresentando na Figura 31, para armazenar a data e hora dentro do projeto. Este módulo possui uma bateria recarregável e uma memória EEPROM, útil para armazenar as informações do relógio, com isso, mesmo que o sistema não tenha alimentação, ainda é realizada a devida contagem de data e hora.

Figura 31 – Real Time Clock – RTC DS3231



Fonte: (MAXIM, 2019)

Desta forma, foi definido que todo dia 01 do mês às 00 horas 00 minutos será redefinido o valor do consumo inicial do usuário. Após essa configuração, o valor de energia consumida é subtraído do consumo inicial do mês, assim é obtido o valor do consumo mensal, de acordo com a programação abaixo apresentado na Figura 32.

Figura 32 – Descrição do cálculo do consumo e custo mensal

```
 DateTime now = rtc.now();
 if(((now.day(), DEC)==1)&& ((now.minute(), DEC)==00) && ((now.hour(), DEC)==00) && (e > 0) ){ // Reajuste do consumo de energia para todo dia 1 às 00:00
   cons_aux=e;
 }

// ---------------------------------- Cálculo do consumo mensal de energia ----------------------------------
  //cons_aux=e; // Condição inicial utilizada apenas para definição do consumo mensal inicial
  // ** DEVE SER RETIRADO APÓS A PRIMEIRA ITERAÇÃO

  if (e < cons_aux){ // Caso a constante de consumo auxiliar for maior que o consumo atual
    consumo_mensal=cons_aux;
    Serial.print("Problemas com o consumo mensal");
  }
  else{
    consumo_mensal=e-cons_aux;  // Subtração do consumo base do mês com o consumo decorrido do mês
  }

// Cálculo do custo mensal
  if(consumo_mensal >= 0){
    custo_mensal=0.65823*consumo_mensal; // Multiplicação R$ 0,65823 por kWh (Dados obtidos em SP: 2019)
  }
  else{
    Serial.print("Problemas com o custo mensal");
  }
```

Fonte: Autor

Com o valor de consumo mensal previamente calculado, é possível realizar o cálculo do custo mensal do consumo residencial. Para esse projeto, baseado nos dados do custo da tarifa de baixa tensão de energia elétrica na cidade de São Paulo em outubro de 2019, foi definido o valor equivalente a R$ 0,65823 por kWh, de acordo com a Tabela 7.

Tabela 7 – Composição da tarifa de energia elétrica do estado de São Paulo

TARIFAS DE BAIXA TENSÃO - R$/kWh - Outubro/2019					
Classe de consumo	Tarifa com PIS/COFINS e ICMS				
	Faixa consumo				
	até 50 kWh	de 51 até 300 kWh	até 300 kWh	de 301 até 450 kWh	acima de 450 kWh
	Residencial	Residencial	Demais Classes	Todas as Classes	Todas as Classes
	(isento de ICMS)	(ICMS de 18%)	(ICMS de 20%)	(ICMS de 31%)	(ICMS de 30%)
Residencial	0,65823	0,81201	-	0,97681	0,96180

Fonte: (LIGHT, 2019)

4.3 CONEXÕES FÍSICAS DO SENDER

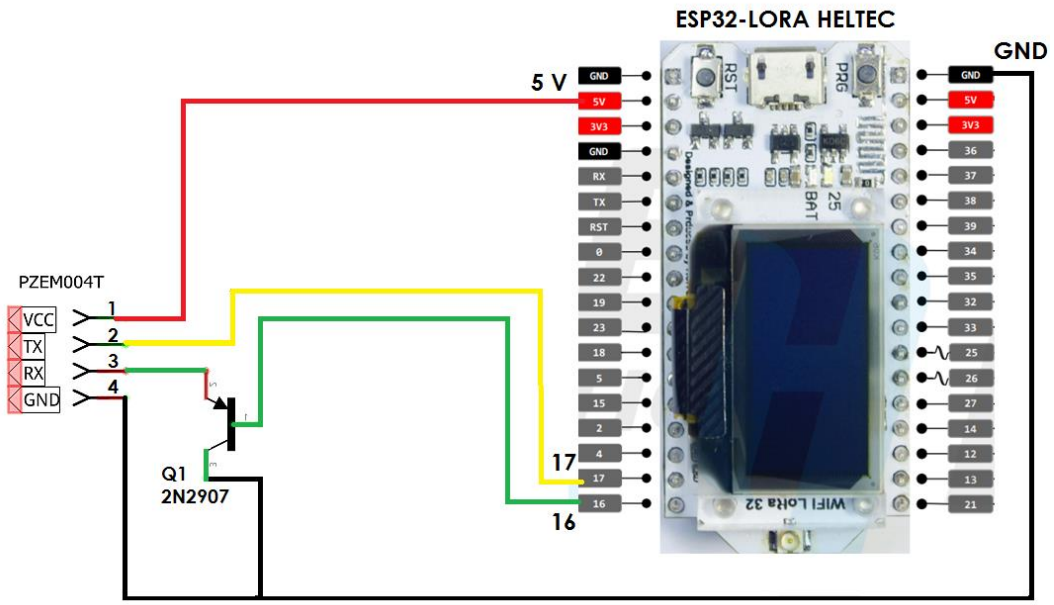
Nesta etapa foram realizados diferentes testes práticos com as conexões do ESP-32 com o PZEM, juntamente com os módulos de comunicação via I2C para apresentar data e hora e diminuir a quantidade de entradas e saídas do sistema.

4.3.1 Conexões do PZEM-004T

O módulo PZEM é alimentado com +5 Vcc, esta alimentação foi realizada diretamente através do ESP32. De acordo com a configuração da comunicação serial definida na biblioteca, apenas algumas entradas do ESP-32 funcionam para conexão dos transmissores (Rx e Tx) do PZEM. Para identifica-las foi realizado um escaneamento nas portas seriais até que houvesse um retorno de comunicação nível TTL. Outras portas RX e TX que estão disponíveis no ESP são utilizadas para a comunicação do LORA ou pré-definidas para outras aplicações. Logo, foram definidas as portas I0-16 e I0-17 para a transmissão RX e TX, respectivamente.

Conforme a fundamentação de outros projetos, para evitar problemas com a inicialização do módulo, é necessário inserir um transistor PNP com o pino RX (I0-16), o qual garante que a porta esteja em nível lógico baixo durante a reinicialização do ESP. Caso o nível lógico estiver alto, o microcontrolador não entra em modo de gravação do programa, *modo firmware*, ou seja, não compila o *sketch* desenvolvido no Arduino. Se apresentar esse erro, será necessário o reset físico, procedimento já mencionado anteriormente. O diagrama com as conexões é apresentado na Figura 33.

Figura 33 - Conexão do PZEM com o ESP utilizando o transistor



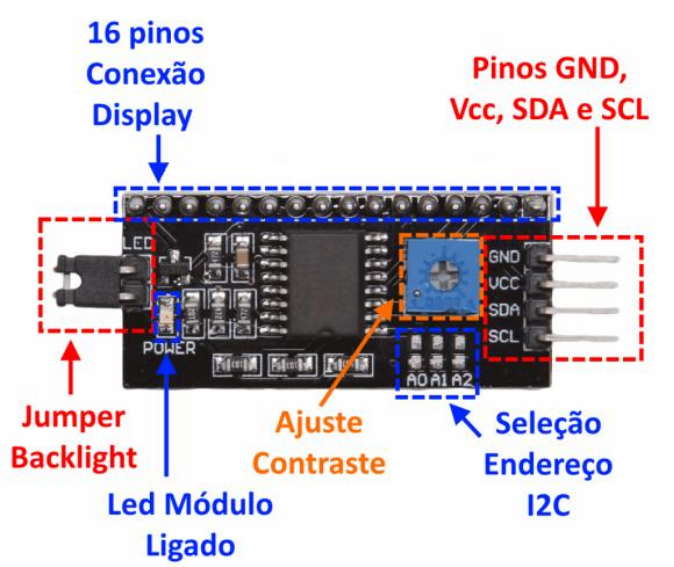
Fonte: Autor

4.3.2 Comunicação I2C com o RTC e Display

Com o objetivo de diminuir a quantidade de entradas e saídas no sistema para utilizar os periféricos, foi utilizado à comunicação I2C. Este barramento facilita a integração de circuitos com diferentes aplicações utilizando dois sinais para a transmissão, *Serial Data* (DAS) e *Serial Clock* (SCL), especificados na Figura 34.

Neste projeto, para a conexão do display LCD e para o registro de data e hora, os módulos foram conectados nas portas mencionadas acima. Não foi necessário nenhum tipo de endereçamento, apenas realizar a programação com o setup de cada módulo.

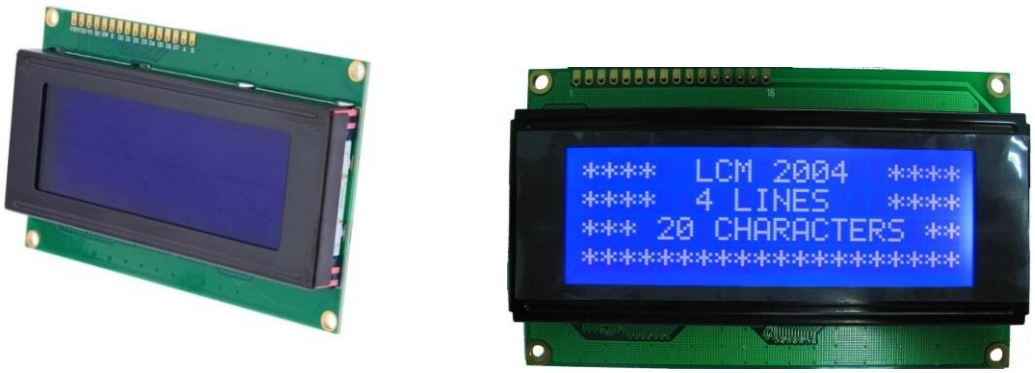
Figura 34 – Descrição do módulo I2C



Fonte: (CIA, 2019)

O módulo I2C para o display diminui consideravelmente a quantidade de conexões com o ESP-32. Além disso, ao utilizar um conector, é possível ligar diretamente ao display sem a necessidade de qualquer tipo de solda. O display utilizado neste projeto foi o 20x4, *backlight* azul, 20 caracteres e 4 linhas, conforme a Figura 35, logo facilita a apresentação dos resultados medidos de forma clara, caso o cliente não deseje realizar a abertura do aplicativo.

Figura 35 – Display 20x4 utilizado no projeto



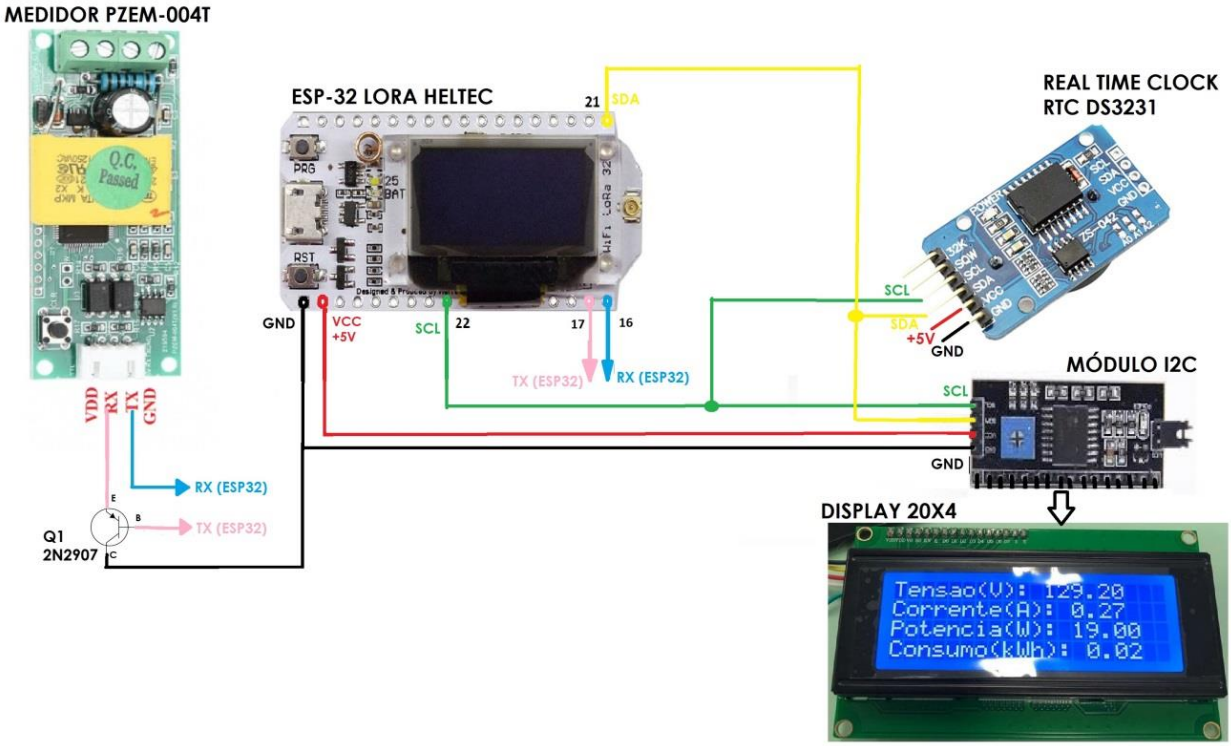
Fonte: (CIA, 2019)

4.3.3 Integração dos módulos no circuito Sender

Para realizar a integração de todos os sistemas mencionados acima, inicialmente para a etapa de testes, foram realizadas as conexões utilizando um *protoboard* e fios de conexão, utilizando o diagrama de fiação conforme a Figura 36, onde os valores medidos são apresentados no display 20x4.

Após os testes apresentarem bons resultados, foi desenvolvida uma placa em circuito impresso, com todos os detalhes no APÊNDICE C e D desenvolvida na plataforma *Eagle*.

Figura 36 – Diagrama de fiação do circuito do Sender



Fonte: Autor

4.3.4 Desenvolvimento da placa em circuito impresso

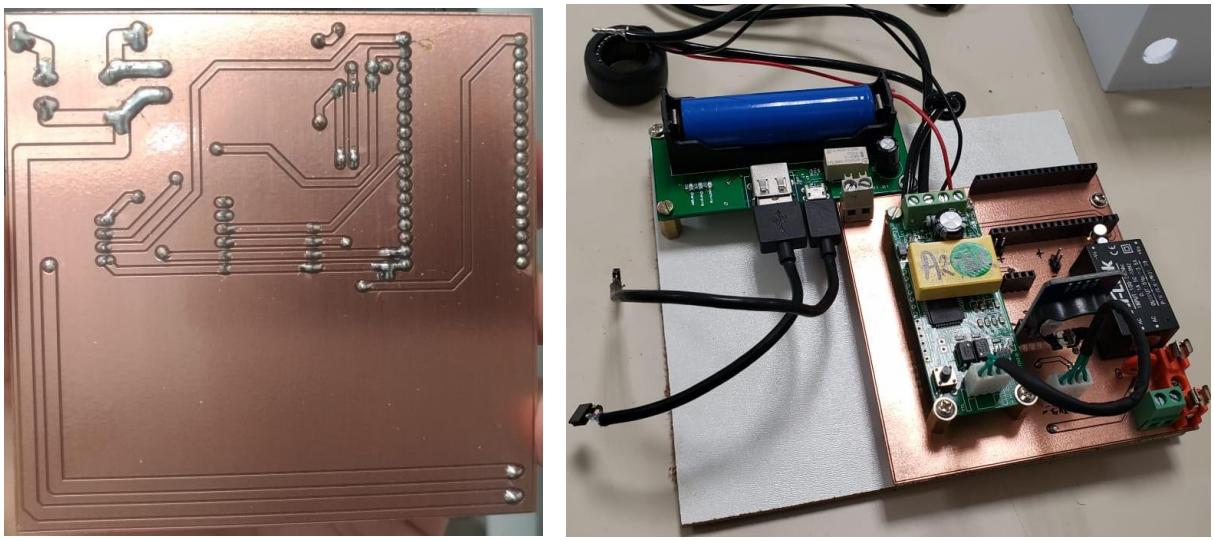
A montagem do circuito consiste basicamente em um aglomerado de módulos. Aos quais foi decidido juntar através de conectores fêmeas para facilitar a troca do componente caso houvesse alguma falha após a confecção completa do projeto. A montagem do circuito se tornou mais prática, tendo em vista que havia um protótipo no protoboard antes da placa final.

