Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Curso de Bacharelado em Ciências da Computação

Caetano Colin Torres

Solução Escalável para Monitoramento do Estado da Corrente Elétrica das Redes em Ambientes Remotos

Florianópolis-SC, Brasil
2022

Caetano Colin Torres

Solução Escalável para Monitoramento do estado da Corrente Elétrica das Redes em Ambientes Remotos

Trabalho de Conclusão de Curso submetido ao Programa de Graduação em Ciência da Computação para a obtenção do Grau de Bacharel em Ciência da Computação na Universidade Federal de Santa Catarina

Orientador: Prof. Dr. Carlos Becker Westphall

Florianópolis-SC, Brasil

2022

Caetano Colin Torres

Solução Escalável para Monitoramento do estado da Corrente Elétrica das Redes em Ambientes Remotos

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Ciência da Computação”, e aprovado em sua forma final pelo Programa de Graduação em Ciência da Computação.

Florianópolis, 01 de dezembro de 2022.

Prof. Dr. Carlos Becker Westphall

Professor Orientador e Responsável

Banca Examinadora:

Prof. Dr. Carlos Becker Westphall

Universidade Federal de Santa Catarina (UFSC)

Carla Merkle Westphall

Universidade Federal de Santa Catarina (UFSC)

Rodrigo Pescador

Rede Nacional de Ensino e Pesquisa (RNP)

Edison Melo

Universidade Federal de Santa Catarina (UFSC)

RESUMO

A solução engloba o desenvolvimento de um dispositivo IoT estável, de baixo custo, escalável, de fácil instalação e manutenção que irá monitorar o estado da rede elétrica em sites remotos. Um hardware de placa única (Raspberry Pi Zero) será acoplado com um módulo de bateria com fornecimento de energia elétrica ininterrupto, junto a um modem USB que usará tecnologia de telefonia móvel para a transmissão de dados na falta de conexão com a rede local do ambiente remoto. Esse hardware irá ler informações dos pinos GPIO e informar se há corrente elétrica disponível na rede. Além disso, o sistema desenvolvido será integrado com o sistema de monitoramento atual do PoP-SC/RNP, cujo nome é Zabbix, e enviará informações adicionais periódicas do dispositivo (uptime, intensidade do sinal do modem, nível da bateria e consumo da franquia de dados) para o ambiente central. Com essas informações enviadas pelo dispositivo de monitoramento será possível tomar decisões mais assertivas, facilitando e agilizando o processo de abertura de chamados com as operadoras de transporte contratadas pela empresa

Palavras-chave: Raspberry Pi, IoT, Energy Availability

ABSTRACT

The solution encompasses the development of a stable, low cost, scalable, easy to install and maintain IoT device that will monitor the state of the electrical grid at remote sites. A single board hardware (Raspberry Pi Zero) will be coupled with a battery module with electrical power supply, along with a USB modem that will use mobile phone technology for data transmission in the absence of connection to the environment's local network. This hardware will read information from the GPIO pins and tell if there is electrical current available on the network or not. In addition, the developed system will be integrated with the current monitoring system of the PoP-SC/RNP, Zabbix, and will send periodic additional information from the device (uptime, modem signal strength, battery level and consumption of the data) to the central environment. With this information sent by the monitoring device, it will be possible to make more assertive decisions, facilitating and speeding up the process of making requests with the connectivity transport providers hired by the company.

Palavras-chave: Raspberry Pi, IoT, Energy Availability

Lista de Figuras

Lista de abreviaturas e siglas

APN - Access Point Name (Nome do Ponto de Acesso);

CELESC - Centrais Elétricas de Santa Catarina;

CPE - Customer Premises Equipment (Equipamento das Instalações do Cliente);

GPIO - General Purpose Input Output (Entrada e Saída de Propósito Geral);

GPRS - General Packet Radio Service;

GSM - Global System for Mobile Communications;

ICCID - Integrated Circuit Card Identification Number;

IFSC - Instituto Federal de Santa Catarina;

IMSI - International Mobile Subscriber Identity;

IoT - Internet of Things;

LED - Light-emitting diode;

LoRa - Long Range;

PoA - Ponto de Agregação;

PoE - Power over Ethernet;

PoP - Ponto de Presença;

PoP-SC - Ponto de Presença da RNP em Santa Catarina;

REMEP-FLN - Rede Metropolitana Comunitária de Educação e Pesquisa da Região de Florianópolis;

RNP - Rede Nacional de Ensino e Pesquisa;

RT - Request Tracker;

SIM - Subscriber Identity Module;

SLA - Service Level Agreement;

TCC - Trabalho de Conclusão de Curso;

UFSC - Universidade Federal de Santa Catarina;

UMTS - Universal Mobile Telecommunications System;

UPS - Uninterruptible Power Supply;

USB - Universal Serial Bus;

VPN - Virtual Private Network;

WAN - Wide Area Network.

SUMÁRIO

1. INTRODUÇÃO

1.1 MOTIVAÇÃO

No Ponto de Presença da RNP em Santa Catarina, atende-se às necessidades operacionais da rede acadêmica brasileira e a conectividade das instituições parceiras, que são instituições de ensino e pesquisa. E, na Rede Metropolitana Comunitária de Educação e Pesquisa da Região de Florianópolis, que faz parte de uma iniciativa da RNP, atende-se às necessidades das organizações conectadas na rede metropoliFIBUtana. Em ambas iniciativas, é necessário monitorar a conectividade dos clientes à rede e gerenciar as possíveis falhas que possam surgir. Na maioria dos casos, quando um assinante da rede fica sem conexão, acionamos a operadora parceira que provê o enlace até a última milha para efetuar os eventuais consertos e possibilitar a reconexão do cliente na rede.

Nesse sentido, quando alguma Organização Usuária aparece indisponível no sistema de monitoramento, uma das principais causas é a indisponibilidade de energia no campus ou unidade. Somente comunicamos a operadora que fornece o transporte entre a organização e o Ponto de Presença da RNP dessa indisponibilidade quando confirmamos que o local remoto está com energia elétrica, porque é inviável acionar uma operadora para solucionar um problema inexistente na infraestrutura de trânsito dela, afinal, o ofensor nesse caso é a instituição que está sem energia, e não a operadora.

Esse projeto é um projeto IoT desenvolvido em conjunto com o PoP-SC/RNP e REMEP-FLN. Atualmente, quando ocorre indisponibilidade de energia elétrica em um site remoto, é necessário estabelecer contato com a instituição ou organização cliente via telefone ou e-mail para confirmar a disponibilidade, acontece que na maior parte das vezes a instituição que está sem energia elétrica não está de prontidão para dar o retorno em relação a disponibilidade. Por isso, o sistema a ser desenvolvido será elaborado com o fim de agilizar o processo de verificação de disponibilidade de energia elétrica em sites remotos, de modo que seja assertivo e praticamente instantâneo, facilitando a abertura de chamados com as provedoras dos serviços de comunicação quanto a indisponibilidade da conexão em sites remotos e melhorando os tempos de respostas nos SLA's.

A Internet das Coisas (IoT) é um ecossistema em constante expansão que integra software, hardware, objetos físicos e dispositivos de computação para comunicar, coletar e trocar dados. A IoT fornece uma plataforma perfeita para facilitar as interações entre humanos e uma variedade de coisas físicas e virtuais (KASHANI, 2021)[7]. Peter Newman prevê que haverá mais de 41 bilhões de dispositivos IoT conectados até 2027. Ele também estima, que o

mercado de IoT está a caminho de crescer para mais de US$2,4 trilhões anualmente até 2027 [9].

1.2 JUSTIFICATIVA

Com o objetivo de visualizar a carga de trabalho que a equipe do PoP-SC tem sobre esses chamados de energia elétrica, foi executada uma consulta no sistema RT do PoP-SC/RNP. O RT é uma ferramenta de rastreamento de chamados de nível empresarial, ela permite a organização acompanhar tarefas e suas atribuições. Ou seja, todos os chamados abertos com as operadoras parceiras estão nesse sistema de forma estruturada, sendo possível realizar consultas e visualizar gráficos.

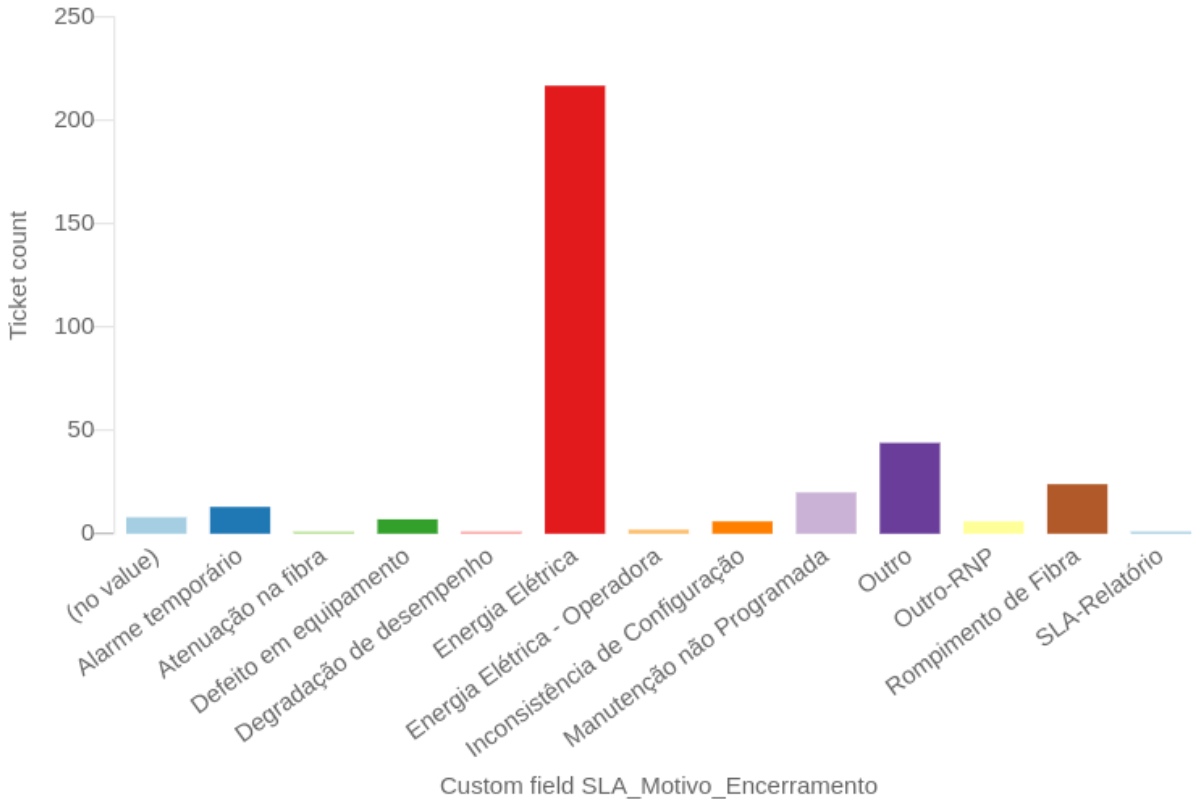

Figura 1. Gráfico do RT.

O gráfico explicitado na Figura 1 acima, replica a consulta que busca os chamados na fila de *SLA* do PoP-SC, e criados a partir de 01 de janeiro de 2022 até 26 de fevereiro de 2022, agrupados pelo campo "SLA_Motivo_Encerramento" que indica o motivo do encerramento do chamado. O gráfico mostra que dos 350 chamados buscados, 217 deles, ou seja, 62% dos chamados, foram encerrados por uma falha ou indisponibilidade de energia elétrica. Mostrando que a maioria dos incidentes abertos com o PoP-SC/RNP na fila de SLA são incidentes elétricos. Além disso, em qualquer um desses chamados, o primeiro procedimento de um operador é verificar se existe energia elétrica no local da

indisponibilidade, para descartar que a indisponibilidade ocorreu simplesmente por uma falha elétrica, portanto, em qualquer um dos casos o dispositivo de monitoramento iria ajudar na operação.