Para a placa do PZEM-004T foi alterado seu conector de fábrica para um conector MOLEX vertical 2.54mm a fim de melhorar não só a aparência, como também a conexão entre a placa principal e o PZEM, conforme a Figura 37.

Inseriu-se também uma fonte compacta *Hi-Link,* para fornecer ao nobreak a alimentação para a placa principal, em que no caso de ausência de energia elétrica, o circuito envia o sinal de falta de energia elétrica para o receptor. Além do mais, foi cogitado usar o conector para baterias externas que fica embaixo da entrada micro USB do ESP32 LoRa, porém o socket de 2 pinos *raster* 1.25 mm precisava ser importado e então foi encontrado a solução do nobreak no mercado nacional a um preço acessível, conforme dito anteriormente, o mesmo precisa de uma bateria lítio do tipo 18650 com uma tensão de 3.7V obtida de uma luz de emergência.

Uma das grandes dificuldades para o desenvolvimento desta placa foi alocar os espaços entre os pads e das trilhas para facilitar a soldagem, assim como colocar uma distância razoável, para separar as regiões de controle e potência com o intuito de garantir nenhuma interferência, obtendo assim, a eficiência na transmissão de dados esperada.

Figura 37 – Confecção da placa de circuito impresso



Fonte: Autor

4.4 ENVIO DOS DADOS PELA COMUNICAÇÃO LORA

Com o objetivo de realizar o envio dos dados de forma correta e precisa, foi necessária a manipulação de bibliotecas com linhas de códigos específicos para a criação do pacote e transporte do mesmo para o *Receiver*.

4.4.1 Preparações para utilização da comunicação LoRa

Para utilizar as respectivas funções do chip LoRa disponíveis através do Arduino IDE é necessário primeiramente instalar a biblioteca LoRa por *Sandeep Mistry* através da função de busca na IDE (Sketch - Incluir biblioteca - Gerenciar bibliotecas) e após isso, declarar a biblioteca específica para essa utilização juntamente com a <*SPI.h*> que permite a comunicação serial do LoRa, o setup mencionado, assim como, a declaração dos pinos estão descritos na Figura 38.

Figura 38- Setup do LORA no Sender

```
#include <SPI.h> //responsável pela comunicação serial
#include <LoRa.h> //responsável pela comunicação com o Lora

// Definição dos pinos utilizados pelo LoRa
  #define SCK     5    // GPIO5  -- SX127x's SCK
  #define MISO    19   // GPIO19 -- SX127x's MISO
  #define MOSI    27   // GPIO27 -- SX127x's MOSI
  #define SS      18   // GPIO18 -- SX127x's CS
  #define RST     14   // GPIO14 -- SX127x's RESET
  #define DI00    26   // GPIO26 -- SX127x's IRQ(Interrupt Request)

// Frequência do radio - podemos utilizar ainda : 433E6, 868E6, 915E6
   #define BAND    915E6
   #define PABOOST true

// Parâmetros: address,SDA,SCL
   SSD1306 display(0x3c, 4, 15); //construtor do objeto que controlaremos o display
   String rssi = "RSSI --";
   String packSize = "--";
   String packet ;
```

Fonte: Autor

Na variável BAND foi definida a frequência de transmissão e recepção de dados utilizada. Como a antena que veio em conjunto com o ESP trabalha com 915 MHz, foi aplicada a mesma frequência na programação. Além disso, é possível definir um ganho no envio declarando a variável *PABOOST* como *true*.

4.4.2 Inicialização e envio dos dados

Após o preparo dos componentes iniciais descrito no tópico anterior, é feito a inicialização da comunicação da placa com o componente LoRa com o comando *SPI.begin* com as informações de entrada/saída do ESP-32.

Para a construção e transmissão do pacote, são necessários 3 comandos:

LoRa.beginPacket() utilizado para indicar que um novo pacote de dados foi aberto e está disponível para receber informações.

LoRa.print() que escreve o que estiver entre parênteses dentro do pacote aberto anteriormente. Nesse passo é possível escrever quantas vezes for desejada, levando em conta que existe um limite de memória igual a 50 KB para cada pacote.

E por fim, *LoRa.endPacket* que é usado para indicar o fechamento do pacote aberto e disparar o envio para o LoRa configurado para ser o receptor (Receiver). Os comandos são listados na Figura 39.

Figura 39 – Preparo e envio dos dados via LoRa

```
SPI.begin(SCK,MISO,MOSI,SS); // Inicia a comunicação serial com o Lora
LoRa.setPins(SS,RST,DI00); // Configura os pinos que serão utlizados pela biblioteca (deve ser chamado antes do LoRa.begin)

// ----------------------------------------  Envio dos dados através do LoRa
 //beginPacket : abre um pacote para adicionarmos os dados para envio
 //print: adiciona os dados no pacote
 //endPacket : fecha o pacote e envia -> retorno= 1:sucesso | 0: falha

  LoRa.beginPacket();
  LoRa.print("Tensão (V): ");
  LoRa.print(v);       //Tensão
  LoRa.endPacket();
  Serial.println("Tensao enviada");
  delay(2000);
```

Fonte: Autor

4.5 DESENVOLVIMENTO DO RECEIVER

A programação do receptor é dividida em basicamente 2 frentes: recepção dos dados transmitidos pelo Sender através do LoRa e a transmissão para o aplicativo Blynk via WiFi.

4.5.1 Recepção e preparação do LoRa

Para preparação dos dados a serem recebidos, foi utilizado os mesmos parâmetros que na programação do Sender, com a adição do comando *LoRa.receive()* para colocar o dispositivo no modo de recepção.

Além disso, declararam-se algumas variáveis que receberão os valores dos pacotes recebidos, de acordo com o seu o tamanho e força do sinal (RSSI), conforme a Figura 40 apresenta.

Figura 40 – Preparação dos dados no Receiver

```
                                                    String rssi = "RSSI --";
//LoRa.onReceive(cbk);                              String packSize = "--";
  LoRa.receive(); // Habilita o Lora para receber dados   String packet ;
```

Fonte: Autor

Dentro do loop de aquisição dos dados, declarou-se a variável *packetSize,* a qual é responsável pela verificação do tamanho dos pacotes recebidos pelo Sender. Caso essa variável for maior que 0, significa a chegada de um novo pacote e é possível chamar uma sub-rotina para o tratamento destes dados, descrito na Figura 41.

Figura 41 – Verificação da chegada de novos pacotes

```
//parsePacket: checa se um pacote foi recebido
//retorno: tamanho do pacote em bytes. Se retornar 0 (ZERO) nenhum pacote foi recebido
  int packetSize = LoRa.parsePacket();
//caso tenha recebido pacote chama a função para configurar os dados que serão mostrados em tela
  if (packetSize) {
    cbk(packetSize);
  }
```

Fonte: Autor

Na sub-rotina de tratamento dos dados, é iniciada uma variável do tipo *string*, denominada na programação como *packet*, a qual recebe um valor nulo e é construída caractere por caractere, até que o seu tamanho seja igual ao tamanho recebido na variável *packetSize*. Ademais, é realizado a medição da força do sinal recebido através do comando *LoRa.packetRssi(),* conforme os testes práticos e diferentes abordagens técnicas, definiu-se o valor de atenuação do sinal até -124 dB como um sistema capaz de ter um certo tipo de comunicação, assim determinando a distância de envio e recepção de dados do sistema, conforme apresentado na Figura 42.

Figura 42 – Recuperação e transformação do pacote recebido

```
//função responsável por recuperar o conteúdo do pacote recebido
//parametro: tamanho do pacote (bytes)
void cbk(int packetSize) {
  packet ="";
  packSize = String(packetSize,DEC); //transforma o tamanho do pacote em String para imprimirmos
  for (int i = 0; i < packetSize; i++) {
    packet += (char) LoRa.read(); //recupera o dado recebido e concatena na variável "packet"
  }
  rssi = "RSSI=  " + String(LoRa.packetRssi(), DEC)+ "dB"; //configura a String de Intensidade de Sinal (RSSI)
  //mostrar dados em tela
  loraData();
}
```

Fonte: Autor

4.5.2 Aquisição e envio de dados para o aplicativo Blynk

Assim como a biblioteca do LORA, a biblioteca do Blynk também deve ser instalada manualmente conforme o site da própria plataforma menciona (BLYNK, 2019).

Na programação, além da biblioteca do Blynk, também foi declarado à biblioteca do Wifi, que realiza a conexão com o aplicativo para o envio dos dados. Inicialmente, na programação é definido para cada parâmetro, uma nova variável relacionada com o aplicativo.

Como pré-requisito para a transmissão, é necessário indicar 3 variáveis para conexão do medidor com o aplicativo Blynk que disponibiliza a informação em um *dashboard* virtual, conceito melhor descrito no Capítulo 4.7 - Desenvolvimento do aplicativo Blynk.

A primeira é a variável "*auth*" do tipo *char* recebe o código enviado pelo próprio Blynk após a criação do dashboard no celular do usuário final. A segunda e terceira recebem as credenciais da rede WiFi que será utilizada para conexão a internet. São elas "*ssid*" e "*pass*" todas do tipo *char*, recebendo respectivamente o nome da rede e a senha.

Para conexão e inicialização do Blynk devem ser utilizados os comandos conforme a Figura 43. É importante salientar que, a cada interação que o ESP realizar com o aplicativo, é necessário executar o *Blynk.run()*.

Figura 43 – Autenticação das credencias do Blynk e Wifi local

```
  char auth[] = "ReTdIP_oDz5MDJ37-vQ3GLTrtRfV0DDM";

// Credenciais da Wifi Local
  char ssid[] = "XXXX";          //colocar o ssid da rede
  char pass[] = "XXXX";          //colocar a senha da sua rede

// Início do APP Blynk
  Blynk.begin(auth, ssid, pass); //Faz a conexão com o Blynk
// Inicialização do Blynk
  Blynk.run();
```

Fonte: Autor

A rotina de envio dos valores para o aplicativo é acionada após a leitura do pacote recebido, escrito dentro da variável "*packet*". Como todos os pacotes de dados serão escritos e sobrescritos dentro de uma única variável após seu recebimento no LoRa Receiver, é necessário uma verificação e tratamento para categorizar esse conteúdo como um dos parâmetros de medição de energia (tensão, corrente, etc.) e transmitir corretamente permitindo que o aplicativo encaixe o valor recebido em cada um dos respectivos gráficos.

Para um exemplo mais didático, levando em consideração o processo de envio do parâmetro (**Tensão)**. Após o envio através do LoRa Sender, o pacote com o conteúdo "Tensão(V):120.00" é recebido pelo LoRa receiver e escrito dentro da variável "*packet*".

No processo de tratamento é primeiramente analisado o valor da última posição do pacote, neste exemplo igual a 15 (cada caractere equivale a uma posição começando do zero). Após isso, é realizado um teste para todas as unidades de medidas utilizadas no projeto (V, A, W, kW, kW/h, R$ ou falta de energia) a fim de verificar em qual desses parâmetros o conteúdo recebido se encaixa.

A escrita é realizada a partir do valor da variável "*packet*" e entre as posições achadas após os dois pontos (:) até a posição final do pacote. Com isso, a respectiva variável, no exemplo "*voltage_blynk*", é transformada em um tipo *float* para que seja transmitida ao Blynk como um número, ao invés de texto. Após essa rotina de identificação e manipulação dos dados, descrita na Figura 44, é realizado o envio através do comando *Blynk.virtualWrite*.

Figura 44 – Rotina de identificação dos dados e envio ao Blynk

```
// Indica o tamanho total do pacote recebido/posição do final
int tamanho = packet.length();
// stringaAProcurar.indexOf(stringProcurada)
// Verifica qual é o parametro/número resultante indica a posição que a variável foi achada
// Se o retorno for -1 que dizer que não foi encontrado
int teste = packet.indexOf("(V)");
if (teste > -1)
    {
        // String procurada encontrada
// Procura 2 posições anteriores ao valor do parametro
//pois o valor sempre estará após um : e um espaço (2 posições)
        int doispontos = packet.indexOf(": ");
        String voltage_blynk = packet.substring(doispontos + 2, tamanho);
// Monta o valor que existe duas posições após o : até a posição ao final do pacote
        Serial.println("Tensão recebida");
        voltage_blynk.toFloat();
        Serial.println(voltage_blynk);
        Blynk.virtualWrite(V1, voltage_blynk); // Envia o valor para o Blynk
    }
```

Fonte: Autor

Uma exceção ocorre quando o pacote recebido é o de falha de energia na rede elétrica. Pois neste instante, são enviados comandos para zerar os parâmetros de tensão e corrente, assim como, é enviado um alerta para que o aplicativo notifique ao usuário pelo celular e a concessionária via e-mail sobre a possível falha na rede, conforme a Figura 45.

Figura 45 – Envio das informações de falha de energia ao Blynk

```
teste = packet.indexOf("Falta de energia");
if (teste > -1)
    {
        int doispontos = packet.indexOf(": ");
        String falta_blynk = packet.substring(doispontos + 2, tamanho);
        Serial.println("Falta de energia detectada");
        Blynk.virtualWrite(V9, 255); //Enviar 255 pra setar o LED em NL alto e 0 pra NL baixo
        Blynk.virtualWrite(V1, 0.0); // Atualizar o valor da tensão pra 0
        Blynk.virtualWrite(V2, 0.0); // Atualizar o valor da corrente pra 0
    }
```

Fonte: Autor

4.6 DESENVOLVIMENTO DO LORA

Para a contextualização do LoRa, foi necessário inicialmente estipular modelos e a frequência de utilização do projeto, além disso, foi possível observar o impacto que a escolha da antena pode trazer ao projeto. Por fim, foi realizado alguns testes para mensurar a distância máxima que os dados percorrem.

4.6.1 Definição frequência LoRa

O chip LoRa, criado pela empresa francesa *Semtech*, são fabricados em 4 modelos: SX1276/77/78/79 , cada um com uma respectiva frequência de trabalho.

Para este projeto, foi escolhido o modelo SX1276, que conforme o seu *datasheet* possui um alcance maior e uma maior robustez ao ruído, fornecendo maiores opções caso for encontrada alguma dificuldade em uma das frequências de utilização.

A *LoRa Alliance,* que é a responsável pela tecnologia *LoRaWan*, definiu as frequências para a transmissão conforme o país, os quais são mencionados na Tabela 8.

Tabela 8 – Faixa de frequência utilizada por cada país

País	**Faixa de frequência utilizada**
Estados Unidos	902-928 MHz
Brasil	902-928 MHz
China	779-787 e 470-510 MHz
Europa	863-870 e 433 MHz

Fonte: (FERNANDO K, 2019)

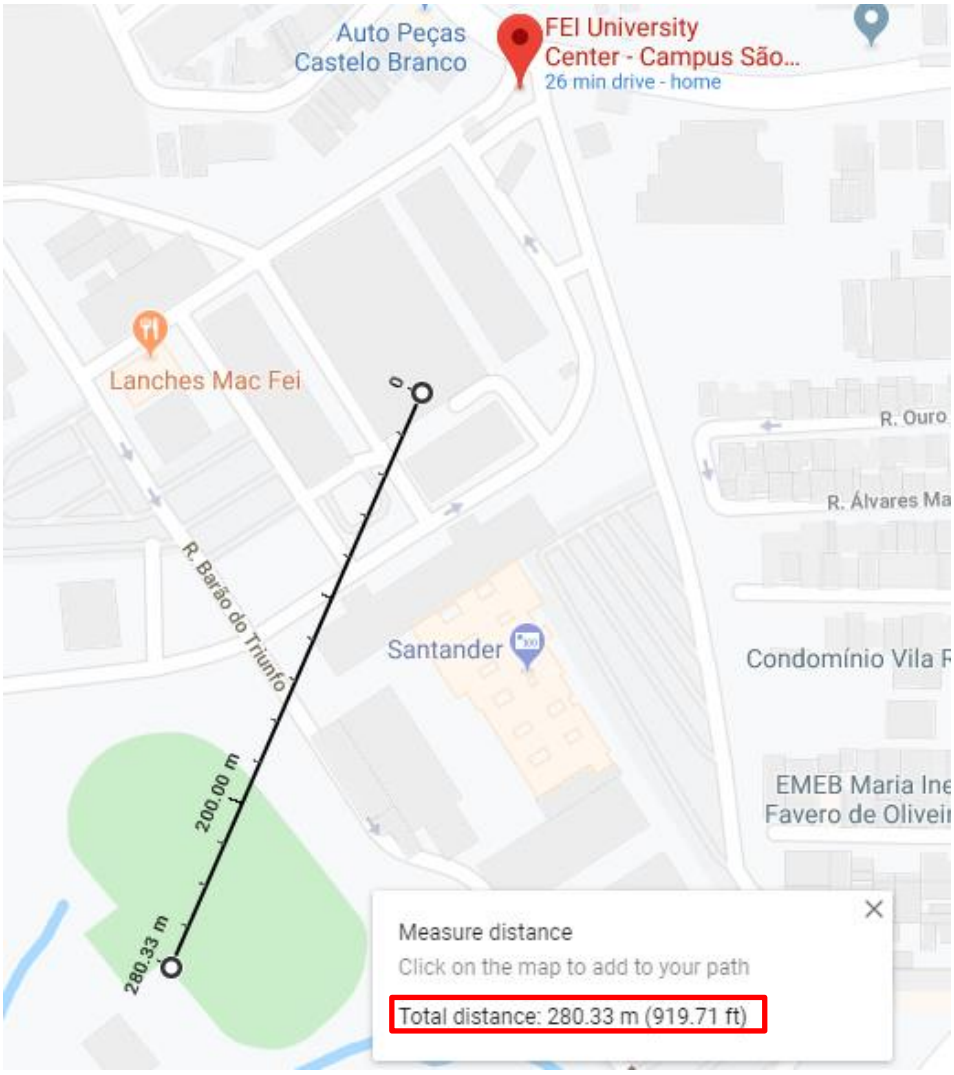
Analisando a Tabela 8, é possível observar a faixa de frequência adequada para a utilização desta tecnologia, e por esse motivo, foi escolhido o microcontrolador ESP-32 LoRa da empresa *Heltec* com uma antena de 915MHz.

4.6.2 Testes de distância com LoRa

Com o objetivo de comprovar que realmente existe uma comunicação à longa distância, foram realizados testes de envio e recepção dos dados fornecidos pelo PZEM, dentro do Campus do Centro Universitário da FEI com pouca interferência urbana.

Inicialmente, verificando a comunicação com as antenas de TX e RX próximas uma da outra, o RSSI (*Received signal strength indication* ou Indicação de força do sinal recebido) apresentava um valor igual a -9 dB. Conforme a distância aumentava, este valor ia se alterando, até que foi possível obter uma recepção das informações de forma clara com uma atenuação máxima de -120 dB, com a distância equivalente de 280 metros do transmissor, destaca-se o teste na Figura 46 a partir de uma vista do *Google Maps*.

Figura 46 – Teste de distância na comunicação entre Sender e Receiver



Fonte: Autor

Conforme os testes de comunicação via LoRa realizados por outros usuários, para a transmissão chegar a faixa dos quilômetros é necessária uma antena de maior qualidade, com um valor de frequência igual a 915 MHz, para que o sinal não seja refletido e atenuado.

Entretanto, como o foco do projeto é apenas apresentar o conceito da aplicação desta comunicação, não foi determinado como um ponto crucial, a compra de uma nova antena com tais especificações. A comprovação da aquisição de dados está apresentada na Figura 47.

Em relação ao valor do RSSI, é possível interpretá-lo através do conceito do custo do link, que permite visualizar como a linha dissipa potência durante a transmissão, e em quais regiões é necessário realizar uma ação corretiva para melhorar o link.

Figura 47 – Teste de aquisição de dados à longa distância

Fonte: Autor

Analisando melhor a questão da comunicação entre o transmissor e o emissor durante os testes, foi possível relacionar a fórmula do cálculo de potência do receptor, levando-se em conta os ganhos e perdas do sinal durante a transmissão dos dados, de acordo com a fórmula abaixo.

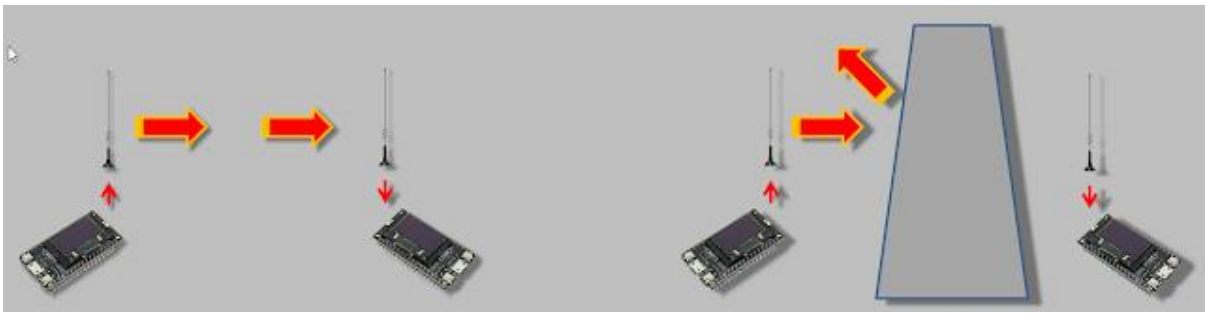
$$Potência\ recebida\ (dB) = Potência\ transmitida\ (dB) + Ganhos\ (dB) - Perdas\ (dB)$$

Adota-se nos cálculos o fator de ganho especificado pela antena, enquanto para as perdas relaciona a questão de ruídos, perdas no canal de propagação, barreiras físicas, entre alguns outros entraves no meio físico.

Além dos cuidados para evitar o aumento das perdas durante a transmissão, outro fator que não deve ser ignorado é a Linha Clara de Visão entre o transmissor e o receptor, destacado na Figura 48.

Pois mesmo com o aumento de ganhos sobre as perdas, o sinal pode ser interrompido por obstáculos, como prédios, telhados, árvores, morros, estruturas e etc, os quais influenciam diretamente na recepção das informações. Embora o cálculo leve em consideração a propagação da onda, o mesmo pressupõe uma transmissão direta sem obstáculos. Por isso, que é sempre importante a realização de testes práticos para verificar a real distância de comunicação alcançada.

Figura 48 – Exemplo de envio e recebimento de dados com obstáculos



Fonte: (FERNANDO K, 2019)

4.7 DESENVOLVIMENTO DO APLICATIVO BLYNK

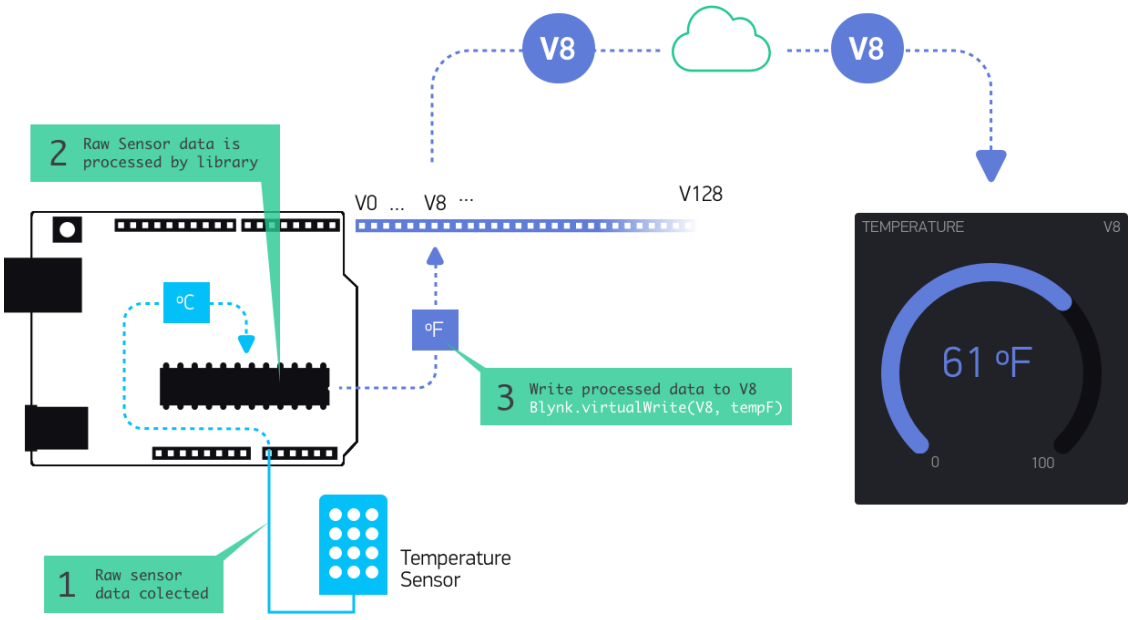
Para o desenvolvimento do aplicativo, foi definido como premissa básica a busca por um sistema que funcione com iOS e Android, e que tenha uma boa conectividade com o ESP. A partir disto, diversos sites de pesquisas acadêmicas e iniciações científicas, utilizavam o aplicativo Blynk como supervisório aos seus projetos, devida a praticidade na programação e boa interação a plataforma do Arduino IDE por possuir bibliotecas com exemplos e aplicações. Com esta motivação, foram realizados alguns testes de envio de informações para a plataforma e os mesmos atenderam as expectativas e requisitos para o projeto.

Como já citado anteriormente, o dispositivo apenas precisa estar conectado a um ponto de internet, Wifi Local, por exemplo, para que os dados sejam transmitidos via protocolo MQTT, denominado como Blynk Server. Por isso, utilizou-se o celular para o roteamento de internet ao ESP. Contudo, é possível programar roteadores dedicados para tal operação.

Todo o desenvolvimento da interface gráfica foi construído através de *widgets* já disponíveis na plataforma. A partir disto, as informações disponíveis para o usuário são apresentadas na Figura 49.

Foi necessária a criação de pinos virtuais, o qual é um conceito inventado pela *Blynk Inc.* para fornecer troca de dados entre o hardware e o aplicativo, de acordo com a Figura 49. Os pinos virtuais permitem a interface com qualquer sensor, qualquer biblioteca ou qualquer atuador, onde se utilizou para o envio das medições realizadas pelo PZEM.

Figura 49 – Envio dos dados para o aplicativo Blynk



Fonte: (BLYNK, 2019)

Na interface, o usuário consegue visualizar o valor da tensão e corrente proveniente da rede elétrica. Também é possível visualizar um gráfico interativo, com variação de períodos, como horas, dias e meses, além disso, possui um sistema para salvar todos os valores medidos e exportá-los para um arquivo de Excel (.csv). A Figura 50 apresenta todas essas funcionalidades.

Figura 50 – Interface para o monitoramento de energia residencial

Fonte: Autor

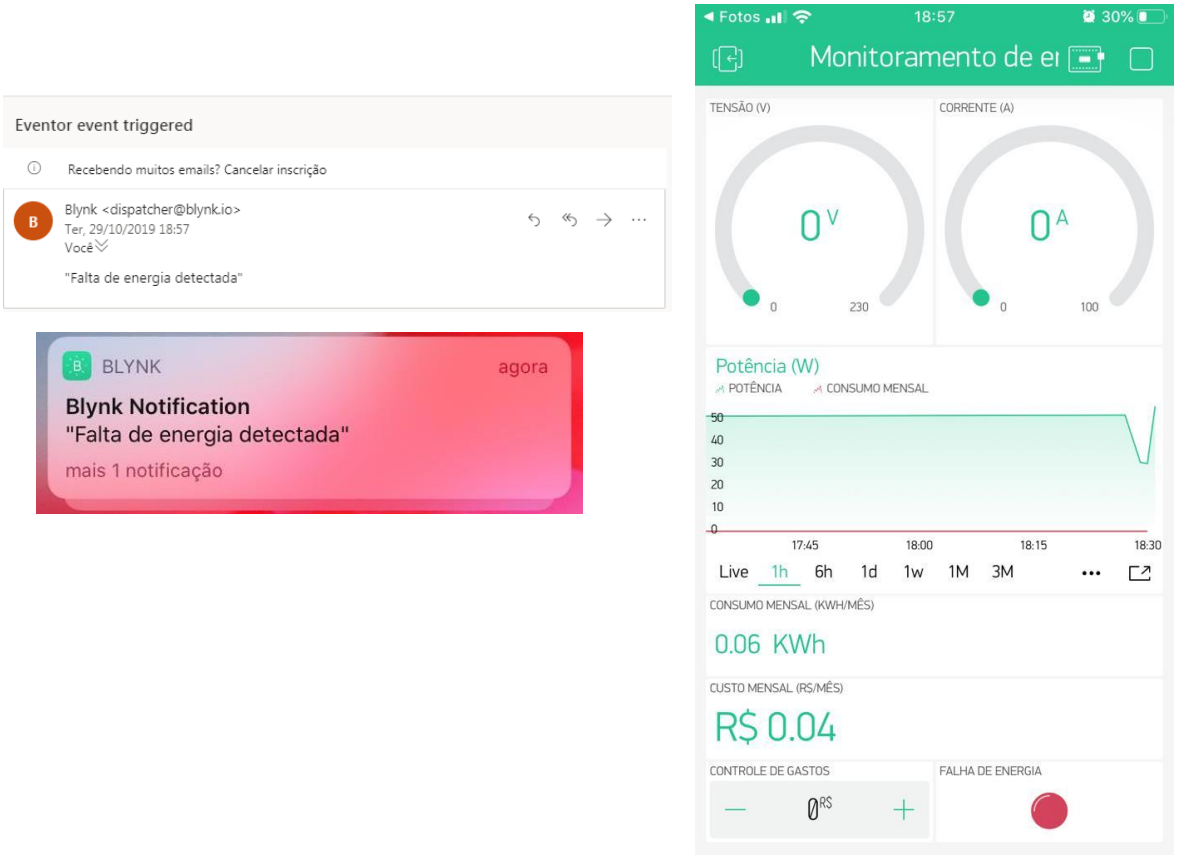
Outra proposta muito importante para o usuário é o consumo mensal e o custo obtido naquele mês, estes valores são incrementados durante o aumento da energia consumida no local. Lembrando que, todo dia 01 do mês, a partir da meia-noite, este valor é zerado.