1.3. OBJETIVOS

1.3.1 OBJETIVOS GERAIS

Como objetivo geral, espera-se ao final do trabalho, um *hardware* agregado e funcional, montado num computador de placa única e acoplado em um módulo de bateria com fornecimento ininterrupto, capaz de enviar informações de indisponibilidade elétrica via modem USB e uma integração nos sistemas de monitoramento do PoP-SC/RNP e REMEP-FLN.

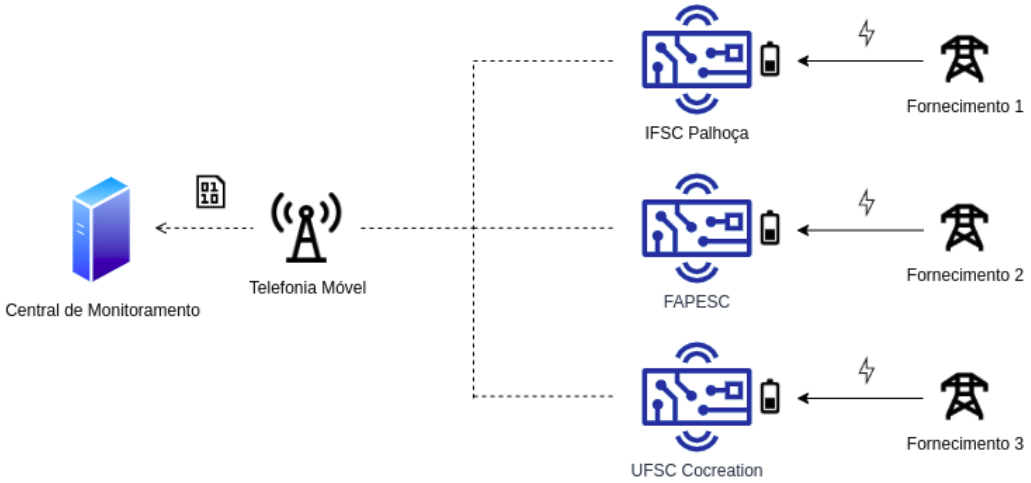

Figura 2. Arquitetura da Solução.

O diagrama acima explicita o objetivo geral de modo a expor a arquitetura da solução em uma aplicação prática, onde por exemplo, existem 3 dispositivos em 3 campi remotos cada um conectado com uma rede de fornecimento elétrico, denominados de UFSC Cocreation, FAPESC e IFSC Palhoça, esses dispositivos são independentes mas enviam variáveis controladas localmente para um ambiente central de monitoramento através de uma conexão WAN via Rádio.

1.3.2 OBJETIVOS ESPECÍFICOS

De modo mais específico, será necessário, além de montar o dispositivo, desenvolver um script de verificação de tensão e corrente, que irá ler os pinos do *raspberry pi* e determinar se está sendo fornecido energia elétrica via rede elétrica e a carga da bateria. Também será necessário o desenvolvimento de um script de controle de conexão *out-of-band* via rádio, que irá monitorar o estado da conectividade do dispositivo com o ambiente central e se necessário restabelecer a conexão.

Além disso, espera-se também um utilitário de rede para monitorar o tráfego na interface USB, porque deseja-se controlar e monitorar o uso de banda deste dispositivo, visto que geralmente, os planos com as operadoras de telefonia ou IoT são proporcionais ao uso de banda. Outro utilitário que deseja-se desenvolver é um script de controle de *LED* que indica o estado da conexão *WAN*.

Considera-se também o funcionamento desse projeto como parte do escopo, um script para enviar status genéricos periódicos para o ambiente central de monitoramento, por exemplo, *uptime* do equipamento e intensidade do sinal de rede. Todos esses envios de dados devem estar integrados com o sistema de monitoramento do PoP-SC/RNP e REMEP-FLN que, atualmente, é a ferramenta *Zabbix*.

Em relação a premissas e restrições, temos como premissa o acesso ao dispositivo raspberry pi junto com os módulos adicionais, que são, o modem USB, a bateria e a fonte de alimentação. Para facilitar, apelidamos o dispositivo pronto para uso de *PiMon*. Como restrições, temos que o custo de um dispositivo *PiMon* deve permitir a escalabilidade.

1.4 ORGANIZAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO

O Trabalho está estruturado em seções, na seção de escopo será descrito o escopo que o trabalho abrange, o que será desenvolvido e quais são as expectativas a serem entregues ao final do desenvolvimento do trabalho. Na seção de Metodologia, serão descritas informações adicionais sobre escolha de ferramentas, linguagens e procedimentos do ambiente de trabalho em que o projeto foi desenvolvido. Na seção de conceitos básicos serão explorados conceitos fundamentais em que o trabalho se baseia. Na seção de trabalhos correlatos serão apresentados de maneira sucinta trabalhos similares a este que estão sendo desenvolvidos, apontando as principais diferenças e semelhanças entre eles. A seção de implementação, será separada em 4 partes, a parte de modelagem inicial, onde será apresentada a ideia e a modelagem inicial do dispositivo, a seção de componentes físicos, na qual serão descritos os componentes eletrônicos e físicos que compõem o projeto, a seção de programas e processo onde serão apresentadas as principais ferramentas usadas e programas desenvolvidos para a integração entre os componentes físicos e o monitoramento, e finalmente serão descritas as principais configurações no ambiente de monitoramento. Então serão apresentados os principais resultados e alguns exemplos práticos de análise de dados, junto de contribuições do projeto e ao final serão descritos possíveis direções para o aprimoramentos do projeto.

1.5 ESCOPO E METODOLOGIA

Espera-se com esse projeto, montar um hardware verificador de energia elétrica com um computador de placa única *raspberry pi* acoplado com um módulo de bateria inteligente (fonte de alimentação ininterrupta) e um modem USB para comunicação fora de banda (*out-of-band*). Além do hardware, tem-se como objetivo o desenvolvimento dos scripts e funcionalidades de verificação do estado da alimentação elétrica e bateria, comunicação de rede, e integração com os sistemas de monitoramento da organização.

O hardware final montado e com o software funcional, a priori, só necessita de um sinal de telefonia móvel no site remoto e fornecimento elétrico direto da transmissora de energia. É comum, em *datacenters*, o uso de geradores e *Nobreaks* para manter o fornecimento de energia elétrica em uma eventual falha da distribuidora de energia elétrica, então necessita-se a atenção de conectar o dispositivo direto na rede transmissora, porque, ele não será funcional para validar quedas de energias na concessionária se ele estiver conectado em um gerador ou bateria.

O dispositivo utiliza o paradigma de rede baseado em edge-computing, onde os processos executam com proximidade maior a fonte de dados [12], ou seja, a coleta e processamento de dados é feita em sua maioria no dispositivo Raspberry Pi, desse modo não existem custos adicionais de processamento em nuvem e reduz o tempo de atraso entre o processamento e envio da informação, além de facilitar a implementação.

No passado a REMEP-FLN adquiriu mini computadores para outros projetos, com o surgimento da ideia de um hardware IoT verificador de energia elétrica, começou-se a desenvolver primeiros protótipos desta solução de monitoramento de energia elétrica, e a versão final é a implementada neste trabalho.

Este trabalho será feito em conjunto com um projeto em andamento no PoP-SC/RNP e REMEP-FLN, o desenvolvimento será responsabilidade do aluno enquanto a aprovação do projeto conforme o proposto, o acompanhamento e os eventuais trâmites internos da organização serão responsabilidade do membro da banca avaliadora do trabalho na concedente, Rodrigo Pescador. Deseja-se usar a ferramenta ***git*** para versionamento do código desenvolvido e controle do repositório de códigos e documentação do projeto.

Diante das diversas linguagens disponíveis no mercado hoje em dia, foi escolhido Python como a principal linguagem de programação deste projeto, visto que é uma ferramenta eficaz para scripts e automação, além de ser amplamente adotada dentro do POP-SC/RNP, facilitando a manutenção do código desenvolvido. A ferramenta escolhida para o monitoramento dos dados enviados foi o Zabbix por ser uma das ferramentas de

monitoramento mais completas disponíveis no mercado, sendo simples e flexível, possibilitando a integração do sistema desenvolvido com todo o monitoramento da infraestrutura da Rede Metropolitana de Florianópolis e da Rede do Ponto de Presença da Rede Nacional de Ensino e Pesquisa de Santa Catarina.

2. CONCEITOS BÁSICOS

2.1 INTERNET DAS COISAS (IOT - *INTERNET OF THINGS)*

O termo é utilizado com o fim de se referir aos dispositivos/objetos em casas, empresas e cidades que estão conectados à Internet, geralmente via conexões de rádio de baixo custo e baixa capacidade de banda passante. Além disso, os dispositivos muitas vezes podem processar dados, também com limitações. Desse modo, podem executar tarefas específicas remotamente. No caso deste trabalho existe processamento na borda e chamamos esse paradigma de *IoT edge-computing*. Exemplos comuns de aplicações IoT são: geladeiras que se comunicam diretamente com mercados e solicitam a reposição de produtos sem a necessidade de intervenção de seus donos, carros que dirigem sozinhos pelas ruas a um determinado destino, prédios que possuem informação de onde estão localizadas, pessoas que são capazes de analisar e controlar diferentes parâmetros do ambiente. Outro exemplo de aplicação do conceito de Internet das Coisas (IoT) são dispositivos de baixo custo para monitorar ambientes remotos, ou enviar dados de ambiente para um gerente. Os requisitos comuns de gerência de redes IoT de baixa potência são: Escalabilidade, Tolerância a falhas, Qualidade de Serviço (QoS), eficiência energética, segurança e auto-configuração.

2.2 GERÊNCIA DE REDES

O termo IoT (*Internet of Things*) ou Internet das Coisas, em português, vêm recebendo notoriedade ao longo da última década. O desenvolvimento de dispositivos IoT conta com tecnologias de estado da arte e a sua gerência entra como destaque na área de gerência de redes. O uso de dispositivos inteligentes como sensores inteligentes ou agentes atuadores causa diversos problemas de performance na rede, e a gerência de redes eficiente para esses dispositivos é fundamental para uma boa performance (ABOUBAKAR, 2021)[5].

O processo de administração e gerência de redes é necessário para manutenção das redes de computadores. Inclui a provisão de infraestrutura de rede para conectividade dos usuários, análise e monitoramento de incidentes, gerenciamento de segurança e manutenção da qualidade de serviço (QoS). Existem diversos softwares de gerenciamento de rede no mercado e geralmente são administrados por times de operações de redes ou engenheiros de redes.

O *Simple Network Management Protocol* (SNMP), é um protocolo de gerência de redes, considerado o padrão de mercado usado amplamente em praticamente todas áreas que necessitam de uma rede gerenciada. É considerado o “protocolo padrão da internet TCP/IP”. É comum encontrar switches, roteadores, máquinas e até dispositivos IoT de baixo custo que já vêm com suporte SNMP, facilitando a configuração e gerência desses dispositivos. De forma sucinta, é um protocolo que contém uma base de dados e troca mensagens entre processos gerenciados e o processo gerente.

No caso deste trabalho usaremos o Zabbix como ferramenta de gerência de redes. O Zabbix, é uma ferramenta de software de código aberto para monitorar a infraestrutura de TI, como redes, servidores, máquinas virtuais e serviços em nuvem. Ele será usado para coletar as métricas que serão analisadas. Vale destacar que será utilizado o utilitário Zabbix Sender, um utilitário comumente usado em scripts que desejam enviar dados periódicos, aplicando perfeitamente para o caso deste trabalho de monitoramento de ambientes remotos.

2.3 ONDAS DE RÁDIO

Heinrich Hertz foi um físico alemão que foi responsável pela comprovação científica das ondas eletromagnéticas. A radiação eletromagnética é uma mudança em fase de campos elétricos e magnéticos que se auto sustentam. As ondas de rádio são um sub espectro da radiação eletromagnética que possui uma frequência baixa (comprimento de onda grande) comparado com a radiação infravermelha, que é emitida pelo corpo humano, podemos visualizar na Figura 1 abaixo a faixa de frequência dessas ondas e que elas são usadas para transmissão de dados em redes TCP/IP. Como as ondas de rádio são facilmente transmitidas pela atmosfera e não causam danos comprovados para o corpo humano, e essas características são ideais para transmissão de dados, elas são usadas para telecomunicação, inclusive em dispositivos de Internet das Coisas (IoT).