Para uma interativa análise do consumo, foi utilizado o *Eventor* do Blynk, função que permite a criação de eventos a partir dos valores recebidos. Com isso, o usuário pode definir um controle de gastos para o mês, onde estabelece um valor máximo a ser gasto. Logo, caso o consumo real ultrapasse 50% ou 100% do valor esperado, será encaminhado uma notificação informando a respeito do gasto naquele instante.

Caso tenha alguma falha na rede elétrica do consumidor, o led de aviso irá acender e, além disso, será encaminhado um e-mail para a concessionária de energia, bem como é enviada a notificação para o celular do cliente, informando sobre possível problema detectado, de acordo com a Figura 51.

Por fim, caso deseje realizar uma análise com um período maior com os valores mais específicos, é possível obter o histórico de consumo com os valores de todas as variáveis medidas e calculadas.

Figura 51 – Envio de mensagens durante a detecção de falha de energia



Fonte: Autor

4.8 PAINEL ELÉTRICO

Para a construção do painel elétrico foi necessário estipular premissas de projeto, dimensionar disjuntores e seções nominais dos cabos elétricos, assim como, ferramentas e materiais a serem utilizadas. Por fim, foi desenvolvido um diagrama multifilar para auxiliar na montagem do mesmo.

4.8.1 Montagem do Painel Elétrico

Tendo em vista que era necessário simular o consumo residencial para validação do projeto, foi necessário montar um painel elétrico em um quadro de comando com as seguintes medidas (50 x 40 x 20 cm). Como premissa do projeto, foi padronizada região de proteção da instalação elétrica (disjuntores, DPS e interruptor diferencial residual) na parte inferior do quadro, e para a parte superior, ficou o medidor de energia desenvolvido, de acordo com a Figura 52. É de conhecimento nesse trabalho que esta configuração dificilmente é encontrada em um quadro elétrico residencial, contudo, esse sistema foi definido para que não houvesse a necessidade de colocar o medidor em uma região fora do painel elétrico.

Figura 52 – Construção do painel elétrico

Fonte: Autor

Outro ponto definido como premissa do projeto foi à alimentação do circuito como monofásica (127 Volts), pois é possível encontrar diferentes aplicações e a maioria das tomadas na FEI são 127 Volts. Além disso, um circuito trifásico aumentaria o custo do projeto, optamos assim pelo circuito monofásico.

Para o dimensionamento do painel elétrico, primeiramente foi definido o circuito de proteção para o circuito monofásico 127 Volts, juntamente com a professora Michele Rodrigues Hempel Lima, foi estipulado que seriam 5 cargas distintas com a corrente máxima de 16 Amperes e potência ativa de 1200 Watts. Com essas informações, foi determinada a potência total do circuito.

$$Potência\ total\ (P) = 1200\ x\ 5 = 6000W$$

Admitindo o fator de potência, cosØ, igual a 0.8 para as cargas aplicadas ao sistema, foi possível obter o valor da potência aparente do circuito.

$$cos\emptyset = \frac{P}{S}$$

$$Potência\ Aparente\ (S) = \frac{6000}{0.8} = 7500VA$$

Ao dividir a potência aparente pela alimentação do painel, 127 Volts, obtêm-se a corrente aproximada do disjuntor de proteção geral e do disjuntor diferencial residual (IDR).

$$I\ = \frac{7500}{127} = 60A$$

A partir do valor obtido e através da disponibilidade comercial, foi utilizado um disjuntor com a corrente nominal de 63A.

Para realizar o cálculo do dispositivo de proteção contra surtos (DPS) foi aplicada a norma NBR 5410 (ABNT, 2008) na *seção 6.3.5.3.4,* a qual descreve que este tipo de disjuntor deve atender a norma IEC 61643-1 e sua escolha deve ser com base nas características de nível de proteção, máxima tensão de operação continua, resistência à sobre-tensões temporárias, entre outras características.

Sem desconsiderar a importância dessas características, foi analisada a máxima tensão de operação contínua "Uc", conforme a figura abaixo retirada da norma NBR dita anteriormente. De acordo com a mesma, descreve que a Uc (Máxima tensão de operação contínua) deve ser maior ou igual aos valores indicados na Tabela 9 dependendo do esquema de aterramento.

Tabela 9 – Valor mínimo de Uc exigido para o DPS

DPS conectado entre				Esquema de aterramento				
Fase	Neutro	PE	PEN	TT	TN-C	TN-S	IT com neutro distribuído	IT sem neutro distribuído
X	X			1,1 U_o		1,1 U_o	1,1 U_o	
X		X		1,1 U_o		1,1 U_o	$\sqrt{3}$ U_o	U
X			X		1,1 U_o			
	X	X		U_o		U_o	U_o	

NOTAS

1 Ausência de indicação significa que a conexão considerada não se aplica ao esquema de aterramento.

2 U_o é a tensão fase–neutro.

3 U é a tensão entre fases.

4 Os valores adequados de U_C podem ser significativamente superiores aos valores mínimos da tabela.

Fonte: (ABNT, 2008)

Aplicando o coeficiente da tabela e multiplicando pela tensão da rede elétrica, obtêm-se o valor calculado de Uc.

$$Uc = 127 * 1.1 = 140\ Volts$$

Com o valor obtido e de acordo com a disponibilidade fornecida pelo mercado, foi definido a utilização do *DPS Classe II 175V / 15KVA*.

Quanto à fiação para este projeto, o dimensionamento foi feito de forma empírica, assim, foi estipulado a seção nominal dos cabos conectados com a rede elétrica igual a 10 mm² e 6 mm² para o restante.

Por fim, para a montagem do painel elétrico foi utilizado o diagrama multifilar para facilitar o entendimento, pois representa os condutores elétricos que realizam a conexão dos componentes, conforme a Figura 53 e a Tabela 10.

Figura 53 – Diagrama multifilar do quadro de luz

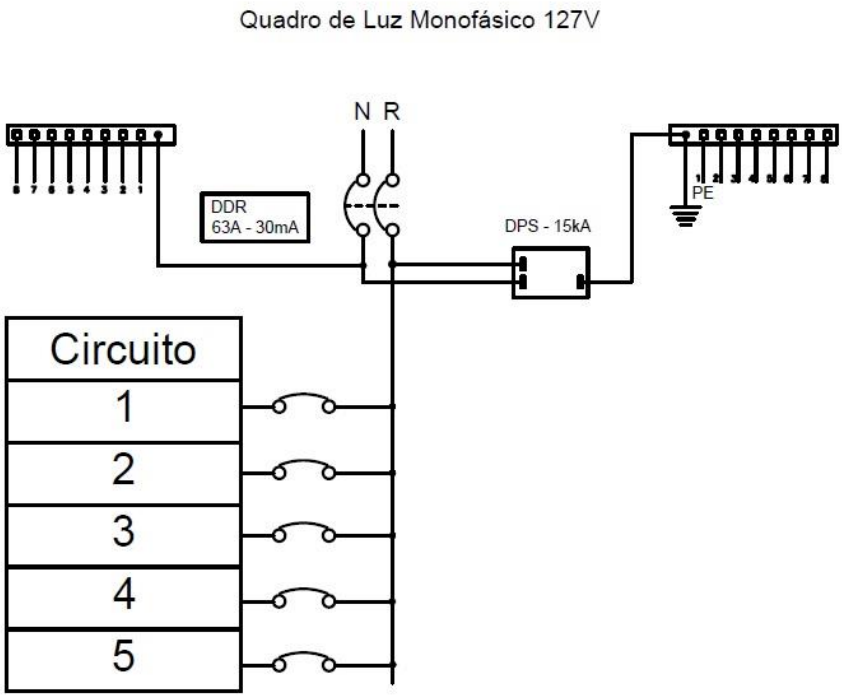


Tabela 10 – Descrição dos circuitos do painel elétrico

Recinto	Destino	Cond(mm²)	Disjuntor(A)	Circuito
Residencial	Iluminação	3 X 6 mm²	16A	1
Residencial	Tomadas	3 X 6 mm²	16A	2
Residencial	Chuveiro	3 X 6 mm²	40A	3
Residencial	Geladeira	3 x 6 mm²	16A	4
Residencial	Máquina lavar	3 x 6 mm²	16A	5

Fonte: Autor

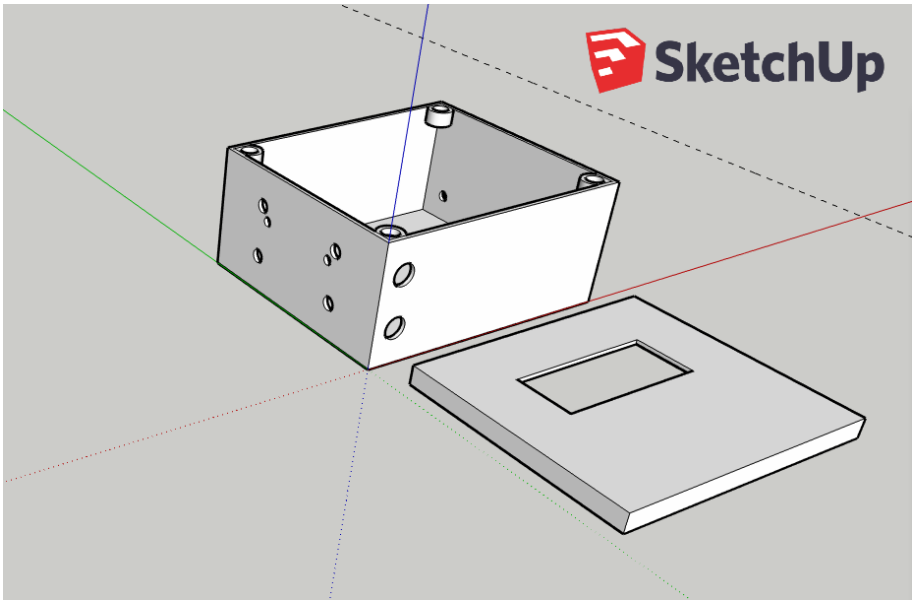
4.9 CONFECCIONAMENTO DO MEDIDOR

Para armazenar o medidor, foi definida a utilização de uma caixa produzida por meio de impressão 3D. A mesma foi projetada e modelada no *SketchUp*, software de criação em modelo 3D.

4.9.1 Modelo 3D

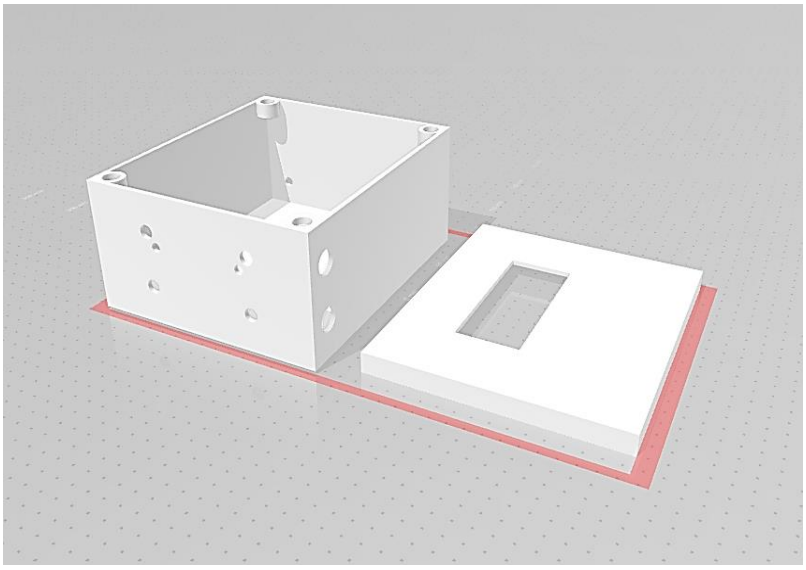
Com o progresso da montagem do painel de energia e do circuito do medidor, foi possível estimar as medidas do molde da caixa e o posicionamento no painel. Com estas medidas, e com as especificações previstas no projeto (tampa móvel, abertura para o display, furos para passagem de cabos e antena, e furos para fixação dos conectores), o modelo 3D foi criado no software *SketchUp* e exportado em formato .stl para impressão, a Figura 54 e Figura 55 exemplifica todo o processo de criação e confecção da caixa do medidor.

Figura 54 – Modelo do medidor 3D desenvolvido no SketchUp



Fonte: Autor

Figura 55 – Modelo do medidor 3D pronto para impressão (formato “.stl”)

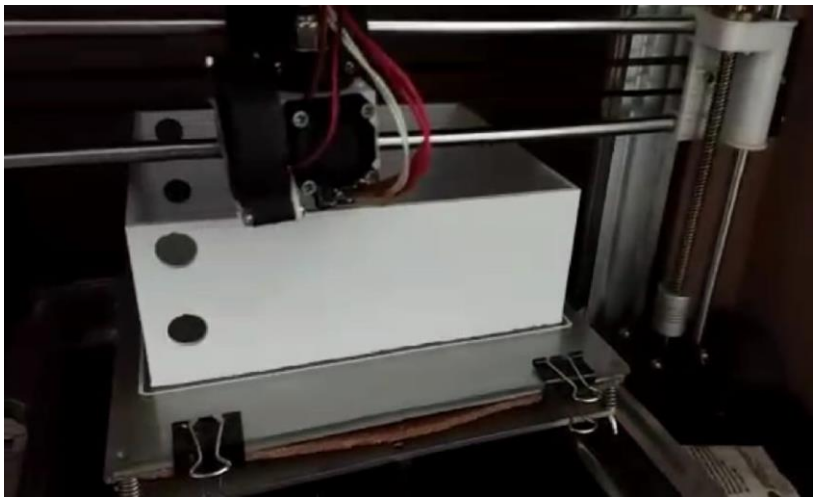


Fonte: Autor

4.9.2 Impressão 3D

O filamento utilizado para a impressão foi o termoplástico PLA (ácido polilático), cujas propriedades resultam em ótima qualidade visual e rigidez. O objetivo era utilizar a impressora 3D disponibilizada pela FEI, porém não foi possível pela mesma estar em manutenção e sem prazo para retorno. Para não comprometer os prazos do projeto, o grupo contratou um especialista para realizar a impressão, o processamento do molde é visto na Figura 56.

Figura 56 – Molde do medidor em processo de impressão



Fonte: Autor

Após a impressão do molde da caixa, foi realizado o teste para avaliar se todos os componentes estavam bem alocados, vide Figura 57, Para o Receiver, foi armazenado em uma caixa de passagem, que após adaptações realizadas, satisfez a aplicação do projeto.

Figura 57 – Testes com o molde da caixa

Fonte: Autor

4.10 RESULTADOS DO PROJETO FINAL

A partir de todo o desenvolvimento descrito acima, o projeto foi finalizado e a sua proposta desenvolvida conforme o esperado. Os resultados finais do painel elétrico e do medidor elétrico podem ser observados na Figura 58 e Figura 59.

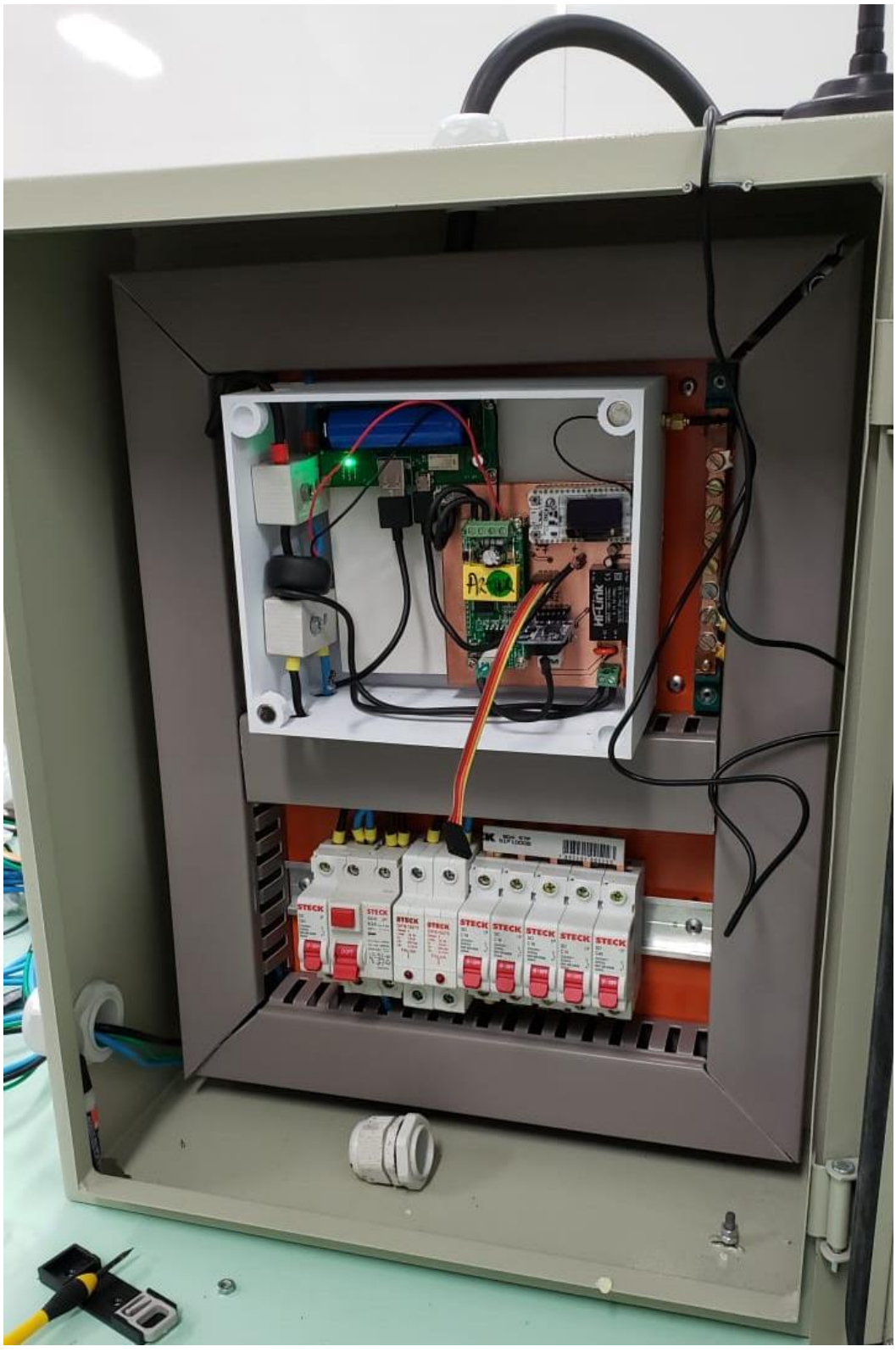
4.11 GITHUB

O GitHub é uma plataforma de repositório de código-fonte que permite aos programadores, utilitários ou qualquer usuário cadastrado na plataforma, contribuam em projetos privados e/ou *Open Source* de qualquer lugar do mundo.

Como este projeto teve diversos *insights* e exemplos de códigos *open-source,* foi compartilhada a programação do projeto, juntamente com sua descrição, com o intuito que o mesmo possa ser utilizado como base para o desenvolvimento de outros projetos e melhorias da nossa sociedade.

Site do repositório: https://github.com/galmchd92

Figura 58 – Resultados finais do projeto I



Fonte: Autor

Figura 59 – Resultados finais do projeto II

Fonte: Autor

5 CONCLUSÃO

Durante a escolha da proposta para a realização do trabalho de conclusão de curso, a motivação do grupo foi desenvolver um projeto para melhorar ou auxiliar um problema social do Brasil. A partir disso, foram listados e destacados diferentes problemas que poderiam ser mais bem estudados, ao se deparar com a situação atual do mercado elétrico neste país e seus diferentes problemas que impactam principalmente as pessoas de baixa renda, assim o projeto foi definido.

Neste projeto foi possível abordar e aplicar diferentes conceitos da área de engenharia, tais como programação em alto nível, desenvolvimento de circuitos, aplicação de sensores e circuitos de proteção. Visando o usuário final, desenvolveu-se o aplicativo inserindo as informações de forma clara e concisa para que o mesmo encontre as informações desejadas de forma rápida. O Centro Universitário da FEI nos proporcionou todo o material necessário para a realização do projeto e as ferramentas para os testes, o que nos facilitou a prática durante o desenvolvimento do projeto.

Por fim, foi adquirido muito conhecimento através dos códigos *open-source* de diferentes sites e repositórios, também foi utilizado outras pesquisas científicas para absorver o conteúdo do medidor PZEM-004T, por exemplo. Desta forma, por isso que o projeto será disponibilizado para que outras pessoas tenham acesso e possam continuar desenvolvendo e melhorando a ideia do projeto.

REFERÊNCIAS

ABRADEE. **Redes de Energia Elétrica**. Site da Associação Brasileira de Distribuidores de Energia Elétrica, 2018. Disponivel em: <hhttp://www.abradee.com.br/setor-eletrico/redes-de-energia-eletrica/>. Acesso em: 08 out. 2019.

ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 5410, A**: Instalações elétricas de baixa tensão. Rio de Janeiro: ABNT, 2008.

ALIANÇA. **Como funciona uma usina hidrelétrica**. Site da Aliança Energia, 2017. Disponivel em: <https://aliancaenergia.com.br/br/como-funciona-uma-usina-hidreletrica/>. Acesso em: 08 out. 2019.

ANEEL. NT 409-SRE. **Anexo 1 - Audiência Pública 052/2009**, 2009. Disponível em: < http://www2.aneel.gov.br/aplicacoes/audiencia/arquivo/2009/052/documento/anexo_vi_nt_304_-_medicao_de_energia.pdf>. Acesso: 01 nov. 2019.

ANEEL. **Saiba mais sobre o setor elétrico brasileiro**. Site da Agência Nacional de Energia Elétrica, 2019. Disponivel em: <https://bit.ly/2JSXnxZ>. Acesso em: 08 out. 2019.

BARROS, C. **Conheça o sistema de geração e transmissão de energia no Brasil**. Site da iG - Seção Último Segundo, 2009. Disponivel em: <https://bit.ly/36oZYZM>. Acesso em: 29 out 2019

BENEVIDES, M. **Cobrança por estimativa da conta de luz e sua ilegalidade**, 2018. Disponivel em: <https://bit.ly/36nJrFO>. Acesso em: 05 out. 2019.

BERTOLETI, P. **Projetos com ESP32 e LoRa**. [S.l.]: Instituto NCB, 2019.

BLYNK. **How Blynk works.** Site do Blynk, 2019. Disponivel em: <http://docs.blynk.cc/#intro-how-blynk-works>. Acesso em: 24 out. 2019.

CCEE. **Razão de Ser**. Site da Câmara de Comercialização de Energia Elétrica, 2019. Disponivel em: <https://bit.ly/2plY6kc>. Acesso em: 15 out. 2019.

CIA, A. E. **Módulo i2c para Display**, 2019. Disponivel em: <https://www.arduinoecia.com.br/modulo-i2c-display-16x2-arduino/>. Acesso em: 23 out. 2019.

COELBA, E. **Aprenda a Ler seu Medidor**. Site da Coelba Energia, 2018. Disponivel em: <https://bit.ly/2pv5GJ9>. Acesso em: 10 out. 2019.

DISNEY, W. **Pensador**. Sites de Frases, 1965. Disponivel em: <https://www.pensador.com/frase/ODIzMjI0/>. Acesso em: 20 out. 2019.

DISTRELEC. **Sensor LV 25-P**, 2019. Disponivel em: <https://www.distrelec.biz/en/current-sensor-14-ma-25ma-lem-lv-25/p/17696503>. Acesso em: 13 out. 2019.

ELECTRONICS, M. **Datasheet SCT013**. Site da Empresa MCI Electronics, 2019. Disponivel em: <https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_SCT013.pdf>. Acesso em: 13 out. 2019.
ELETROBRAS. **Como a energia elétrica é gerada no Brasil,** 2017. Disponivel em: <https://bit.ly/2PM8blh>. Acesso em: 07 out. 2019.

ELETRÔNICA, B. D. **Sensor de Corrente Não Invasivo SCT-013**. Site do Baú da Eletrônica, 2019. Disponivel em: <https://bit.ly/321iyE1>. Acesso em: 13 out. 2019.

ELETROPAULO. **Impostos e outros encargos**. Site da ENEL Distribuição, 2018. Disponivel em: <https://www.eneldistribuicaosp.com.br/para-sua-casa/impostos-e-outros-encargos>. Acesso em: 04 out. 2019.

EPE. **Balanço Energético Nacional 2019**. Site da Empresa de Pesquisa Energética, 2019. Disponivel em: <https://bit.ly/2JAvvOD>. Acesso em: 02 out. 2019.

FERNANDO K. **Introdução ao ESP32**. Fernando K - Tutoriais, Tecnologias, Tendências, 2019. Disponivel em: <https://www.fernandok.com/2017/11/introducao-ao-esp32.html>. Acesso em: 18 out. 2019.

GIPSON, M. **Sensors**: The Lifeblood of the Internet of Things. Semieletronics, 2019. Disponivel em: <https://semielectronics.com/sensors-lifeblood-internet-things/>. Acesso em: 15 out. 2019.

HELTEC, A. ESP-32 LORA. **Site da empresa Heltec Automation**, 2019. Disponivel em: <https://heltec.org/project/wifi-lora-32/>. Acesso em: 20 out. 2019.

INNOVATORS. **AC Digital Multifunction Meter using PZEM-004T,** 2019. Disponivel em: <https://innovatorsguru.com/ac-digital-multifunction-meter-using-pzem-004t/>. Acesso em: 20 out. 2019.

BRASIL. Casa Civil - Subchefia para Assuntos Jurídicos. **LEI Nº 9.427** de 26 dez. 1996. Disponivel em: <http://www.planalto.gov.br/ccivil_03/leis/L9427cons.htm>. Acesso em: 04 out. 2019.

LEM. **LV-25P.** Site da Empresa LEM, 2019. Disponivel em: <https://www.lem.com/sites/default/files/products_datasheets/lv_25-p.pdf>. Acesso em: 13 out. 2019.

LEM. **Sensor LA 125-P**. Site da LEM, 2019. Disponivel em: <http://pdf.datasheetcatalog.com/datasheet/lem/LA125-P.pdf>. Acesso em: 13 out. 2019.

LIGHT. **Composição da Tarifa**. Site da Empresa Light, 2019. Disponivel em: <https://bit.ly/326fp63>. Acesso em: 23 out. 2019.

LORA. **Case Studies.** Site do LORA - Developer Portal, 2019. Disponivel em: <https://lora-developers.semtech.com/library/case-studies/>. Acesso em: 16 out. 2019.

MAX, D. **Na sua conta de luz, você paga 26 taxas, incluindo IPVA, IPTU e carvão**. Site da UOL, 2018. Disponivel em: <https://bit.ly/2JBrv0q>. Acesso em: 04 out. 2019.

MAXIM, I. **Datasheet RTC DS3231**, 2019. Disponivel em: <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>. Acesso em: 20 out. 2019.
MME. **Consumo de energia das famílias brasileiras cresce em 2019 e acena positivamente para o consumo sustentável**. Site do Ministério de Minas e Energia, 2019. Disponivel em: <https://bit.ly/2WuaPgC>. Acesso em: 01 out. 2019.

NANSEN. **Medidores de energia**. Site da fabricante Nansen, 2019. Disponivel em: <http://nansen.com.br/medidores/lumen3/>. Acesso em: 11 out. 2019.

NEVES, F. D. S. **Uma Abordagem de Segurança para os Dados Transmitidos por Dispositivos em Internet das Coisas**. Recife. 2017.

PEVEDUTO. **Distribuição de energia elétrica no Brasil: como acontece?** Site do Grupo Pevedeto, 2017. Disponivel em: <http://peveduto.com.br/distribuicao-de-energia-eletrica-no-brasil-como-acontece/>. Acesso em: 11 out. 2019.

ROUSE, M. **IoT analytics guide: Understanding Internet of Things data.** Tech Target, 2019. Disponivel em: <https://internetofthingsagenda.techtarget.com/definition/machine-to-machine-M2M>. Acesso em: 15 out. 2019.

RSDELIVERS. **LEM LA 125-P.** Site da Empresa RSDELIVERS, 2019. Disponivel em: <https://bit.ly/2Wu6c6k>. Acesso em: 13 out. 2019.

SAOKAEW, A.; CHIEOCHAN, O.; BOONCHIENG, E. A smart photovoltaic system with Internet of Thing: A case study of the smart agricultural greenhouse. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE AND SMART TECHNOLOGY (KST), 10., Tailândia, 2018. Tailândia: IEEE, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8426071>. Acesso em: 01 nov. de 2019

SEMTECH. **What is LoRA ?** Sita da Semtech Products, 2019. Disponivel em: <https://www.semtech.com/lora/what-is-lora>. Acesso em: 17 out. 2019.

SENA, G. E. D. O. **Medidor de Consumo de Energia Elétrica com acesso local e remoto usando a plataforma ESP8266**. 110 f. Alegrete. TCC (Graduação em Engenharia Elétrica) - UNIPAMPA, Alegrete, 2018.

SIN. Sistema Interligado Nacional. **Site do Operador Nacional do Sistema Elétrico**, 2018. Disponivel em: <https://bit.ly/2qga9iL>. Acesso em: 09 out. 2019.

TRISUL, U. **Como é feita a distribuição de energia elétrica no Brasil.** Site da Universidade Trisul, 2019. Disponivel em: <https://bit.ly/34hZaUJ>. Acesso em: 05 out. 2019.

TYAGI, S.; JAIN, P. **Internet of Things using LPWAN.** Bangkok: [s.n.], 2019.

WANG, Z.; ZHAI, M. **Research on Application of LPWAN in State Monitoring of Distribution Network.** [S.l.]: IOP Conference Series: Materials Science and Engineering, 2019.