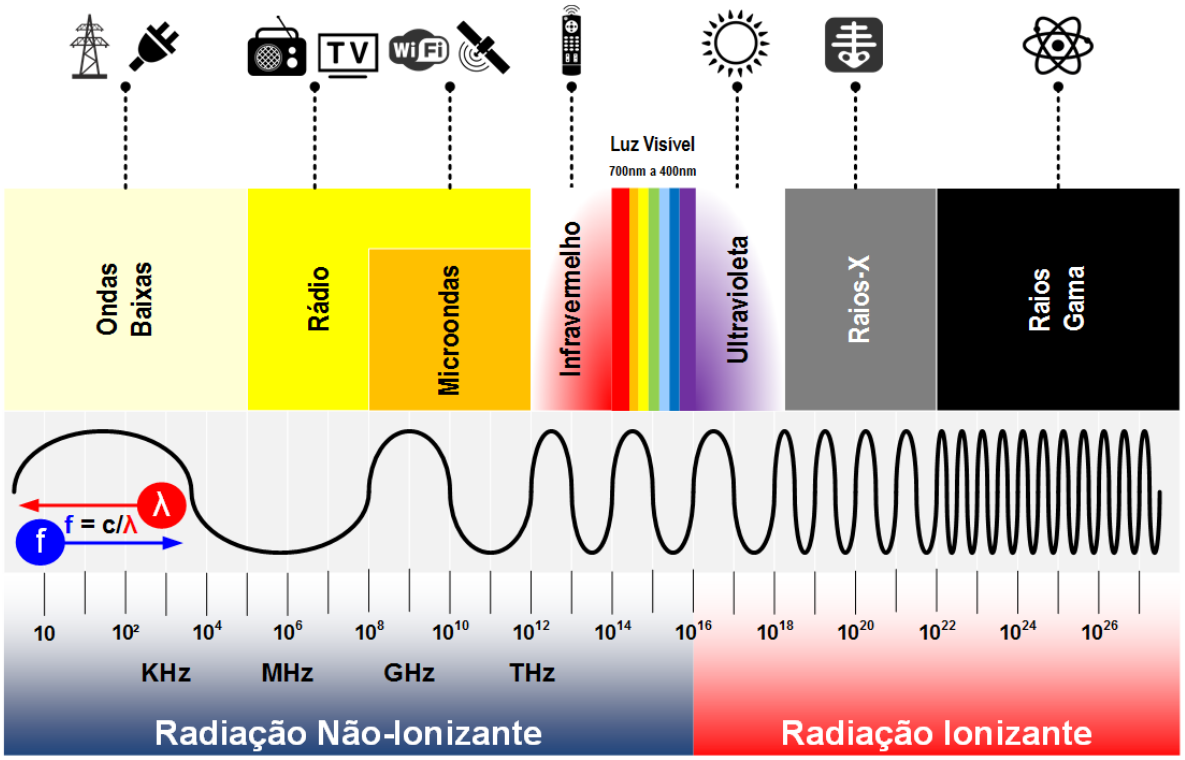

Figura 3. Espectro Eletromagnético [1].

2.4 REDES TCP/IP VIA ONDAS DE RÁDIO

Com o advento da internet, foi necessário o desenvolvimento de novas tecnologias para transmissão de dados, incluindo a transmissão de informações como e-mails, vídeos, áudio e dados via telefone celular. Desse modo, houve um estímulo para o avanço das tecnologias de transmissão de dados via ondas de rádio e visualizamos as gerações de telefonia móvel sendo implantadas, como o 2G, 3G, 4G e 5G. Hoje, é possível transmitir dados com a tecnologia 5G na ordem de bilhões de bits por segundo. A transmissão dos dados se dá via torres de estação de base celular que combinam transmissão e recepção de dados que sofrem modulação e demodulação. Essa infraestrutura de redes de telefonia móvel é indispensável para a conexão de dispositivos IoT remotos que se comunicam via rádio. Ainda nesta seção serão explicados três conceitos importantes para redes TCP/IP via infraestrutura de telefonia móvel, os conceitos de GSM, UMTS e GPRS.

1. GSM: Global System for Mobile Communications (GSM) é um padrão desenvolvido para descrever redes de telecomunicações 2G, é uma tecnologia antiga, sendo empregada pela primeira vez na Finlândia em 1992 [3]. O advento principal foi a mudança de transmissão em analógico para transmissão digital de dados, assim possibilitando o surgimento de novas tecnologias como comutação de pacotes.
2. GPRS: Essa tecnologia, o General Packet Radio Service (GPRS), permite a transmissão de dados por comutação de pacotes, ou seja, no padrão TCP/IP da internet. Diferentemente da comutação de circuitos, nesse tipo de tecnologia não é necessário alocar os recursos para transmissão de dados antes de estabelecer conexão com outro host, dedicando esses recursos por toda duração da comunicação.
3. UMTS: Universal Mobile Telecommunications System (UMTS), popularmente conhecido como 3G é um novo padrão GSM que é mantido pelo 3GPP (3rd Generation Partnership Project). Os principais avanços nessa tecnologia foi o uso de W-CDMA (code-division multiple access), uma tecnologia de interface de rádio que provê velocidades de dados superiores às tecnologias legado, assim fornecendo mais eficiência e permitindo a transmissão de maior banda passante. Além disso também surge aqui o conceito autenticação de usuários via cartão SIM, posteriormente explicada no trabalho.

3. TRABALHOS CORRELATOS

Foi realizada algumas pesquisas baseadas em palavras chave na ferramenta de pesquisa Google Acadêmico (*Google Scholar*), que é um mecanismo virtual de pesquisa

gratuito que organiza textos acadêmicos em diferentes formatos. A primeira busca foi a busca do termo chave IoT, resultando em aproximadamente 1.410.000 resultados de textos acadêmicos, mostrando que o tema de IoT é um tema de pesquisa relevante, Ao adicionar a palavra chave "Energy Availability", que significa disponibilidade de energia, o resultado desce para aproximadamente 2.390 resultados, mostrando o grau de especificidade da solução. Ao adicionar as palavras chave "Monitoring", que significa monitoramento e a palavra "Raspberry Pi" que é o dispositivo principal usado para basear a solução, os resultados descem para aproximadamente 1.690 e 168 resultados, respectivamente. Foram selecionados alguns trabalhos da área de IoT (Internet das Coisas) com propostas similares em diferentes aspectos, que serão discutidos nas próximas subseções.

PALAVRAS CHAVE	RESULTADOS
IoT	1.410.000
IoT, Energy Availability	2.390
IoT, Energy Availability, Monitoring	1.690
IoT, Energy Availability, Monitoring, Raspberry Pi	168

Tabela 1. Resultados da Revisão de Literatura.

3.1 IOT POWER MONITORING SYSTEM FOR SMART ENVIRONMENTS

Essa publicação, escrita por Diogo Santos (Instituto Federal de Lisboa) e João C. Ferreira (Instituto de Novas Tecnologias), descreve o desenvolvimento de um sistema IoT LoRa (*Long Range*) para monitorar o consumo de energia. A meta deste sistema é comunicar informações em tempo real de consumo de energia e ajudar a identificar desperdício de energia. A solução usa Arduinos como sensores e Raspberry Pi como servidor de aplicação. A comunicação usada é LoRa. A solução é baseada em *edge-computing* e foca também em fácil instalação sem limitações para ela ser usada em diferentes situações, tanto para grandes consumidores como para consumidores pequenos. A Figura 4 mostra a solução desenvolvida nesta publicação. Uma das maiores dificuldades que eles tiveram foi a falta de espaço nas placas elétricas. A topologia do trabalho em questão é bem diferente do trabalho atual de monitoramento, porque nesse caso eles usam um Raspberry Pi para receber dados e guardam eles em um MariaDB, o servidor do Raspberry é responsável por calibrar o relógio de cada dispositivo. No nosso caso, armazenamos no servidor do Zabbix os dados, que é um servidor central com muito mais capacidade que um raspberry pi, além disso usamos o módulo de temporização explicitado na seção de Hardware para realizar a sincronia dos relógios. Eles

também usam antenas e ondas de rádio para comunicações, porém usam o protocolo de comunicação *LoRaWAN*, no nosso caso usamos o protocolo de aplicação do Zabbix Trapper [4], que cria uma conexão TCP com o servidor central para o envio e recebimento de dados em formato JSON.

Pontos-chave: Monitora métricas de energia, usa ondas de rádio para transmissão de dados de longa distância, não usa Raspberry Pi, usa uma solução baseada em Arduino.

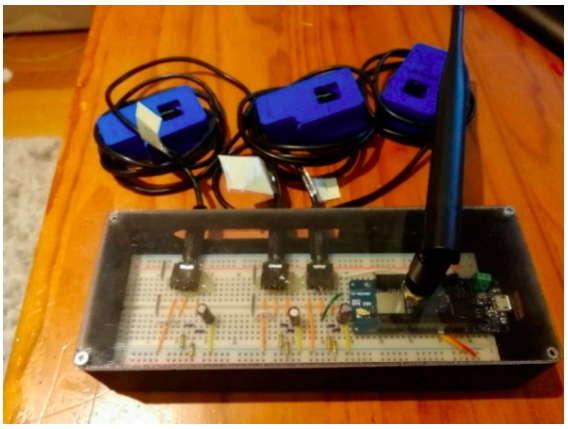

Figura 4. Solução final da publicação pronta para implementação [10].

3.2 AN IOT BASED PATIENT MONITORING SYSTEM USING RASPBERRY PI

No contexto revolucionário da Internet das Coisas, essa publicação é voltada para o setor de assistência médica, mais especificamente no monitoramento em tempo real de pacientes, com o fim de reduzir o erro humano através da coleta automática de dados. A solução procura, no ambiente médico, reduzir custos e administração de recursos ao monitorar métricas como: temperatura corporal, taxa de respiração, batimento cardíaco e movimento do corpo. A figura 5 mostra o setup de conexão para monitoramento de pacientes da solução baseada em raspberry pi e sensores que usam interfaces de dados para comunicação com o dispositivo IoT. O trabalho é similar no quesito de monitorar métricas de algum ambiente remoto, todavia esse trabalho está voltado para área da saúde e focado na integração de sensores de diferentes métricas médicas junto ao dispositivo IoT e não prevê integrações com sistemas de monitoramento ou envio de dados através de uma rede IoT via ondas de rádio, tendo em mente a escalabilidade e resiliência do dispositivo.

Pontos-chave: Monitora métricas de pacientes médicos, usa interfaces locais (Ethernet/Wi-fi/USB) para transmissão de dados, usa Raspberry Pi, usa sensores conectados na placa do Raspberry Pi.

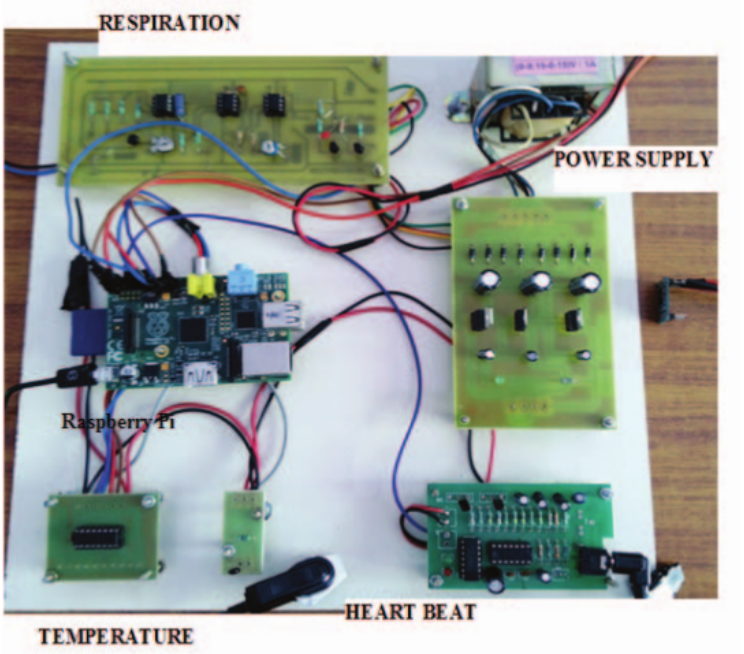

Figura 5. Solução desenvolvida no trabalho [6].