APÊNDICE A – PROGRAMAÇÃO DO SENDER

APÊNDICE A – PROGRAMAÇÃO DO SENDER

```
/* TCC - Medidor de consumo elétrico - PROJETO 8
/*
Testes de utilização do ESP32 com o sensor PZEM
*/
// ---------------------- SENDER --------------------------
// ---------------------- Bibliotecas utilizadas
----------------------------
#include <SPI.h> //responsável pela comunicação serial
#include <LoRa.h> //responsável pela comunicação com o WIFI
Lora
#include <Wire.h> //responsável pela comunicação i2c
#include <HardwareSerial.h> // Hardware do PZEM
#include <PZEM004T.h> // Biblioteca de comunicação do PZEM
#include <RTClib.h> // Biblioteca de comunidação RTC que
funcionou com Arduino/ ESP
#include <LiquidCrystal_I2C.h> // Biblioteca do LCD
// ---------------------- Observações
----------------------------
/*
* Deve-se usar o código acima para identificar os endereço I2C
* D1 (05) = SCL
* D2 (04) = SDA
* Endereços scaneados:
* 0x27 = LCD Display
* 0x57 = EEPROM RTC DS3231
* 0x68 = PCF8574
*/
// ---------------------- PZEM --------------------------
// Definição do Hardware do PZEM
HardwareSerial PzemSerial2(2); // Utilizar o hwserial
UART2 nos pinos IO-16 (RX2) e IO-17 (TX2)
PZEM004T pzem(&PzemSerial2);
IPAddress ip(192,168,1,1);
bool pzemrdy = false; // Variável da conexão do PZEM
// Variáveis para armazenar e tratar leituras do PZEM
float v, i, p, e, maior, menor, x, y, z, a = 0;
float v1,v2,v3 = 0;
float c1,c2,c3 = 0;
float p1,p2,p3 = 0;
float e1,e2,e3 = 0;
float e_aux=0;
float v_offset = 1;
float consumo_mensal, cons_aux, custo_mensal=0;
// ---------------------- LoRA --------------------------
// Definição dos pinos LORA
#define SCK 5 // GPIO5 -- SX127x's SCK
#define MISO 19 // GPIO19 -- SX127x's MISO
#define MOSI 27 // GPIO27 -- SX127x's MOSI
```

```
#define SS 18 // GPIO18 -- SX127x's CS
#define RST 14 // GPIO14 -- SX127x's RESET
#define DI00 26 // GPIO26 -- SX127x's IRQ(Interrupt
Request)
#define BAND 915E6 //Frequencia do radio - podemos
utilizar ainda : 433E6, 868E6, 915E6
#define PABOOST true
// --------------------- LCD --------------------------
LiquidCrystal_I2C lcd(0x27,20,4); // set the LCD address to
0x3F for a 16 chars and 2 line display
// Entradas do LCD
#define SDA1 21
#define SCL1 22
// --------------------- RTC --------------------------
// Variáveis do tempo
uint32_t ts1, ts2, ts3, ts, tfinal, tint=0;
RTC_DS3231 rtc; // Definição da imagem do RTC
char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terça",
"Quarta", "Quinta", "Sexta", "Sábado"};
// Entradas do RTC (Real Time Clock)
//#define SDA2 21
//#define SCL2 22
//======================== SETUP
============================================================
============================
void setup()
{
// Inicialização da comunicação serial
Serial.begin(115200);
// ============================ Inicialização do LCD
============================
Serial.println("Inicialização LCD");
lcd.begin();
// ============================ Inicialização do LORA
============================
SPI.begin(SCK,MISO,MOSI,SS); // Inicia a comunicação
serial com o Lora
LoRa.setPins(SS,RST,DI00); // Configura os pinos que serão
utlizados pela biblioteca (deve ser chamado antes do LoRa.
begin)
Serial.print ("Iniciando LoRa ...");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Iniciando LoRa ...");
delay(1000);
// Inicializa o Lora com a frequencia específica.
if (!LoRa.begin(BAND))
{
Serial.println("Inicialização do LoRa falhou");
//Acompanhamento no serial
lcd.setCursor(0,1);
```

```
lcd.print("Inicialização do LoRa falhou");
while (1);
}
// Indica no display que iniciou corretamente.
Serial.println("LoRa Conectado!");
//Acompanhamento no serial
lcd.setCursor(0,0);
lcd.print("LoRa Conectado!");
delay(3000);
// ============================ Inicialização do PZEM
============================
pzem.setAddress(ip);
while (!pzemrdy)
{
Serial.println("Iniciando o PZEM...");
//Acompanhamento no serial
lcd.setCursor(0,2);
lcd.print("Iniciando o PZEM ...");
pzemrdy= pzem.setAddress(ip);
delay(500); // Aguarda 1/2 seg
}
Serial.println("PZEM Conectado!");
lcd.setCursor(0,3);
lcd.print("PZEM Conectado!");
delay(3000);
lcd.clear();
// ============================ Inicialização do RTC
============================
while (!rtc.begin()) {
Serial.println("RTC NÃO encontrado!");
lcd.setCursor(0,0);
lcd.print("RTC NAO encontrado..");
}
Serial.println("RTC encontrado!");
lcd.setCursor(0,0);
lcd.print("RTC encontrado!");
delay(1000);
if (rtc.lostPower()) {
Serial.println("RTC perdeu alimentação, vamos ajustar
data-hora!");
lcd.clear();
lcd.setCursor(0,1);
lcd.print("Ajuste data-hora");
//http://www.esp32learning.com/code/esp32-and-ds3231-rtc-examp
le.php
// Linhas abaixo devem ser utilizadas apenas para ajuste
da data-hora do RTC
// rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
// Por exemplo p/ Outubro 15, 2019 às 21:27 seria:
//rtc.adjust(DateTime(2019, 10, 15, 21, 27, 0));
// lcd.setCursor(0,1);
```

```
// lcd.print("Manual Adj...");
delay(1000);
}
// ============================ Aquisição da data e hora e
início da rotina ============================
DateTime now = rtc.now();
Serial.print ("Aquisição de data e hora");
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Energy monitoring");
lcd.setCursor(0,1);
lcd.print("Data: ");lcd.print(now.day(), DEC);lcd.
print('/');lcd.print(now.month(), DEC);lcd.print('/');
lcd.print(now.year(), DEC);
lcd.setCursor(0,2);
lcd.print("Hora: "); lcd.print(now.hour(), DEC);lcd.
print(':');lcd.print(now.minute(), DEC);lcd.print(':');
lcd.print(now.second(), DEC);
Serial.print(now.day(), DEC);Serial.print('/');
Serial.print(now.month(), DEC); Serial.print('/');
Serial.print(now.year(), DEC);
Serial.print ("Início das medições...");
lcd.setCursor(0,3);
lcd.print("Medindo....");
delay(3000);
}
//================================== LOOP
================================================================
========================
void loop()
{
DateTime now = rtc.now();
if(((now.day(), DEC)==1)&& ((now.minute(), DEC)==00) &&
((now.hour(), DEC)==00) && (e > 0) ){ // Reajuste do consumo
de energia para todo dia 1 às 00:00
cons_aux=e;
}
else{
// ++++++++++++++++++++ Loop de aquisição dos dados do PZEM
++++++++++++++++++++
// Aquisição dos dados a serem tratados - 3 resultados para
que sejam filtrados posteriormente
// ------------ Tensão ---------------------------------
ts1=millis(); // Início da contagem do loop
v1 = pzem.voltage(ip)+ v_offset; // Aquisição da medição de
tensão
delay(500);
while (v1 <= 0.0){ // Loop condicional enquanto não for
detectado nenhuma tensão
delay(500);
v1 = pzem.voltage(ip) + v_offset; // V_offset = Utilizado
```

```
porque o PZEM devolve um valor igual a -1 no resultado de
tensão igual a 0
ts=millis(); // Contagem do final do loop
if(ts-ts1 > 10000){ // Caso o loop ultrapassar 10
segundos, será considerado como falta de energia de fase
faltadeenergia();
}
}
ts2=millis(); // Repetição do loop para as outras duas
medições
v2 = pzem.voltage(ip)+ v_offset;
delay(500);
while (v2 <= 0.0){
delay(500);
v2 = pzem.voltage(ip)+ v_offset;
ts=millis();
if(ts-ts2 > 10000){
faltadeenergia();
}
}
ts3=millis();
v3 = pzem.voltage(ip)+ v_offset;
delay(500);
while (v3 <= 0.0){
delay(500);
v3 = pzem.voltage(ip)+ v_offset;
ts=millis();
if(ts-ts3 > 10000){
faltadeenergia();
}
}
// ------------- Corrente ----------------------------------
// Repetição da aquisição de dados
c1 = pzem.current(ip);
delay(500);
while (c1 < 0.0){
delay(500);
c1 = pzem.current(ip);
}
c2 = pzem.current(ip);
delay(500);
while (c2 < 0.0){
delay(500);
c2 = pzem.current(ip);
}
c3 = pzem.current(ip);
delay(500);
while (c3 < 0.0){
delay(500);
c3 = pzem.current(ip);
}
```

```
// ------------- Potência ----------------------------------
p1 = pzem.power(ip);
delay(500);
while (p1 < 0.0){
delay(500);
p1 = pzem.power(ip);
}
p2 = pzem.power(ip);
delay(500);
while (p2 < 0.0){
delay(500);
p2 = pzem.power(ip);
}
p3 = pzem.power(ip);
delay(500);
while (p3 < 0.0){
delay(500);
p3 = pzem.power(ip);
}
// ------------- Consumo de energia
----------------------------------
e1 = pzem.power(ip);
delay(500);
while (e1 < 0.0){
delay(500);
e1 = pzem.power(ip);
}
e2 = pzem.power(ip);
delay(500);
while (e2 < 0.0){
delay(500);
e2 = pzem.power(ip);
}
e3 = pzem.power(ip);
delay(500);
while (p3 < 0.0){
delay(500);
e3 = pzem.power(ip);
}
// ---------------------------------- Entrada do Loop do
tratamento de dados espúrios
----------------------------------
espurgov(v1, v2, v3);
espurgoi(c1, c2, c3);
espurgop(p1, p2, p3);
espurgoe(e1, e2, e3);
// Retorno dos dados filtrados (v,i,p,e) ---> e convertido
para kWh
// ---------------------------------- Reajuste do consumo
mensal de energia ----------------------------------
if (e > e_aux){
```

```
e_aux=e; // Recebe o valor do consumo caso for maior que
o valor medido anteriormente para se manter atualizado
}
else {
e=e_aux; // Caso o valor medido for zero ( representando
que não tem carga ) ou menor que o anterior, atualiza o valor
do consumo para o anterior
}
// -------------------------------- Cálculo do consumo
mensal de energia --------------------------------
//cons_aux=e; // Condição inicial utilizada apenas para
definição do consumo mensal inicial
// ** DEVE SER RETIRADO APÓS A PRIMEIRA ITERAÇÃO
if (e < cons_aux){ // Caso a constante de consumo auxiliar
for maior que o consumo atual
consumo_mensal=cons_aux;
Serial.print("Problemas com o consumo mensal");
}
else{
consumo_mensal=e-cons_aux; // Subtração do consumo base
do mês com o consumo decorrido do mês
}
// Cálculo do custo mensal
if(consumo_mensal >= 0){
custo_mensal=0.65823*consumo_mensal; // Multiplicação R$
0,65823 por kWh (Dados obtidos em SP: 2019)
}
else{
Serial.print("Problemas com o custo mensal");
}
// ---------------------------------------- PZEM Serial
Monitor ----------------------------------------
Serial.print("Tensão(V): "); Serial.println(v); // Envio
de dados para a comunicação serial
Serial.print("Corr.(A): "); Serial.println(i); // Envio de
dados para a comunicação serial
Serial.print("Pot. (W):"); Serial.println(p); // Envio de
dados para a comunicação serial
Serial.print("Consumo (kWh):"); Serial.println(e); // Envio
de dados para a comunicação serial
Serial.print("Consumo Mensal (kWh):"); Serial.
println(consumo_mensal); // Envio de dados para a comunicação
serial
Serial.print("Custo Mensal (R$):"); Serial.
println(custo_mensal); // Envio de dados para a comunicação
serial
Serial.println(); // Pula uma tabulação após a finalização
dos dados
delay(3000); // Aguarda 3 milisegundos para a nova iteração
// ---------------------------------------- Configuração do
LCD para apresentar os dados
```

```
-----------------------------------------
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Tensao(V): ");lcd.print(v);
lcd.setCursor(0,1);
lcd.print("Corrente(A): ");lcd.print(i);
lcd.setCursor(0,2);
lcd.print("Potencia(W): ");lcd.print(p);
lcd.setCursor(0,3);
lcd.print("Consumo(kWh): ");lcd.print(e);
//lcd.setCursor(0,3);
//lcd.print("Consumo Mensal(kWh/mês): ");lcd.
print(consumo_mensal);
//lcd.setCursor(0,3);
//lcd.print("Custo mensal(R$/mês): ");lcd.
print(custo_mensal);
delay(3000); // Aguarda 3 milisegundos para a nova iteração
// ---------------------------------------- Envio dos
dados através do LoRa
-----------------------------------------
//beginPacket : abre um pacote para adicionarmos os dados
para envio
//print: adiciona os dados no pacote
//endPacket : fecha o pacote e envia -> retorno= 1:sucesso |
0: falha
LoRa.beginPacket();
LoRa.print("Tensão (V): ");
LoRa.print(v); //Tensão
LoRa.endPacket();
Serial.println("Tensao enviada");
delay(2000);
LoRa.beginPacket();
LoRa.print("Corrente (A): ");
LoRa.print(i); //Corrente
LoRa.endPacket();
Serial.println("Corrente enviada");
delay(2000);
LoRa.beginPacket();
LoRa.print("Potência (W): ");
LoRa.print(p);
LoRa.endPacket();
Serial.println("Potencia enviada");
delay(2000);
LoRa.beginPacket();
LoRa.print("Consumo (kWh): ");
LoRa.print(e); //Watts hora
LoRa.endPacket();
Serial.println("Consumo enviado");
delay(2000);
LoRa.beginPacket();
LoRa.print("Consumo mensal (kWh/mês): ");
```

```
LoRa.print(consumo_mensal); //Watts hora
LoRa.endPacket();
Serial.println("Consumo mensal enviado");
delay(2000);
LoRa.beginPacket();
LoRa.print("Custo mensal (R$/mês): ");
LoRa.print(custo_mensal); //Watts hora
LoRa.endPacket();
Serial.println("Custo mensal enviado");
delay(2000);
LoRa.beginPacket();
LoRa.print("Data (DD/MM/YYYY): ");
LoRa.print(now.day(), DEC);LoRa.print('/');LoRa.print(now.
month(), DEC);LoRa.print('/');LoRa.print(now.year(),
DEC); // Data RTC
Serial.print(now.day(), DEC);Serial.print('/');
Serial.print(now.month(), DEC); Serial.print('/');
Serial.println(now.year(), DEC);
LoRa.endPacket(); // Envio das horas RTC
Serial.println("Data enviada");
delay(2000);
LoRa.beginPacket();
Serial.print(now.hour(), DEC);Serial.print(':');
Serial.print(now.minute(), DEC);Serial.print(':');
Serial.println(now.second(), DEC);
LoRa.print("Hora (HH:MM:SS): ");
LoRa.print(now.hour(), DEC);LoRa.print(':');LoRa.print(now.
minute(),D EC);LoRa.print(':');LoRa.println(now.second(),
DEC);
LoRa.endPacket(); // Envio das horas RTC
Serial.println("Hora enviada");
delay(2000);
Serial.println();
} // Fechamento do else da potência mensal
// ---------------------------------- Cálculo do tempo de
iteração do sistema ----------------------------------
tfinal=millis(); // Início da contagem do loop
tint=tfinal-ts1;
Serial.print("Tempo de iteração do sistema (seg): ");
Serial.println(tint);
}
/*============================================================
==========================================================
* BLOCO DE FUNÇÕES UTILIZADAS
* Funções de espurdo dados quebrados
*/
/* Tratamento de valores espúrios (a lib do PZEM foi escrita
para arquitetura AVR, não ESP
* Armazena 3 valores em sequência e prepara para comparação/
limpeza de valores espúrios
*/
```

```
// Leitura e Tratamento dos valores de Tensão
void *espurgov(float x, float y, float z){
float maior = x; float menor = x;
if (y > menor){
menor = y;
}
else {
menor = 0.0;
}
if (y > maior && y >= 0.0){
maior = y;
}
if (z > menor){
menor = z;
}
else {
menor = 0.0;
}
if (z > maior && z >= 0.0){
maior = z;
}
v = maior;
}
// Leitura e Tratamento dos valores de Corrente
void *espurgoi(float x, float y, float z){
float maior = x; float menor = x;
if (y > menor){
menor = y;
}
else {
menor = 0.0;
}
if (y > maior && y >= 0.0){
maior = y;
}
if (z > menor){
menor = z;
}
else {
menor = 0.0;
}
if (z > maior && z >= 0.0){
maior = z;
}
i = maior;
}
// Leitura e Tratamento dos valores de Potência Instantânea
void *espurgop(float x, float y, float z){
float maior = x; float menor = x;
if (y > menor){
menor = y;
```

```
}
else {
menor = 0.0;
}
if (y > maior && y >= 0.0){
maior = y;
}
if (z > menor){
menor = z;
}
else {
menor = 0.0;
}
if (z > maior && z >= 0.0){
maior = z;
}
p = maior;
}
// Leitura e Tratamento dos valores de Energia Consumida
void *espurgoe(float x, float y, float z){
float maior = x; float menor = x;
if (y > menor){
menor = y;
}
else {
menor = 0.0;
}
if (y > maior && y >= 0.0){
maior = y;
}
if (z > menor){
menor = z;
}
else {
menor = 0.0;
}
if (z > maior && z >= 0.0){
maior = z;
}
e = maior/1000; // Conversão para kWh
}

// Tratamento de dados para execução do ciclo de detecção de
falta de energia de fase
void *faltadeenergia(){
LoRa.beginPacket();
LoRa.print("Falta de energia detectada ! ");
LoRa.endPacket();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Falta de energia ");
```

```
lcd.setCursor(4,1);
lcd.print("detectada ! ");
Serial.println("Falta de energia detectada ! ");
delay(2000);
a=0;
v1 = pzem.voltage(ip)+ v_offset;
delay(500);
while (v1 <= 0.0){ // Loop condicional enquanto não retorna
a energia ao sistema
v1 = pzem.voltage(ip)+ v_offset;
delay(500);
a++;
Serial.println("Tentativa de reconexão: "); Serial.
print(a);
lcd.setCursor(0,2);
lcd.print("Reconexao.... ");
lcd.setCursor(0,3);
lcd.print(a);
}
}
```