3.3 AN AUTONOMOUS LOW-POWER LORA-BASED FLOOD-MONITORING SYSTEM

Esse artigo desenvolvido no contexto de redes de sensores wireless em ambiente de Internet das Coisas, apresenta o design, a implementação e resultados de testes de um sistema de monitoramento de enchentes, usando a tecnologia LoRa para transmissão de dados e testado em um ambiente real. As informações são armazenadas em um dispositivo equipado com sensores e um microcontrolador, conectados em um módulo LoRa para comunicação e então é processado em uma estrutura web que implementa os alarmes no caso de uma enchente. O Trabalho é parecido em alguns aspectos arquiteturais, como o de um dispositivo remoto monitorando algum estado em um outro ambiente, então envia informações e baseado nelas dispara alarmes ou não. Esse dispositivo não usa nem raspberry pi nem arduino, mas uma placa customizada, como podemos visualizar na Figura 6. Esse artigo utiliza o módulo de temporização RTC DS3231, usado também nesse trabalho.

Pontos-chave: Usa sensor resistivo, usa protocolo de comunicação LoRa, usa uma placa customizada com microcontrolador.

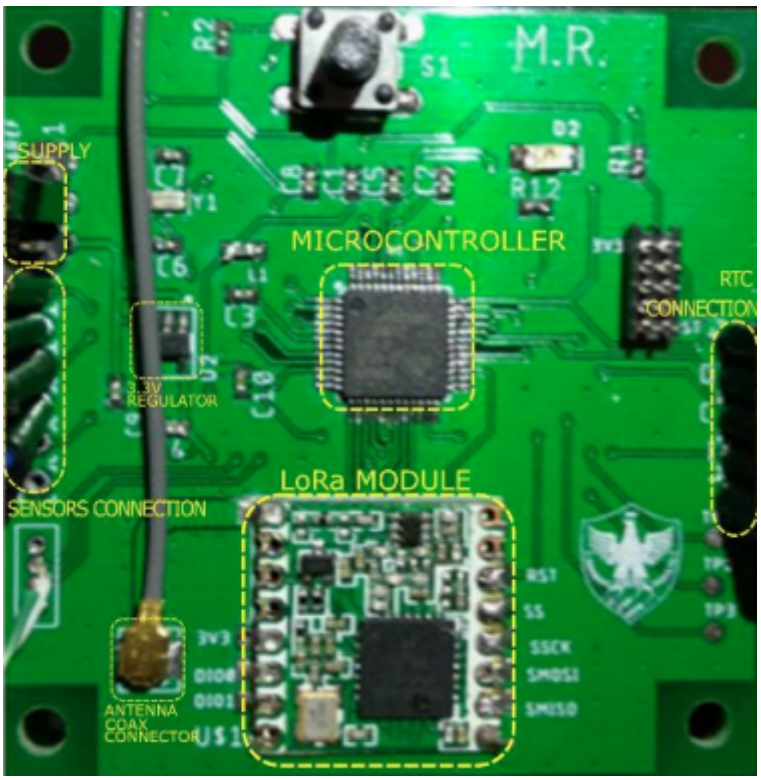

Figura 6. Solução Implementada do Trabalho [11].

3.4 SMART IRRIGATION SYSTEM WITH SOLAR POWER AND GSM TECHNOLOGY.

O trabalho Smart Irrigation System with Solar Power and GSM Technology [13] desenvolve-se um sistema que automatiza o processo de irrigação com o auxílio de energia solar. O sistema é capaz de otimizar o uso de água baseado em diferentes parâmetros (previsão do tempo, umidade do solo, etc.). O operador é notificado das condições do solo e do motor via tecnologia GSM. A modelagem permite o acionamento e desligamento automático do motor baseado na demanda de água no campo. A tecnologia GSM é usada para atualizar o fazendeiro/operador sobre o acionamento ou desligamento do motor. A corrente contínua é fornecida pelo painel solar, armazenado em baterias e depois usa-se um inversor para transformar em corrente alternada. O microcontrolador usado é um Arduino, existem trabalhos com raspberry pi também [16].

Pontos-chave: Usa sensores e atuadores, utiliza tecnologia GSM para transmissão remota de dados, utiliza microcontrolador Arduino, energia limpa.

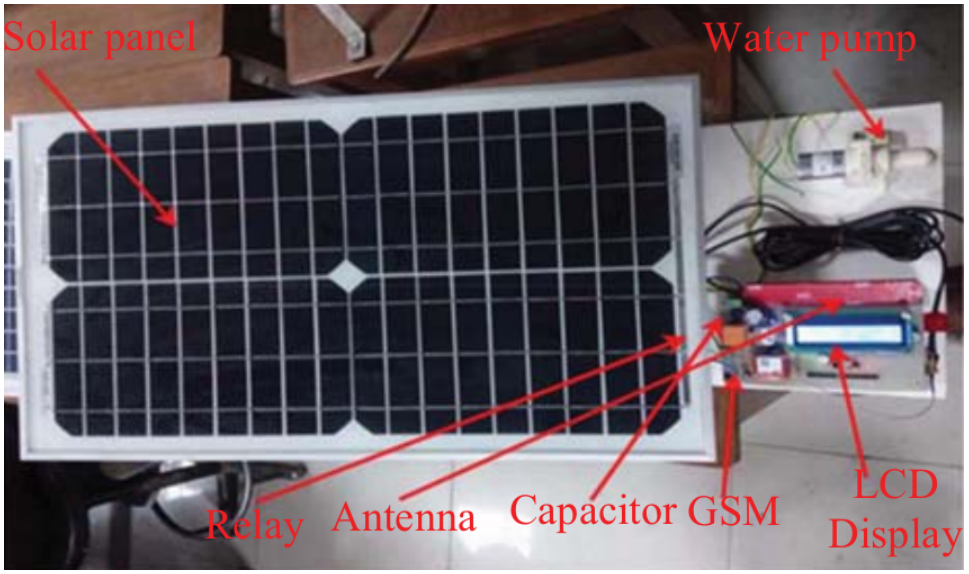

Figura 7. Solução IoT do Trabalho [13].

4. IMPLEMENTAÇÃO

4.1 MODELAGEM INICIAL

O modelo inicial pensado para a montagem do dispositivo físico está demonstrado na figura 8, que mostra um diagrama onde temos o computador de placa única acoplado em um módulo de bateria, sendo alimentado por um cabo de energia elétrica e com um modem USB ligado ao computador, tudo isso dentro de uma caixa protetora de plástico, também chamado de patola.

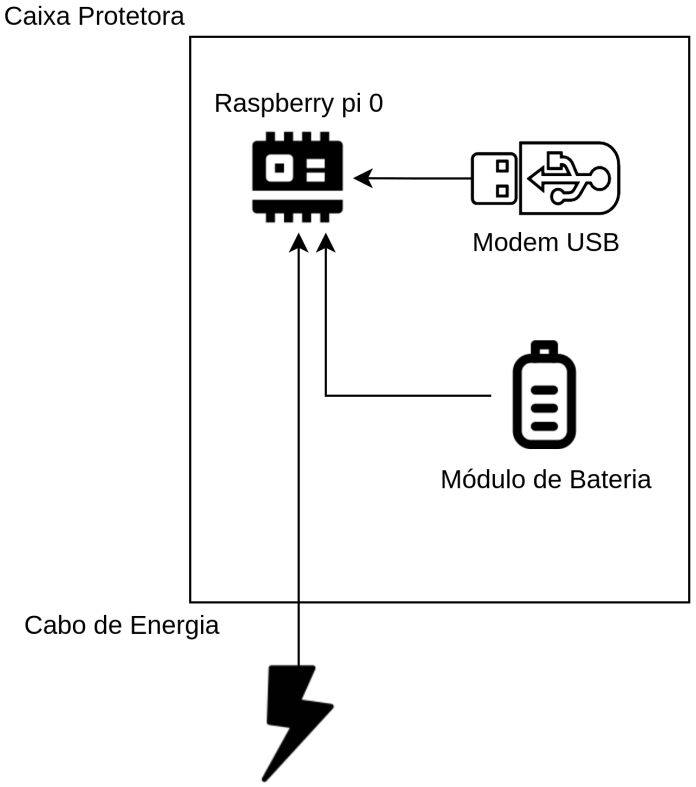

Figura 8. Modelo Inicial de Montagem.

A partir da modelagem inicial, iniciamos a montagem dos componentes na caixa protetora de eletrônicos e obtemos um resultado similar ao da Figura 9, onde podemos visualizar a localização de cada componente na patola. Nessa figura, não está instalado o módulo de sincronização de tempo RTC e o LED opcional indicador do estado da conectividade WAN.

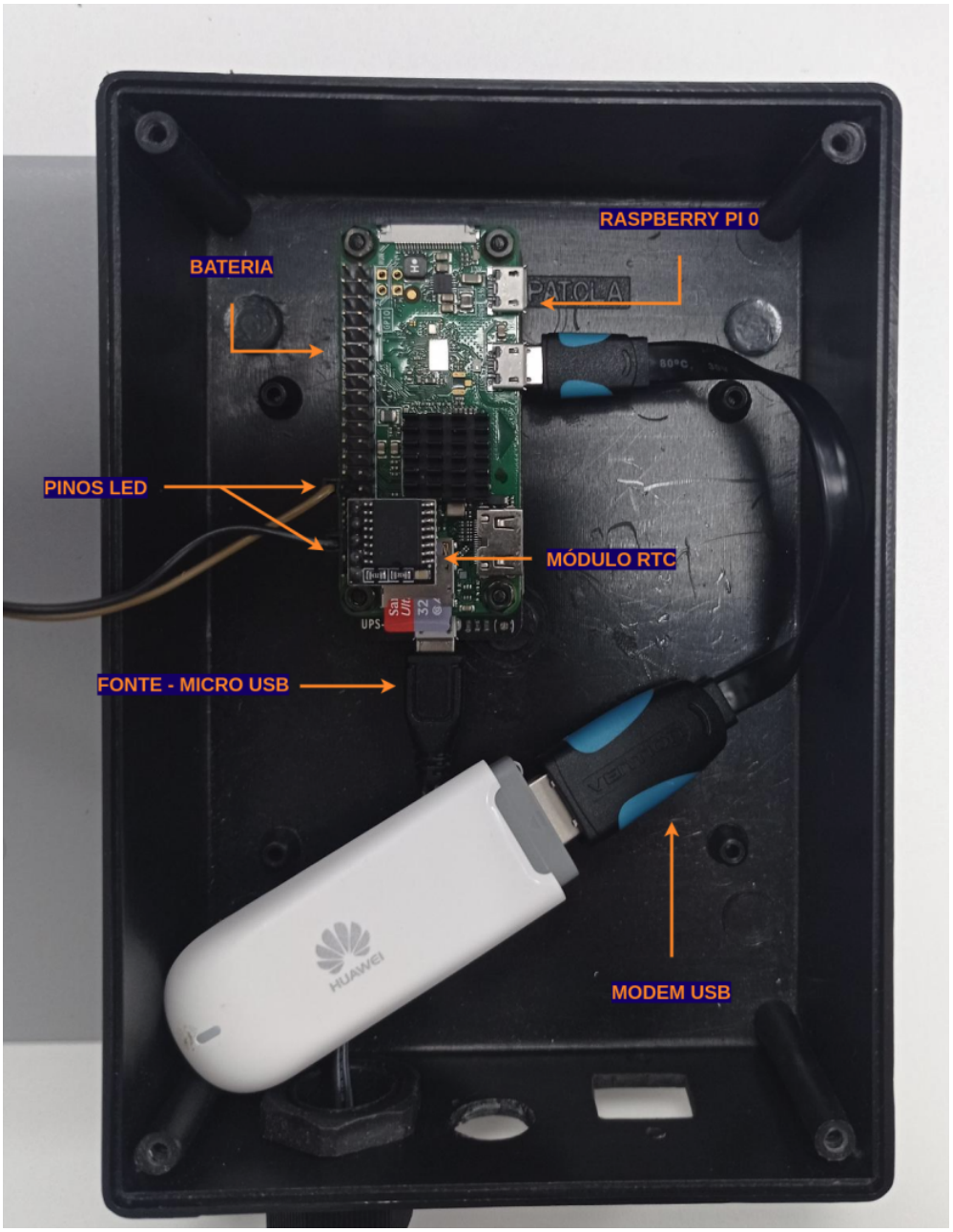

Figura 9. Montagem Inicial dos componentes.

4.2 COMPONENTES FÍSICOS ELETRÔNICOS (HARDWARE)

4.2.1 CAIXA DE MONTAGEM

A caixa de montagem será uma caixa plástica patola PB-115 preta e com parafusos. As caixas patolas são geralmente utilizadas como gabinetes para eletrônicos. A finalidade principal da caixa é a proteção dos componentes modulares que serão estruturados dentro dela. As dimensões da caixa escolhida são, conforme visualizamos na Figura 10.:

- Altura: 50mm.
- Largura: 112mm.
- Comprimento: 160mm.

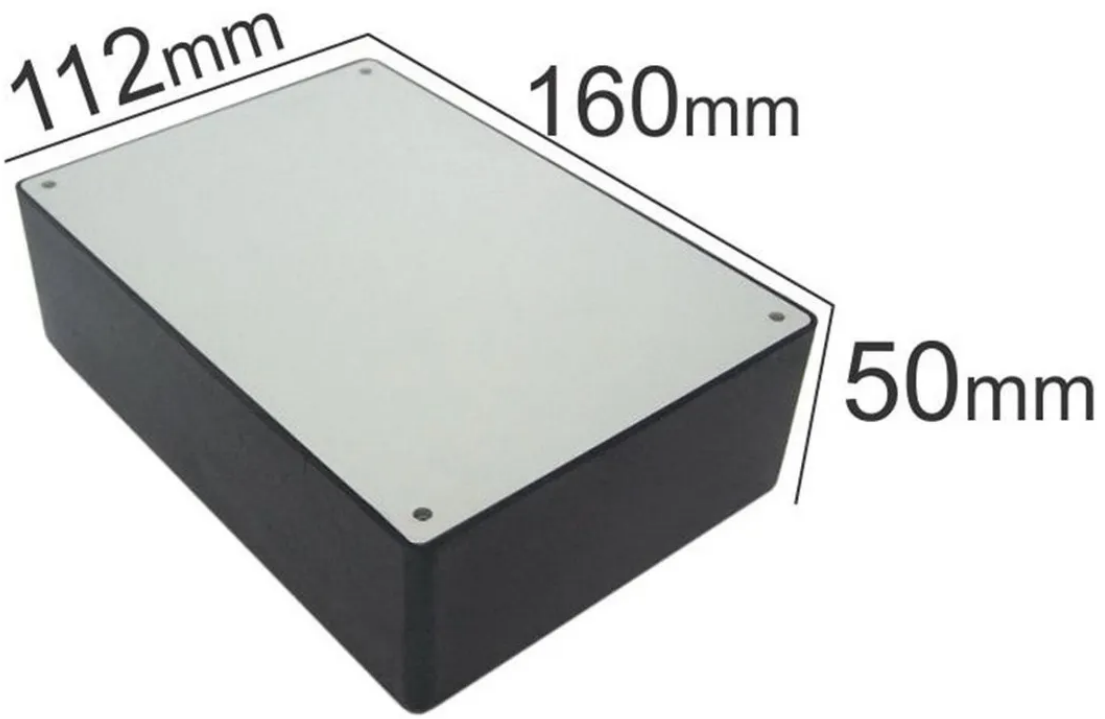

Figura 10. Caixa de Montagem.

4.2.2 ACESSÓRIOS

Classificamos como acessórios a fonte de alimentação micro-usb (Figura 11) e o adaptador micro-USB ➢ USB (Figura 12). A fonte de alimentação serve para fornecer energia elétrica para o computador raspberry pi, permitindo ele a realizar as computações necessárias para o monitoramento IoT. O adaptador micro-USB para USB serve para conectar o terminal micro-usb na placa do Raspberry pi e conectar um modem USB no terminal USB. Foi necessário escolher uma fonte de alimentação homologada porque é comum ocorrer problemas de alimentação elétrica no raspberry pi ao usar qualquer tipo de fonte, a fonte escolhida é a fonte do modelo "**STONTRONICS** Power Supply for Raspberry Pi, 5V, 2.5A, Multi", de 5V e 2.5A.

Figura 11. Fonte de alimentação STONTRONICS.

Figura 12. Adaptador Micro-USB para USB.

4.2.3 MODEM USB - HUAWEI

Um Modem 3G USB Huawei E303C/E303 deve ser conectado ao computador de placa única para fornecer conectividade a redes móveis ao dispositivo. Posteriormente, será descrito o script de controle de conexão responsável por conectar o dispositivo na rede e manter a conexão saudável sem intervenção humana. O Modem tem suporte para tecnologias 2G e 3G, a tecnologia de transmissão a ser escolhida depende dos sinais do site remoto monitorado.

Figura 13. Modem USB

4.2.4 CARTÃO SIM

O cartão SIM (Figura 14) é popularmente conhecido como *chip*, ele é um circuito responsável por identificar o dispositivo em uma rede móvel GSM e é necessário para o funcionamento completo do Modem, ou seja, para a conexão do dispositivo na rede móvel. Ele tem parâmetros como ICCID, IMSI, Chave de autenticação entre outros. A operadora ARQIA fornecerá os chips para este projeto IoT, com um plano especial para a RNP. A Arqia é uma empresa de telecomunicações em São Paulo especializada em soluções IoT. Será usada uma APN privada que permite o tráfego rápido e prioritário para os pacotes de dados que serão enviados para o ambiente central de monitoramento através de ondas de rádio. A APN é o nome do ponto de acesso do *gateway* que leva o dispositivo para a rede pública.

Figura 14. Cartão SIM da ARQIA.

4.2.5 RASPBERRY PI ZERO

O Raspberry Pi Zero, é o componente principal desse trabalho. Ele é um mini computador de placa única fabricado pela *Raspberry Pi Foundation*, criada com o intuito de estimular o estudo na área de Ciências da Computação. Esse modelo de mini-computador é uma versão de computador com 512MB de memória RAM, processador com frequência de 1GHz e suporte para conexão wi-fi. Ele possui pinos programáveis GPIO (General Purpose Input/Output) que são utilizados para prover uma interface entre dispositivos conectados nas portas e o computador. O dispositivo é alimentado por uma porta micro-usb e também recebe dados via porta micro-usb. As dimensões do computador são as seguintes:

- Comprimento: 65mm.
- Largura: 30mm.
- Altura: 5,4mm.

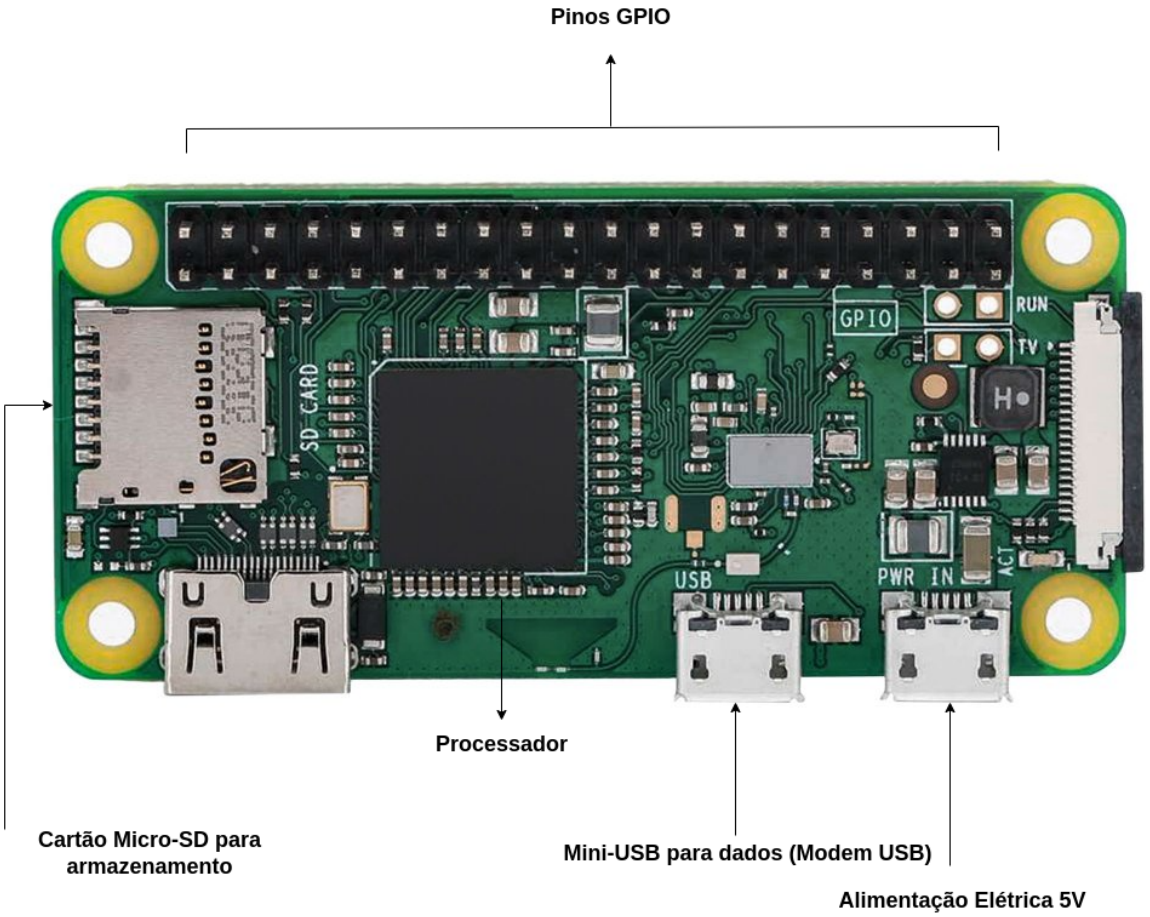

Figura 15. Raspberry Pi Zero.

Na Figura 15, podemos visualizar os principais componentes usados, entre eles o slot de cartão SD onde iremos inserir o cartão SD com o sistema operacional e dados necessários para execução dos programas desenvolvidos, o processador que executará os scripts, o mini-usb de alimentação elétrica, que não será usado porque usaremos um módulo de bateria para fornecer energia, o mini-usb no qual o modem USB será conectado através do adaptador e os Pinos GPIO, onde iremos instalar um módulo de temporização e um LED indicador do estado da conexão com a rede de telefonia móvel.