APÊNDICE B – PROGRAMAÇÃO DO RECEIVER

APÊNDICE B – PROGRAMAÇÃO DO RECEIVER

```
/* TCC - Medidor de consumo elétrico - PROJETO 8
/*
Testes de utilização do ESP32 com o sensor PZEM
*/
// ---------------------- RECEIVER ---------------------------
// ---------------------- Bibliotecas utilizadas --------------
-------------
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <SPI.h> //responsável pela comunicação serial
#include <LoRa.h> //responsável pela comunicação com o Lora
#include "SSD1306.h" //responsável pela comunicação com o
display
#include <Wire.h> //responsável pela comunicação i2c
#include <LiquidCrystal_I2C.h> // Biblioteca do LCD
// Definição das variáveis do APP Blynk
float voltage_blynk=0;
float current_blynk=0;
float power_blynk=0;
float energy_blynk=0;
float consumo_blynk=0;
float custo_blynk=0;
float falta_blynk=0;
// Colocar o Auth Token disponível no Blynk App
/* Inserir o "Auth Token" obtido no Blynk App. Vá até "Project
Settings" (ícone
parafuso),
* entre "" a seguir.
* É enviado ao e-mail cadastrado ou pode ser copiado para área
transferência
SmartPhone.
* WiFi: Fazer o NodeMCU Lolin conectado na Rede,
* SSID e Senha devem ser escritos entre respectivas "" a
seguir.
*/
char auth[] = "ReTdIP_oDz5MDJ37-vQ3GLTrtRfV0DDM";
// Credenciais da Wifi Local
char ssid[] = "XXXX"; //colocar o ssid da rede
char pass[] = "XXXX"; //colocar a senha da sua rede
/*============================================================
====================
================================
* DEFIINIÇÃO DAS VARIÁVEIS DO LORA
*/
// Definição dos pinos
#define SCK 5 // GPIO5 -- SX127x's SCK
#define MISO 19 // GPIO19 -- SX127x's MISO
```

```
#define MOSI 27 // GPIO27 -- SX127x's MOSI
#define SS 18 // GPIO18 -- SX127x's CS
#define RST 14 // GPIO14 -- SX127x's RESET
#define DI00 26 // GPIO26 -- SX127x's IRQ(Interrupt Request)
// Frequência do radio - podemos utilizar ainda : 433E6,
868E6, 915E6
#define BAND 915E6
#define PABOOST true
// Parâmetros: address,SDA,SCL
SSD1306 display(0x3c, 4, 15); //construtor do objeto que
controlaremos o display
String rssi = "RSSI --";
String packSize = "--";
String packet ;
// --------------------- LCD ---------------------------
LiquidCrystal_I2C lcd(0x27,20,4); // set the LCD address to
0x3F for a 16 chars
and 2 line display
// Entradas do LCD
#define SDA1 21
#define SCL1 22
/*============================================================
====================
==================================
* SETUP, INICIALIZAÇÃO DAS VARIÁVEIS, FUNÇÕES E BIBLIOTECAS
*/
void setup() {
//Setup OLED
pinMode(16,OUTPUT); //RST do oled
digitalWrite(16, LOW); // Reseta o OLED
delay(50);
digitalWrite(16, HIGH); // Enquanto o OLED estiver ligado,
GPIO16 deve estar
HIGH
display.init();
display.flipScreenVertically();
display.setFont(ArialMT_Plain_10);
delay(1500);
display.clear();
// Setup LCD
Serial.println("Inicialização LCD");
lcd.begin();
// Indica no display que inicilizou corretamente.
display.drawString(20, 10, "Processo Iniciado com sucesso");
display.drawString(20, 30, "Preparando Periféricos...");
display.display();
delay(000);
// Setup LoRa
Serial.begin(115200); // Inicialização da comunicação serial
SPI.begin(SCK,MISO,MOSI,SS); //inicia a comunicação serial com
o Lora
```

```
LoRa.setPins(SS,RST,DI00); //configura os pinos que serão
utlizados pela
biblioteca (deve ser chamado antes do LoRa.begin)
// Debug Lora
Serial.println("Iniciando LoRa ..."); //Acompanhamento no
serial
display.clear();
display.drawString(0, 0, "Iniciando LoRa...");
display.display();
delay(1000);
// Inicializa o Lora com a frequência específica.
if (!LoRa.begin(BAND,PABOOST)) // PABOOST não funciona
{
Serial.println("Inicialização do LoRa falhou !");
//Acompanhamento
no serial
while (1);
}
delay(1000);
// Indica no display que inicilizou corretamente.
Serial.println("LoRa iniciado com sucesso!"); //Acompanhamento
no
serial
display.clear();
display.drawString(0, 0, "LoRa iniciado com sucesso!");
display.display();
delay(1000);
//LoRa.onReceive(cbk);
LoRa.receive(); // Habilita o Lora para receber dados
//NTP - Contagem do tempo
// timerun = millis();
// Início do APP Blynk
Serial.println("Iniciando Blink...");
display.clear();
display.drawString(0, 0, "Iniciando Blink...");
display.display();
delay(1000);
Blynk.begin(auth, ssid, pass); //Faz a conexão com o Blynk
Serial.println("Blink Iniciado!");
display.clear();
display.drawString(0, 0, "Blink Iniciado!");; //Imprime
Sucesso na conexão...
display.display();
delay(500);
// Início do APP Blynk
Blynk.begin(auth, ssid, pass); //Faz a conexão com o Blynk
// Inicialização do Blynk
Blynk.run();
Serial.println("Blink run...");
display.clear();
display.drawString(0, 0, "Blink run...");
```

```
display.display();
delay(1000);
}
/*============================================================
====================
==================================
* LAÇO DE EXECUÇÃO DAS ROTINAS
*/
void loop() {
// Inicialização do Blynk
Blynk.run();
//parsePacket: checa se um pacote foi recebido
//retorno: tamanho do pacote em bytes. Se retornar 0 (ZERO)
nenhum pacote foi
recebido
int packetSize = LoRa.parsePacket();
//caso tenha recebido pacote chama a função para configurar os
dados que serão
mostrados em tela
if (packetSize) {
cbk(packetSize);
}
}
//função responsável por recuperar o conteúdo do pacote
recebido
//parametro: tamanho do pacote (bytes)
void cbk(int packetSize) {
packet ="";
packSize = String(packetSize,DEC); //transforma o tamanho do
pacote em String
para imprimirmos
for (int i = 0; i < packetSize; i++) {
packet += (char) LoRa.read(); //recupera o dado recebido e
concatena na
variável "packet"
}
rssi = "RSSI= " + String(LoRa.packetRssi(), DEC)+ "dB";
//configura a String
de Intensidade de Sinal (RSSI)
//mostrar dados em tela
loraData();
}
//função responsável por configurar os dadosque serão exibidos
em tela.
//RSSI : primeira linha
//RX packSize : segunda linha
//packet : terceira linha
void loraData(){
display.clear();
display.setTextAlignment(TEXT_ALIGN_LEFT);
display.setFont(ArialMT_Plain_10);
```

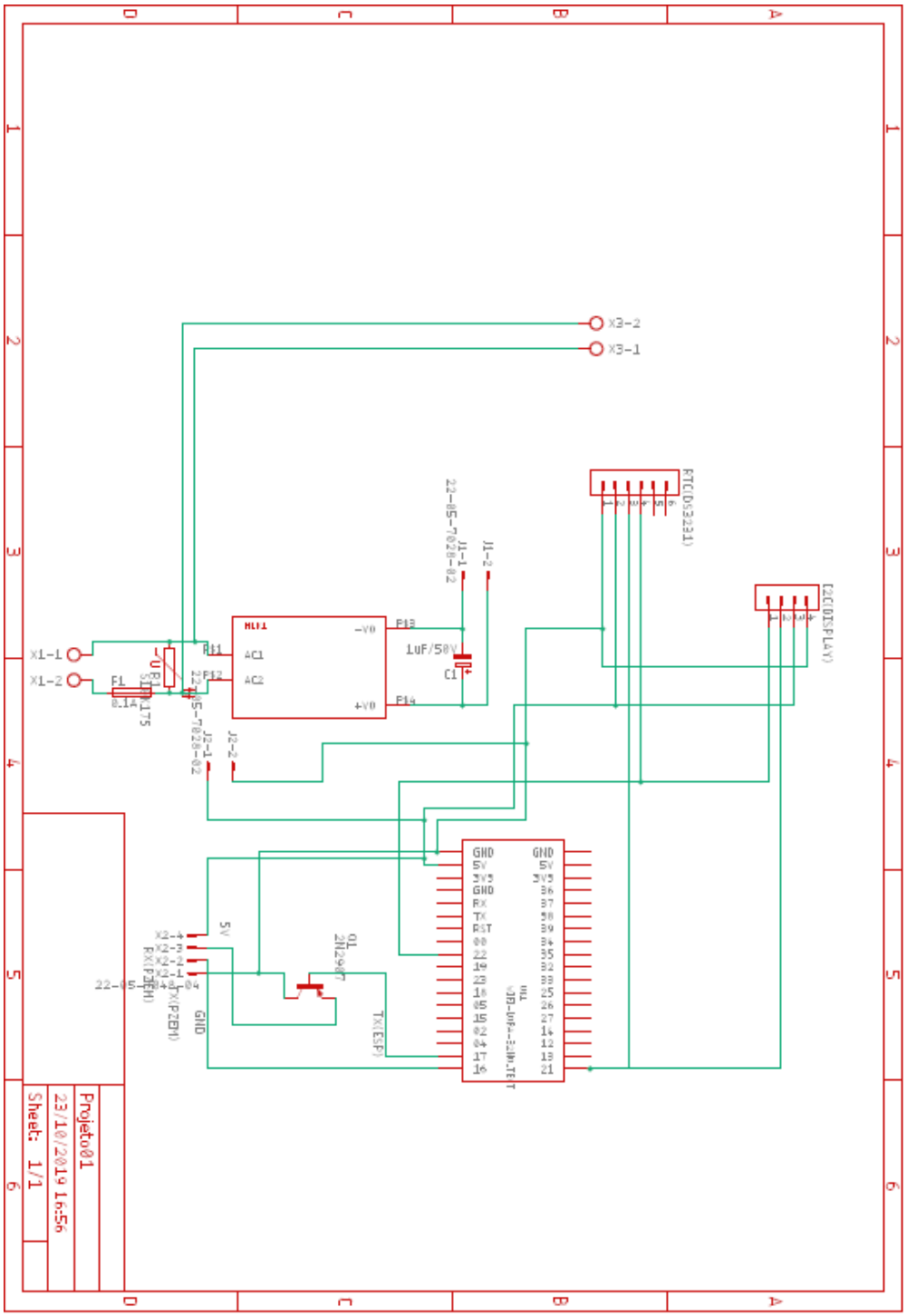
```
display.drawString(0 , 18 , "Adquirido "+ packSize + " bytes
de dados");
display.drawString(0 , 39 , packet);
display.drawString(0, 0, rssi);
display.display();
blynk();
}
// ---------------------------------------- BLYNK
----------------------------------------
void blynk (){
int tamanho = packet.length(); // Indica o tamanho total do
pacote
recebido/posição do final
// stringaAProcurar.indexOf(stringProcurada)
// Verifica qual é o parametro/ o número resultante indica a
posição que a
variável foi achada
// Se o retorno for -1 que dizer que não foi encontrado
int teste = packet.indexOf("(V)");
Blynk.virtualWrite(V9, 0); // apenas preparação pra futuro
evento de falta de
energia
if (teste > -1)
{ lcd.clear();
lcd.setCursor(0,0);
lcd.print(packet);
// String procurada encontrada
// Procura 2 posições anteriores ao valor do parametro pois o
valor
sempre estará após um : e um espaço (2 posições)
int doispontos = packet.indexOf(": ");
// Monta o valor que existe duas posições após o : até a
posição ao final
do pacote
String voltage_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Tensão recebida");
voltage_blynk.toFloat();
Serial.println(voltage_blynk);
// Envia o valor para o Blynk
Blynk.virtualWrite(V1, voltage_blynk);
}
teste = packet.indexOf("(A)");
if (teste > -1)
{ lcd.setCursor(0,1);
lcd.print(packet);
int doispontos = packet.indexOf(": ");
String current_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Corrente recebida");
current_blynk.toFloat();
```

```
Serial.println(current_blynk);
Blynk.virtualWrite(V2, current_blynk);
}
teste = packet.indexOf("(W)");
if (teste > -1)
{ lcd.clear();
lcd.setCursor(0,0);
lcd.print(packet);
int doispontos = packet.indexOf(": ");
String power_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Potência recebida");
power_blynk.toFloat();
Serial.println(power_blynk);
Blynk.virtualWrite(V3, power_blynk);
}
teste = packet.indexOf("(kWh)");
if (teste > -1)
{ lcd.setCursor(0,1);
lcd.print(packet);
int doispontos = packet.indexOf(": ");
String energy_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Energia recebida");
energy_blynk.toFloat();
Serial.println(energy_blynk);
Blynk.virtualWrite(V4, energy_blynk);
}
teste = packet.indexOf("(kWh/mês)");
if (teste > -1)
{ lcd.clear();
lcd.setCursor(0,0);
lcd.print(packet);
int doispontos = packet.indexOf(": ");
String consumo_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Consumo recebido");
consumo_blynk.toFloat();
Serial.println(consumo_blynk);
Blynk.virtualWrite(V5, consumo_blynk);
}
teste = packet.indexOf("(R$/mês)");
if (teste > -1)
{ lcd.setCursor(0,1);
lcd.print(packet);
int doispontos = packet.indexOf(": ");
String custo_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Custo mensal recebido");
custo_blynk.toFloat();
Serial.println(custo_blynk);
```

```
Blynk.virtualWrite(V6, custo_blynk);
}
teste = packet.indexOf("Falta de energia");
if (teste > -1)
{
int doispontos = packet.indexOf(": ");
String falta_blynk = packet.substring(doispontos + 2,
tamanho);
Serial.println("Falta de energia detectada");
Blynk.virtualWrite(V9, 255); //Enviar 255 pra setar o LED em
NL alto e
0 pra NL baixo
Blynk.virtualWrite(V1, 0.0); // Atualizar o valor da tensão
pra 0
Blynk.virtualWrite(V2, 0.0); // Atualizar o valor da corrente
pra 0
}
}
```