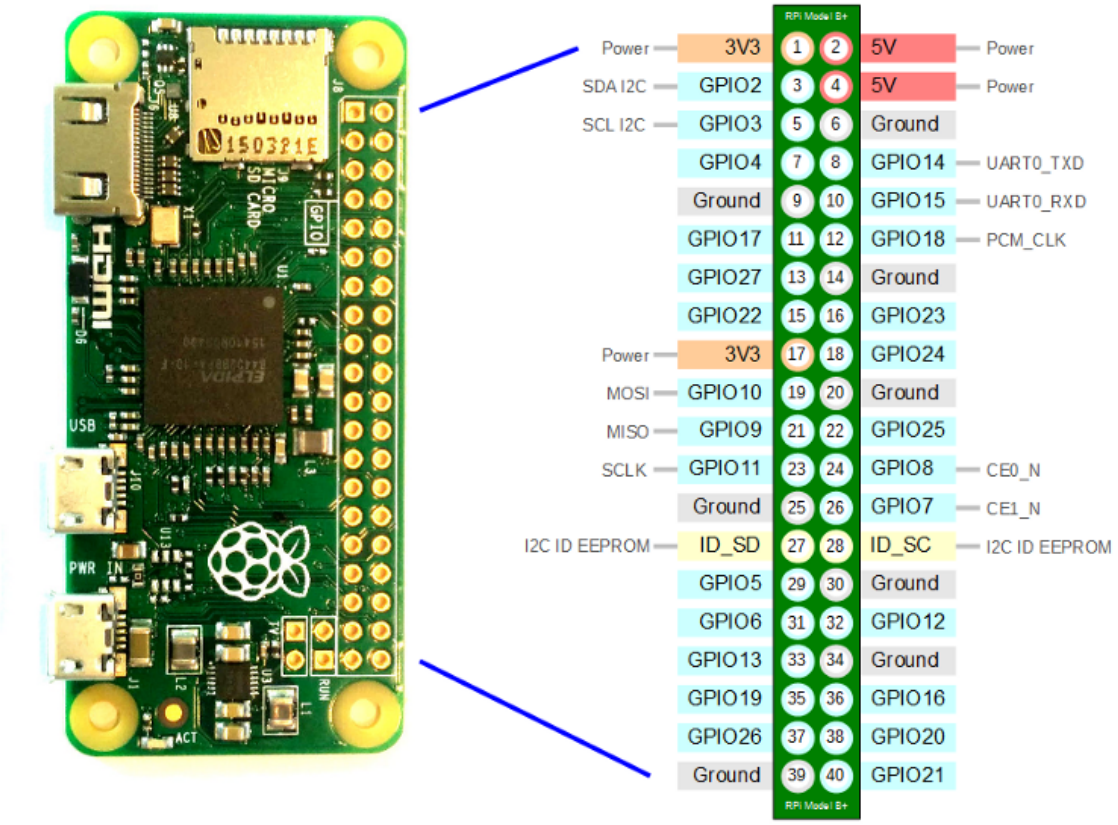

Figura 16. Tabela dos Pinos GPIO.[1]

Na Figura 16, podemos visualizar uma tabela extraída da web [1] na qual é possível analisar as funcionalidades atribuidas para cada um dos 40 pinos GPIO. Usaremos os pino 1, 3, 5, 7, 9 para o módulo de temporização e os pinos 6 e 12 para o led indicador. Essa tabela é particularmente útil para nós sabermos qual pino GPIO executa qual função. Existe um website interativo que é útil para qualquer usuário que deseja trabalhar com os pinos GPIO do raspberry pi que é o https://pinout.xyz/. Nesse website podemos visualizar dinamicamente as funções de cada pino.

4.2.6 PLACA DE ENERGIA UPS LITE V1.2 PARA RASPBERRY PI ZERO

Esse módulo de bateria para Raspberry Pi Zero (Figura 17) é uma fonte de alimentação ininterrupta (*UPS*) usada com o fim de fornecer alimentação elétrica para o dispositivo em caso de indisponibilidade elétrica no site remoto que está sendo monitorado. A placa contém uma bateria com capacidade de 1000mAh e dura em média entre 2h e 6h, dependendo da intensidade do uso e do estado da bateria. A corrente para carregamento da bateria é de 400mA, a tensão de saída é 5V com margem de +- 0.1V, a corrente de saída é de 1.3A e 5V (quando não existe adaptador externo) E 2A com 5V ao inserir um adaptador externo. As dimensões da placa são:

- Comprimento: 65mm
- Largura: 24mm

Figura 17. Placa de Bateria UPS para Raspberry Pi.

4.2.7 MÓDULO RTC DS3231

Esse módulo é responsável por fornecer um relógio em tempo real de precisão e usamos ele para garantir a sincronia de tempo do sistema local mesmo quando o dispositivo é desligado por longos períodos. O módulo RTC DS3231 utiliza a interface I2C para comunicação por meio dos pinos 1 (3.3V), 3 (SDA), 5 (SCL) e 9 (GND) da GPIO do Raspberry. Ele possui uma pequena bateria e realiza a comunicação com o dispositivo via serial I2C. A Tensão de operação é de 2.3V-5.5V, suas dimensões são:

- Comprimento: 18mm
- Largura: 14mm
- Altura: 13mm

Figura 18. Módulo de Temporização RTC.

4.2.8 LED (OPCIONAL)

O LED é opcional e pode ser usado através de um serviço desenvolvido para indicar o estado da conexão do modem USB com as redes de telefonia móvel, ou seja, se estiver conectado na rede o LED fica acesso, e caso não esteja conectado na rede o LED fica apagado. Facilitando a operação do dispositivo em ambientes remotos em caso de precisar consultar a situação do estado da conexão. Para instalar o LED, usaremos cabos *jumper* macho-para-fêmea e conectamos no polo negativo (cátodo) do LED o pino 6 (GND) no GPIO e o polo positivo (ânodo) do LED no pino 12 (GPIO 18 - PCM Clock) do GPIO. O polo negativo do LED é identificado por uma borda reta no bulbo de luz. Dependendo do tipo de LED, o polo positivo pode ser identificado pelo comprimento dos polos condutores, o maior sendo o ânodo.

4.3 PROGRAMAS E PROCESSOS (*SOFTWARE*)

Nessa seção será comentado os principais aspectos relacionados aos principais módulos de software desenvolvido, principalmente na linguagem Python, com o fim de executar as tarefas necessárias e abranger o escopo do trabalho definido anteriormente, compreendendo comentários sobre configuração, implementação e testes realizados. Os trechos de códigos principais que foram julgados importantes para a descrição do funcionamento do *software* estão explicitados com destaque da sintaxe da linguagem de programação escolhida. A tabela abaixo especifica quais são os módulos de implementação própria e quais são os módulos de terceiros, junto da finalidade de cada módulo.

NOME DO MÓDULO	IMPLEMENTAÇÃO	FINALIDADE
rebind.sh	Própria	Utilitário que simula a reconexão

		lógica do modem.
check_ac_in.py	Própria	Script de verificação do estado da corrente elétrica.
lte_dial.py	Própria	Script de controle de conexão WAN e gerência do Modem.
monitor_net.py	Própria	Utilitário de rede que monitora o consumo da banda trafegada.
led_controller.py	Própria	Controle do LED indicativo do estado da conectividade IoT.
periodic_status.py	Própria	Envio periódico de informações do ambiente através de um escalonador.
mmcli (ModemManager CLI)	Terceiros	Interface de linha de comando para gerência de dispositivos com comunicação via rádio, i.e. Modem USB.
nmcli (NetworkManager CLI)	Terceiros	Interface de linha de comando para gerência de redes no sistema operacional.

Tabela 2. Módulos de Software principais e suas finalidades.

4.3.1 MÓDULO *CHECK_AC_IN.PY*: EXAME PERIÓDICO DO ESTADO DA ALIMENTAÇÃO

O script check_ac_in.py foi um script desenvolvido em python que realiza a leitura do barramento I2C e o envio das mensagens de alteração de estado da alimentação de energia para o ambiente central.

Comportamento

O script irá periodicamente (tempo definido pelo usuário) realizar uma leitura dos pinos ***GPIO***, com essa leitura, ele poderá ser capaz de informar se há energia elétrica no local ou não. Podemos visualizar abaixo de maneira mais específica essa leitura, ou seja, se o pino 4 está em um estado **HIGH** (Nível lógico alto ou 3.3V) definimos uma variável que indica que está normal o fornecimento elétrico, caso o estado seja **LOW** (Nível lógico baixo ou 0V) definimos uma variável indicando que falta alimentação elétrica. Se ocorrer uma alteração do estado, tanto de (0 ➤ 1) ou (1 ➤ 0) será enviado dados para o ambiente central replicando a situação atual do fornecimento elétrico e do nível de carga da bateria do dispositivo. Existe um controle adicional de temporização no algoritmo que permite ao usuário definir se, em caso de uma falha elétrica ocorrer, o dispositivo envia dados mais frequentemente do que o

normal. Por exemplo, o usuário pode definir que deseja receber informações do local monitorado a cada 60 segundos em um caso de falha elétrica, e de 1 em 1 hora em um caso de alimentação normal.

```
if (GPIO.input(4) == GPIO.HIGH):
        ac_state_current = 1
        ac_state_display = 'OK: Power Adapter Plugged'

if (GPIO.input(4) == GPIO.LOW):
        ac_state_current = 0
        ac_state_display = 'WARNING: Power Adapter Unplugged'
```

Código 1. Leitura dos estados de fornecimento de corrente alternada no dispositivo.

Dentro desse script definimos a função ***zabbix_sender()***, que será responsável pelo envio das informações de ambiente para o Zabbix, a central de monitoramento. O script tenta, por padrão, 3 vezes enviar a informação para a central. O envio de informações se dá por meio da ferramenta *Zabbix Sender*, que é instalada nos dispositivos. A ferramenta *Zabbix Sender* estabelece uma conexão TCP com o servidor para o envio de dados, ou seja, usa um protocolo orientado a conexão com mecanismos para garantia de entrega, sequenciamento de pacotes e reenvio de informações. Todavia, caso ocorra um *timeout* TCP ou outra falha que impeça o envio da informação, será testado a conectividade da conexão WAN fazendo um teste de *ping* para o servidor Zabbix, caso não tenha conectividade, será executado um comando no Modem que força a desconexão e o posterior reset dessa conexão por outro script chamado ***lte_dial.py*** que terá seu comportamento descrito posteriormente nesta mesma seção. Após esse comando de desconexão o processo entra em estado ***sleep*** por ***ZABBIX_SLEEP*** segundos, o tempo padrão é de 5 minutos. O script de controle de conexão, responsável por restabelecer essa conexão leva, em média, de 2 a 3 minutos para restabelecer a conectividade WAN. Quando o processo volta a executar no processador, é esperado que a conexão esteja funcionando novamente e é feito uma nova tentativa de envio das informações. Esse processo é repetido, por padrão, 5 vezes ou até a informação ser enviada com sucesso para o servidor.

```
def zabbix_sender(args):
    for i in range (ZABBIX_SENDER_TRIES):
        if LOG_ZABBIX_SENDER_OUTPUT:
            with open(LOG_FILE, 'a') as logfile:
                logfile.write ('-- Try[{}/{}] ARGS: {}
--\n'.format(i+1, ZABBIX_SENDER_TRIES, args))
        completed_process_returncode = run_and_log(args)
        if completed_process_returncode == 0:
            break
        else:
            connectivity = ping_central()
            #log(f'connectivity result is {connectivity}')
            # if there is no wan connection, force lte_dial.py
```

```
reconnection
            if not connectivity:
                # force nmcli con down gsm-con
                disconnect_wan()
                # wait 5minutes; lte dial will lte dial will take
about 30 seconds to detect the failure and will take about 2-3 minutes
to restore the connection. Thus we wait about 5 minutes, in order to
make sure the next time we run zabbix_sender we will have an active
connection take about 30 seconds to detect the failure and will take
about 2-3 minutes to restore the connection. thus we wait about 5
minutes, in order to make sure the next time we run zabbix_sender we
will have an active connection
                time.sleep(ZABBIX_SLEEP)
```

Código 2. Envio de dados ao Zabbix.

Seção de configuração - Zabbix

O script terá uma seção de configuração onde irá definir o IP do servidor central, ou seja, o destinatário das informações de alteração de estado. o nome do Zabbix Host e as chaves dos itens de monitoramento, que nesse caso são a chave do estado de energia e a chave do nível de bateria do módulo de bateria acoplado.

Seção de configuração - Temporização, Logging e Outros.

Na segunda parte da seção de configuração teremos informações relevantes para a temporização do algoritmo, entre outras configurações menores. Podemos visualizar no código abaixo as variáveis de configuração e posteriormente uma descrição para cada uma delas.

```
# send info to server after PERIODIC_SEND_TIME seconds regardless of
ac state. 3600 = 1hour. 0 = don't send
PERIODIC_SEND_TIME=7200
# if ac fault occurs, send info to server faster
SEND_PERIODIC_INFO_WHEN_AC_FAULT=True
INTERVAL_PERIODIC_INFO_WHEN_AC_FAULT=1200
#round check interval (seconds) system will look for state changes
CHECK_PERIOD = 2
# ----- LOGGING -----
#log state changes to a file (true/false)
LOG_ON_STATE_CHANGE = True
# Log zabbix sender output
LOG_ZABBIX_SENDER_OUTPUT = True
#log file to save state changes
LOG_FILE = '/var/log/pimon.log'
# central server ip
PING_IP = '200.237.192.70'
# maximum amount of zabbix sender tries
ZABBIX_SENDER_TRIES = 5
# amount of zabbix sender sleep seconds
ZABBIX_SLEEP = 300
```

Código 3. Envio de dados ao Zabbix.

- PERIODIC_SEND_TIME: Variável inteira responsável por definir, em segundos, em quanto tempo a informação do estado de energia e da bateria será enviada para o ambiente de monitoramento independente de ter ocorrido ou não uma alteração no estado da energia. Intervalos maiores resultam em menos processamento e banda enviada, porém em menos cadência e continuidade das informações enviadas.
- SEND_PERIODIC_INFO_WHEN_AC_FAULT: Variável booleana responsável por indicar se em caso de uma falta de energia, o dispositivo deve enviar dados mais rapidamente de acordo com o tempo definido em outra variável.
- INTERVAL_PERIODIC_INFO_WHEN_AC_FAULT: Variável inteira responsável por definir, em segundos, o intervalo do envio de mensagens para a central de monitoramento em caso de falha elétrica. Intervalos maiores resultam em menos processamento e banda enviada, porém em menos cadência e continuidade das informações enviadas.
- CHECK_PERIOD: Variável inteira, definida em segundos, que determina o intervalo de verificação da alteração do estado de alimentação elétrica. Intervalos mais longos resultam em menos processamento porém mais demora na notificação do evento em caso de falha ou normalização.
- LOG_ON_STATE_CHANGE: Variável booleana, que define se o programa deve registrar informações de alteração de estado em LOG_FILE.
- LOG_ZABBIX_SENDER_OUTPUT: Variável booleana, que define se informações pertinentes a execução da função ***zabbix_sender()*** devem ser registradas no arquivo de log.
- LOG_FILE: String que define o caminho (*path*) do arquivo de log onde serão feito os registros desse programa, por padrão é em '/var/log/pimon.log'.
- PING_IP: String que define o endereço IPv4 do servidor que será usado para testar o estado da conexão na função ***zabbix_sender()***.
- ZABBIX_SENDER_TRIES: Variável inteira que define a quantidade máxima de tentativas de envio de informações de monitoramento na função ***zabbix_sender()***.
- ZABBIX_SLEEP: Variável inteira que define, em segundos, a quantidade de tempo que o processo fica inativo após enviar um comando de desconexão do modem.

4.3.2 SCRIPT *REBIND.SH*: AUTOMAÇÃO EM *BASH* PARA SIMULAR UMA RECONEXÃO DO USB.

O script rebind.sh foi um script desenvolvido em bash que realiza a reconexão lógica do modem USB. Ao executar o script é atribuído o valor ***usb1*** para uma variável que será

usada como caminho para o Modem, assim temos as funções **bind_usb()** e **unbind_usb()** que irão fazer a conexão e desconexão do USB, respectivamente. Então o script executa primeiro a função que desconecta, espera 5 segundos e então executa a função que conecta o USB. Esse script é usado no módulo ***lte_dial.py***.

```
#!/bin/bash
port="usb1" # replace 1 with the actual bus number as shown in lsusb
-t to reset the usb hub
#port="1-1.3" # as shown by lsusb -t: {bus}-{port}(.{subport})

bind_usb() {
  echo "$1" >/sys/bus/usb/drivers/usb/bind
}

unbind_usb() {
  echo "$1" >/sys/bus/usb/drivers/usb/unbind
}

unbind_usb "$port"
sleep 5 # delay
bind_usb "$port"
```

Código 4. *Rebind* do USB.

4.3.3 MÓDULO *LTE_DIAL.PY*: SCRIPT DE CONTROLE DE CONEXÃO WAN E COMANDOS NO MODEM

O script lte_dial.py foi um script desenvolvido em python que realiza comandos no modem e controles lógicos do estado da conexão WAN, restabelecendo a conectividade com a internet sem nenhuma intervenção humana, visto que a gerência dos equipamentos dependendo do local instalado fica inviável. O script foi desenvolvido com o objetivo de ser tolerante a falhas e resiliente, de modo que em qualquer evento na rede o dispositivo consiga se reconectar na rede WAN.

Esse módulo é o maior e de maior complexidade de todos módulos desenvolvidos, visto que existem muitas variáveis que determinam o funcionamento deste programa, como a disponibilidade da rede da operadora provedora de telefonia móvel ou conexão IoT, tempos de espera e resposta do sistema operacional, formatos de resposta do operacional, rotas adicionadas pelo sistema, entre outras variáveis.

Comportamento

O script executa em um loop de tentativas sucessivas de conexão na rede 2G ou 3G, definido pelo operador na seção de configuração. Primeiro as instruções executam uma busca nos caminhos do sistema operacional para achar o modem, usando o módulo ***mmcli***, que será posteriormente descrito. Após realizar o tratamento da *string* de saída do módulo e obter o

caminho para o modem, obtemos a porta primária onde o modem está conectado e dependo do seu valor (cdc-wdm0, ttyUSB2). Ações diferentes são realizadas:

- **cdc-wdm0:** Como esse tipo de reconhecimento de porta os perfis de conexões do não funcionam, usamos o módulo *mmcli* para resetar o modem através do comando ***"mmcli -r -m caminho_do_modem"***.
- **ttyUSB2:** Registramos o valor de **modem.3gpp.registration-state**, ou seja, o estado de registro do modem e verificamos se ele é **'roaming'** ou **'searching'**. No caso de ser **'searching'**, registramos o modem usando o comando ***"mmcli -m caminho_do_modem --3gpp-register-home"***.

Após esse processo, nós buscamos o estado genérico do modem através do item de acesso no módulo *mmcli* de nome **modem.generic.state**. Após realizar o tratamento da string de saída do ModemManager (*mmcli*). Verificamos se o estado do model é igual a **'failed'**. Em caso afirmativo, lançamos uma exceção do tipo ModemFailedState, que será tratada na seção de tratamento de exceções. Em caso negativo, seguimos para a conexão do Modem na porta primária encontrada. usamos o comando ***"nmcli device connect porta_primaria"*** do NetworkManager, módulo do sistema operacional que também será descrito posteriormente. Caso ocorra algum erro nesse processo, lançamos uma exceção do tipo ModemFailedConnection. Então prosseguimos para uma rodada de testes do estado da conexão, com o fim de verificar se a conexão estabelecida está funcionando conforme o esperado, ou seja, a rede está operacional e pronta para transmissão de dados. Buscamos o *gateway* definido dinamicamente pelo ModemManager ao subir a conexão. Se o gateway não é achado ou o seu valor foi determinado como '0.0.0.0' lançamos uma exceção do tipo GatewayNotFound. Depois esperamos um tempo definido pelo usuário, com o fim de esperar o *setup* das rotas pelo módulo do sistema operacional NetworkManager. Após as rotas terem sido atribuidas para o dispositivo, deletamos a rota default via interface ***wwan0*** (modem USB) e adicionamos apenas uma rota para o servidor central de monitoramento, isso foi feito para evitar qualquer tráfego de ida ou de volta para algum dispositivo conectado na internet que não seja os endereços IP's definidos pelo usuário, necessárias para o monitoramento. Depois checamos a conectividade com o servidor central de maneira inteligente para evitar pacotes desnecessários, visto que é um projeto IoT, onde o uso de banda está dentro de um determinado limite. Foi criado uma função para verificar se a rota para um determinado endereço IP está sendo realizada via interface ***wwan0*** (modem USB), em caso negativo geramos uma exceção do tipo WanFailError. Então realizamos um teste de *ping ICMP*, onde é enviado pacotes para o IP central e espera-se receber um retorno, assim sinalizando que o

dispositivo IoT e o servidor central estão se comunicando. A função de *ping ICMP* foi customizada de modo que podemos definir na seção de configuração qual o tempo do *timeout* e qual a quantidade de sondas/pacotes que serão enviadas no teste de *ping*. Caso o teste não seja realizado com sucesso para o ambiente central, continuamos para um teste de uma *pool* de endereços IP's definidos pelo usuário na seção de configuração. Nessa rotina **check_internet_connection()**, realizamos um procedimento específico para cada endereço IP na lista de IP's definida, onde adicionamos uma rota via o dispositivo modem USB, realizamos o teste ICMP de ping e então deletamos essa rota adicionada. Os endereços padrão são os servidores de DNS do Google (8.8.8.8 e 8.8.4.4). Caso nenhum teste ICMP passe, disparamos uma exceção do tipo WanFailError. Caso contrário, prosseguimos para segunda parte do script.

Na segunda parte do script, a lógica é diferente. Nessa parte, a conexão já foi estabelecida e todos os parâmetros foram validados anteriormente, portanto, nesse estado iremos apenas monitorar a saúde da conexão e alguns parâmetros que o sistema operacional informa para nós via módulos do sistema. Então, a cada 30 segundos, por padrão. Verificamos alguns parâmetros. Monitoramos se, por algum motivo, o sistema operacional adicionou uma rota '0.0.0.0'(qualquer endereço), em caso afirmativo deletamos ela. Verificamos informações específicas do modem como: **modem.generic.state, modem.generic.power-state**. Em caso do estado não ser igual a **'connected'** lançamos a exceção do tipo ModemFailedConnection. Também registramos em um log as informações da tecnologia sendo usada para transmissão de dados (UMTS, GSM, GPRS, HSDPA…) e qualidade do sinal registrada pelo modem em %. Nessa segunda parte, além de monitorar a cada 30 segundos essas informações, também a cada 1 hora, por padrão, verificamos o estado da conectividade WAN através de testes ICMP modelados de maneira inteligente para economizar o uso de banda, são realizados testes com o servidor central e os endereços IP's da lista de endereços IP's definida pelo operador. Estabeleceu-se um mecanismo para em última instância, em caso de um evento que o dispositivo tenta executar um número de conexões via modem sem sucesso maior do que o número limite definido pelo usuário, o sistema operacional executa um reboot do dispositivo.

Seção de configuração - Perfil de Conexão

Essa seção foi modelada com o intuito de facilitar a operação em caso de troca de operadoras, na qual definimos o nome do perfil de conexão usado pelo NetworkManager, a APN da operadora e o usuário e senha usados para autenticação.

```
CONNECTION_NAME = 'gsm-con'
CONNECTION_APN = 'operadora-movel.com.br'
```

```
CONNECTION_USER = 'operadora-movel'
CONNECTION_PWD = 'operadora-movel'
```

Código 5. Configuração da Operadora.

Seção de configuração - Variáveis de ambiente e Temporização

```
# number of connection up tries before rebooting the entire system
CONNECTION_TRIES_THRESHOLD = 20
# log information about connections every connection try.
LOG_ON_CONNECT=True
# log all modem info from ModemManager during connection try.
LOG_ALL_FIELDS=False
# log wan connection check information. if true, will log information
about wan connectivity every WAN_CHECK_TIME_SECONDS to log file.
LOG_WAN_CHECK=False
# log periodic modem check connectivity. if true, will log access
tech/signal/status/power state every MODEM_CHECK_TIME_SECONDS to log
file.
LOG_MODEM_CHECK=False
# path where the logs of this application will be stored
LOG_FILE_PATH='/var/log'
# ip that will be added as a route through the USB modem
CENTRAL_SERVER_IP='200.237.192.70'
# ip pool to check the WAN connection status
CHECK_IP=['8.8.8.8','8.8.4.4']
# ping packets amount
PACKETS_COUNT='2'
# time in second for the ping timeout
PING_TIMEOUT='5'
# check wan connection every x seconds 900 = 15min ... 1800 = 30min...
(check array check_ips)
WAN_CHECK_TIME_SECONDS=3600
# check modem signal/power state/connectivity.. every 30 seconds
MODEM_CHECK_TIME_SECONDS=30
# sleep time used in some parts of the code (waiting for usb
rebind/nmcli modules etc..)
SLEEP_TIME=30
# time to wait for route setup from nmcli
NMCLI_WAIT_TIME=5
```

Código 6. Configuração do Script de Gerência de Conexão Móvel.