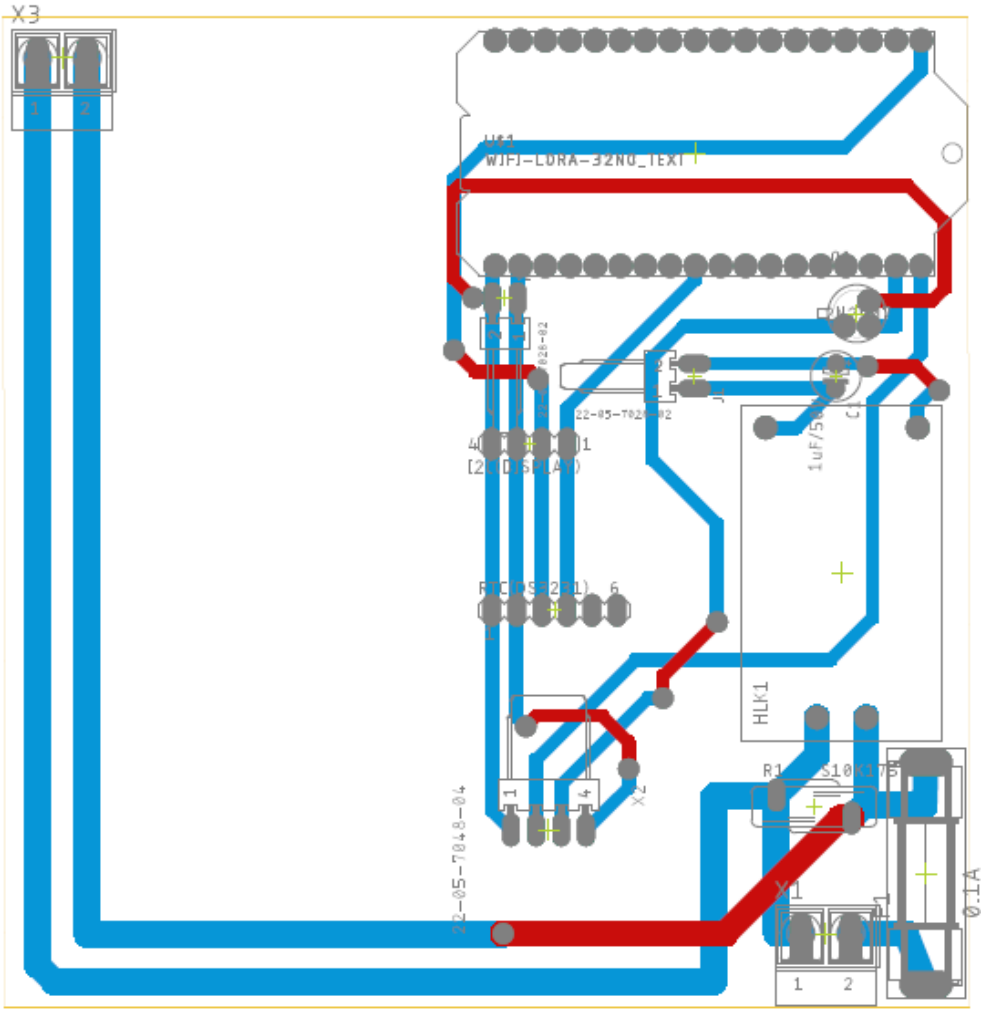
APÊNDICE C – ESQUEMÁTICO DO CIRCUITO SENDER

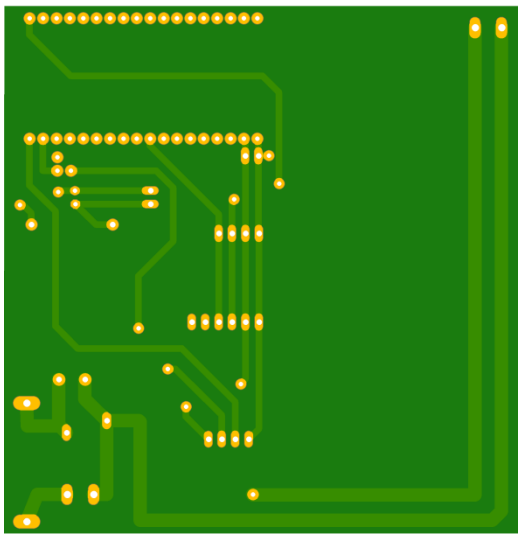
APÊNDICE C – ESQUEMÁTICO DO CIRCUITO SENDER

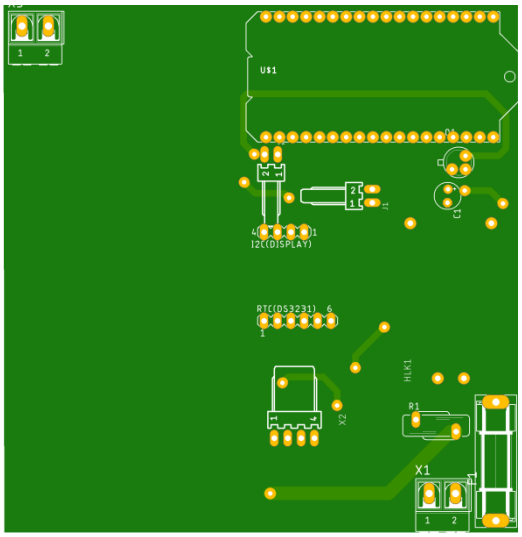


APÊNDICE D – LAYOUT DO CIRCUITO SENDER

APÊNDICE D –LAYOUT DO CIRCUITO SENDER







ANEXO A – DATASHEET DO PZEM-004T

ANEXO A – DATASHEET DO PZEM-004T

Product Type: PZEM-004T(V1.0)

A. Function

1. Electrical parameter measurement function (voltage, current, active power, energy).
2. The reset function of energy key.
3. Store data when power off (store the accumulated energy before power off).
4. PC display function (display voltage, current, active power, energy).
5. Serial communication function (with TTL serial interface itself, can communicate with a variety of terminal through the pin board, read and set the parameters).

Front display and key

Figure 1 PC display diagram

I. Display Interface

PC display interface is formed by four windows, used to display the voltage, current, power, energy parameters, as shown in figure 1.

II. Display Format

1. Power: Test Range: 0 ~ 22kW

   Within 0 ~ 10kW, the display format is 0.000 ~ 9.999;

   Within 10 ~ 22kW, the display format is 10.00 ~ 22.00.

2. Energy: Test Range: 0 ~ 9999kWh

   Within 0 ~ 10kWh, the display format is 0.000 ~ 9.999;

Within 10 ~ 100kWh, the display format is 10.00 ~ 99.99;

Within 100 ~ 1000kWh, the display format is 100.0 ~ 999.9;

1000 ~ 9999kWh and above, the display format is 1000 ~ 9999.

3. Voltage: Test Range: 80 ~ 260VAC

Display Format is 110.0 ~ 220.0.

4. Current: Test Range: 0 ~ 100A

III. Key

There is a key on the panel, it can be used to reset energy, as shown in figure 2.

The method of reset energy: Long press the key for 5 seconds, then release the key. Short press the key again, then the energy data is cleared and quit the reset state, now the reset operation is completed.

C. Wiring diagram

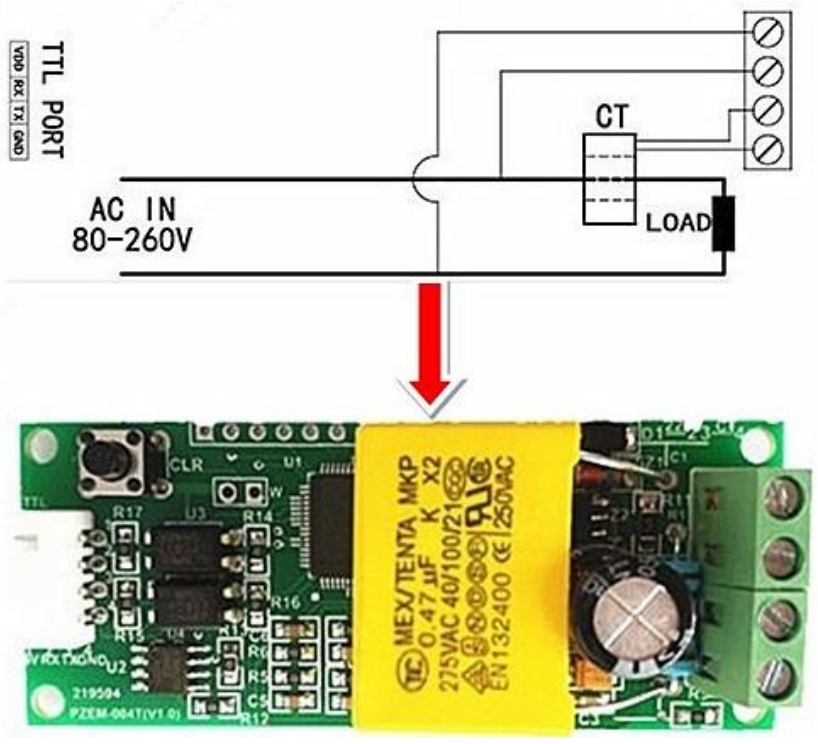


Figure 2 Wiring diagram

The wiring of this module is divided into two parts: the voltage and current test input terminal wiring and the serial communication wiring, as shown in Figure 2; according to the actual needs of the clients, with different TTL pin board to achieve communicate with different terminals.

Display Interface

It display through PC or other terminals (need to programme),the PC display interface is shown as figure 1, the following are brief description of each parameter display:

1. Voltage Display

Measure and display the current power frequency network voltage.

2. Current display

Measure and display the current load (appliances) current. There is supplementary instruction that the current test value is from the beginning of 10mA , but this module belongs to high power test equipment, if you care about the mA level current testing accuracy, it is not be recommended.

3. Energy display

Measure and display the current accumulative power consumption. There is supplementary instruction that the minimum unit of the energy metering is 0.001kWh,which means it begins to accumulate from 1Wh, relatively speaking, the resolution is rather high, for the low-power(within 100W)load test, you can observe the accumulative process rather intuitively.

4. Power display

Measure and display the current load power. There is supplementary instruction that the power test value is from the beginning of 0.001kW , which means it begins to test from 1W, but this module belongs to high power test equipment, if you have the requirement of the testing within 1W, it is not be recommended.

E. Serial communication

This module is equipped with TTL serial data communication interface, you can read and set the relevant parameters via the serial port; but if you want to communicate with a device which has USB or RS232 (such as computer), you need to be equipped with different TTL pin board (USB communication needs to be equipped with TTL to USB pin board; RS232 communication needs to be equipped with TTL to RS232 pin board), the specific connection type as shown in Figure 2. In the below table are the communication protocols of this module:

NO.	Function	Head	Data1- Data5	Sum

1	voltage	B0	C0 A8 01 01 00 (Computer sends a request to read the voltage value)	1A
		A0	00 E6 02 00 00 (Meter reply the voltage value is 230.2V)	88
2	current	B1	C0 A8 01 01 00 (Computer sends a request to read the current value)	1B
		A1	00 11 20 00 00 (Meter reply the current value is 17.32A)	D2
3	Active power	B2	C0 A8 01 01 00 (Computer sends a request to read the active power value)	1C
		A2	08 98 00 00 00 (Meter reply the active power value is 2200w)	42
4	Read energy	B3	C0 A8 01 01 00 (Computer sends a request to read the energy value)	1D
		A3	01 86 9f 00 00 (Meter reply the energy value is 99999wh)	C9
5	Set the module address	B4	C0 A8 01 01 00 (Computer sends a request to set the address, the address is 192.168.1.1)	1E
		A4	00 00 00 00 00 (Meter reply the address was successfully set)	A4
6	Set the power alarm threshold	B5	C0 A8 01 01 14 (computer sends a request to set a power alarm threshold)	33
		A5	00 00 00 00 00 (Meter reply the power alarm threshold was successfully set)	A5

Illustration of the communication protocol example：

1. Set the communication address: 192.168.1.1

Send command: B4 C0 A8 01 01 00 1E

Reply data: A4 00 00 00 00 00 A4

Note: The above example illustrate that setting the communication address as 192.168.1.1 (the user can set their own address based on their preferences and needs), sending commands and replying data automatically are as shown above, the data are expressed in hexadecimal; the last byte of the sending and replying data are 1E and A4, belong to cumulative sum. At sending commands: B4 + C0 + A8 + 01 + 01 + 00 = 21E (use the hexadecimal addition), the cumulative sum data is 21E, take the last two bytes 1E to be used the cumulative sum data in sending commands; data in reply: A4 + 00 + 00 + 00 + 00 + 00 = A4 (use the hexadecimal addition),the cumulative sum data is A4,which is the cumulative sum data in reply.

The explanation of the cumulative sum is now finished, the following parameter examples are the same as this, there is no explanation any more.

2. Set the power alarm threshold:20 KW

Send command: B5 C0 A8 01 01 14 33

Reply data: A5 00 00 00 00 00 A5

Note: 14 in the sending command is the alarm value (14 is a hexadecimal data representation, which converted to decimal is 20). What you should note is the power alarm value of this module is based on KW units, which means the minimum alarm value is 1KW, the maximum value is 22KW.

3. Read the current voltage

Send command: B0 C0 A8 01 01 00 1A

Reply data: A0 00 E6 02 00 00 88

Note: Reply voltage data is D1D2D3 = 00 E6 02, 00 E6 represent the integer-bit of the voltage, 02 represent the decimal of the voltage, the decimal is one digit, converts 00 E6 to decimal is 230; converts 02 to decimal is 2, so the current voltage value is 230.2V.

4. Read the current current

Send command: B1 C0 A8 01 01 00 1B

Reply data: A1 00 11 20 00 00 D2

Note: Reply current data is D2D3 = 11 20, 11 represent the integer-bit of the current, 20 represent the decimal of the current, the current decimal is two digits, converts 11 to decimal is 17; converts 20 to decimal is 32, so the current current value is 17.32 A**.**

5. Read the current power

Send command: B2 C0 A8 01 01 00 1C

Reply data: A2 08 98 00 00 00 42

Note: Reply power data is D1D2 = 08 98, converts 08 98 to decimal is 2200, so the current voltage value is 2200W.

6. Read the energy

Send command: B3 C0 A8 01 01 00 1D

Reply data: A3 01 86 9F 00 00 C9

Note: Reply energy data is D1D2D3 = 01 86 9F, converts 01 86 9F to decimal is 99999, so the accumulated power is 99999Wh.

Illustration of the communication

Connect hard wire according to the wiring diagram in figure 2.

After connect the wire, please choose the communication port, this module's upper computer software support communication port: COM2\COM3\COM4, you can check through device manager, if it is not the above communication port, you should amend it through port.

Precautions

1. This module is suitable for indoor, please do not use outdoor.
2. Applied load should not exceed the rated power.
3. Wiring order can't be wrong.

H. Specification parameters

1. Working voltage: 80 ~ 260VAC
2. Test voltage: 80 ~ 260VAC
3. Rated power: 100A/22000W
4. Operating frequency: 45-65Hz
5. Measurement accuracy: 1.0 grade