Essa seção de configuração foi modelada de forma a definir alguns parâmetros:

- CONNECTION_TRIES_THRESHOLD: Variável inteira que define o número máximo de tentativas do processo de achar o caminho do modem no sistema e tentar subir uma conexão WAN para esse modem. Após o limite, o sistema irá reiniciar.
- LOG_ON_CONNECT: Variável booleana que define a opção do operador de registrar informações sobre as tentativas de conexão no modem.
- LOG_ALL_FIELDS: Variável booleana que define se o programa deve registrar todas informações recebidas pelo módulo do sistema operacional ModemManager.

- LOG_WAN_CHECK: Variável booleana que define se o programa deve registrar o processo de testes ICMP periódicos, com o fim de verificar o estado da conexão móvel.
- LOG_MODEM_CHECK: Variável booleana que define se o programa irá registrar informações de tecnologia de acesso usada na rede WAN, intensidade do sinal. status genérico e estado de energia do modem periodicamente.
- LOG_FILE_PATH: Sequência de caracteres que define o caminho no sistema operacional do arquivo de *logs* deste programa.
- CENTRAL_SERVER_IP: Sequência de caracteres que define o endereço IPv4 do servidor central de monitoramento, para onde os dados serão enviados e rota via dispositivo USB adicionada.
- CHECK_IP: Lista de endereços IPv4 usados para o teste ICMP, para cada IP dessa lista, será adicionado uma rota temporária para realizar o teste e depois essa rota será deletada.
- PACKETS_COUNT: Sequência de caracteres que define a quantidade de pacotes usados para realizar o teste ICMP de ping, mais pacotes resultam em mais banda utilizada.
- PING_TIMEOUT: Sequência de caracteres que define o timeout de espera de resposta de um pacote ICMP.
- WAN_CHECK_TIME_SECONDS: Variável inteira que define, em segundos, a periodicidade dos testes ICMP, tempos menores resultam em ações mais rápidas para eventuais falhas na rede porém resultam em mais banda utilizada.
- MODEM_CHECK_TIME_SECONDS: Variável inteira que define, em segundos, a periodicidade da checagem de status gerais do Modem (não utiliza banda).
- SLEEP_TIME: Variável inteira que define, em segundos, o tempo padrão de espera para os trechos onde o programa fica ocioso.
- NMCLI_WAIT_TIME: Variável inteira que define, em segundos, o tempo de espera para o *setup* de rotas do módulo do sistema operacional NetworkManager.

Exceções e tratamento

Nesse trecho iremos apresentar trechos de códigos usados para o programa tratar diversos tipos de exceção, tornando o sistema tolerante a falhas.

```
# class created to raise an exception when  modem state is failed
class ModemFailedState(Exception):
      pass
```

```
# class created to raise an exception when a modem is not found by the
application
class ModemNotFound(Exception):
      pass
# class created to raise an exception when a parsing error occurs
class ParsingError(Exception):
      pass
# class created to raise an exception when a modem is not able to
establish a connection or it's state is not connected during
monitoring
class ModemFailedConnection(Exception):
      pass
# class created to raise an exception when we don't have a WAN
connection
class WanFailError(Exception):
      pass
class GatewayNotFound(Exception):
      pass
```

Código 7. Exceções Criadas.

- **ModemFailedState:** Esse tipo de exceção ocorre quando buscamos o estado do modem e ele possui o valor de **'failed'**. Primero buscamos o caminho do modem e a razão que o módulo do sistema operacional repassa para o programa e registramos em log, então usamos o comando de resetar o modem ***"mmcli -r -m caminho_do_modem"*** e verificamos o código de retorno do processo, caso o código de retornos seja diferente de 0, significa que o processo falho, então usamos o script ***rebind.sh*** para simular uma reconexão lógica do modem. Se o modem não é encontrado então realizamos reconexão lógica e em ambos casos o programa espera SLEEP_TIME para o sistema operacional tratar essas reconexões. depois uma nova tentativa de conexão será feita, visto que o programa está em um loop. o tratamento se dá pelo seguinte trecho de código:

```
except ModemFailedState as exc:
            log('EXCEPTION {}'.format(exc), LOG_FILE,True)
            try:
                  path = get_modem_path()
                  reason = get_custom_field(path,'reason')
                  log('EXCEPTION TREATMENT failed reason is {} will
reset the modem to power cycle settings.'.format(reason),
LOG_FILE,True)
                  retcode = reset_modem(path)
                  if retcode != 0:
                        log('Reseting modem failed, will rebind the usb
hub', LOG_FILE,True)
                        rebind_usb_hub()
                  log('Sleeping {} seconds.'.format(SLEEP_TIME),
LOG_FILE,True)
                  t.sleep(SLEEP_TIME)
            except ModemNotFound as exc2:
```

```
                    log('EXCEPTION {}'.format(exc), LOG_FILE,True)
                    rebind_usb_hub()
                    log('Sleeping {} seconds.'.format(SLEEP_TIME),
LOG_FILE,True)
                    t.sleep(SLEEP_TIME)
```

Código 8. Exemplo de tratamento de Exceção.

- **ModemNotFound:** Esse tipo de exceção ocorre quando tentamos procurar o caminho do modem pelo sistema operacional e não recebemos nenhuma resposta, nesse caso apenas usamos o script de reconexão lógica ***rebind.sh*** e esperamos SLEEP_TIME para a nova tentativa de conexão.
- **ParsingError:** Esse tipo de exceção ocorre geralmente quando tentamos realizar o *parsing* de uma saída do sistema operacional e o tratamento não ocorreu como esperado, as possíveis causas são: o modem não foi encontrado ou perdemos a conexão quando ela já existia, o processo de tratamento é o mesmo que o de **ModemFailedState**, exceto pela parte em que é registrada a razão do estado **'failed'**.
- **ModemFailedConnection:** Esse tipo de exceção ocorre quando tentamos conectar o modem na parte 1 de ***lte_dial.py*** e encontramos algum erro na mensagem que o sistema operacional repassa para o programa. Ou, quando buscamos o estado genérico do modem e ele não é **'connected'** no processo ***Buscar modem ➤ Buscar porta ➤ Subir conexão na porta***. O tratamento se dá pelo mesmo processo que o de **ModemFailedState**, exceto pela parte em que é registrada a razão do estado **'failed'**.
- **WanFailError:** Esse tipo de exceção ocorre quando os testes ICMP estão sendo realizados e houve algum problema nas rotas do dispositivo, por exemplo, os pacotes não estão sendo encaminhados para a interface ***wwan0***, o modem USB. Também ocorre quando tentamos validar a roda para o servidor central de monitoramento e ocorre o mesmo problema. Essa exceção também ocorre no caso de todos testes ICMP falhar, ou seja, tanto o do servidor central como da lista de endereços IPv4 definida pelo operador. O tratamento se dá por uma nova tentativa de conexão no modem, repetindo todo o processo.
- **GatewayNotFound:** Esse tipo de exceção ocorre quando estamos buscando o *gateway* definido dinamicamente pelo ModemManager ao subir a conexão. Se o gateway não é achado ou o seu valor foi determinado como '0.0.0.0' (significa, qualquer endereço ou seja, rota *default*). O tratamento se dá por uma nova tentativa de conexão no modem, repetindo todo o processo.

Função *check_internet_connection()*:

Essa função foi desenvolvida para realizar testes ICMP de ping de maneira inteligente, de modo que certificamos que as rotas dos pacotes estão sendo feitas via dispositivo modem USB. Nela para cada endereço IPv4 na lista CHECK_IP definida pelo operador, que tem por padrão os endereços dos servidores DNS do Google (8.8.8.8 e 8.8.4.4), iremos subir uma rota via *gateway* passado por parâmetro, que é o *gateway* fornecido pelo sistema operacional e indica o caminho padrão dos pacotes que passam pelo modem para rede WAN. Então realiza-se um teste ICMP de ping customizado para o endereço IP e deletamos essa rota que foi subida anteriormente. Se algum dos testes passar com sucesso, já significa que a conectividade WAN está funcionando, não havendo necessidade para gastar mais banda, paramos o loop e ignoramos eventuais outros testes e retornamos que pelo menos um teste passou. Caso nenhum teste passe, retornamos que nenhum teste foi realizado com êxito.

```
def check_internet_connection(gw):
      one_test_succeeded=False
      for ip in CHECK_IP:
            # if ip route is not via wwan0 return false
            add_route(ip,gw)
            if check_route(ip) == False:
                  log('The route for ip {} is not via
wwan0.'.format(ip),LOG_FILE,True)
                  delete_route(ip,gw)
                  raise(WanFailError('WanFailError: {} route was not
via modem.'.format(ip)))
                  return False
            return_code = ping(ip)
            if (return_code == 0):
                  if not LOG_FILE.closed:
                        log('Ping {} via {}
succeeded.'.format(ip,gw),LOG_FILE,True)
                  delete_route(ip,gw)
                  return True
            else:
                  if not LOG_FILE.closed:
                        log('Ping {} via {} failed. Ping return code
{}.'.format(ip,gw,return_code),LOG_FILE,True)
                  delete_route(ip,gw)
      return one_test_succeeded
```

Código 9. Validação do estado da conexão WAN.

4.3.4 MÓDULO *MONITOR_NET.PY*: CONSOLIDAÇÃO DA BANDA TRAFEGADA PELO MODEM

Esse software foi implementado com o fim de monitorar o uso da franquia de dados de uma interface de rede específica, nesse caso, a interface de rede do Modem USB, com isso, será possível visualizar qual o consumo diário de banda do dispositivo e uma compilação

mensal do uso de banda em *bytes*. Desse modo, temos um processo de auditoria onde verificamos se o consumo de dados repassado pela operadora está conforme esperado.

O script terá dois arquivos principais onde será feito o registro em disco do consumo da franquia, isso foi feito para em caso de reset do dispositivo não perdermos esses dados que estariam em memória volátil. Em intervalos específicos, o script irá realizar leituras dos caminhos do sistema operacional para a interface *wwan0* (modem USB) e irá somar a quantidade de bytes de entrada e de saída na interface. Esse cálculo é realizado pelas seguintes funções:

```
def read_sys_rx_bytes():
      path = '/sys/class/net/{}/statistics/rx_bytes'.format(IFNAME)
      proc = run_cmd(['cat',path])
      return proc.stdout.decode('utf-8')
def read_sys_tx_bytes():
      path = '/sys/class/net/{}/statistics/tx_bytes'.format(IFNAME)
      proc = run_cmd(['cat',path])
      return proc.stdout.decode('utf-8')
def sys_total_bytes():
      tx = read_sys_tx_bytes()
      rx = read_sys_rx_bytes()
      total = int(tx) + int(rx)
      return total
```

Código 10. Leitura da quantidade de Bytes trafegada na conexão móvel.

Após a checagem, é comparado com o valor anterior registrado e então é calculado a variação em Kbytes de banda usada e se essa banda for maior do que os limites definidos será enviado ao Zabbix os dados coletados. Além disso, quando completa-se às 24 horas diárias de um dia e o relógio marca outro dia diferente do anterior, então será enviado o consumo de bytes diário e será calculado o consumo de bytes mensal e esse valor será enviado também. A soma dos registros diários, usado para compilar o consumo mensal é feito através da seguinte função:

```
def sum_daily_records():
      f = open(MONTHLY_FILE, 'r')
      list_of_lines = f.readlines()
      f.close()
      total = 0
      for line in list_of_lines:
            splitted = line.split()
            if len(splitted) > 1:
                  total += float(splitted[1])
            else:
                  sys.exit("Error - while reading monthly file, not in
specified format")
      return total
```

Código 11. Compilação das leituras de banda.

O arquivo de registro DAILY_FILE é usado para registrar em disco as variações de banda e em caso de reset do dispositivo, esse arquivo será lido e verificado através de expressões regulares para o formato especificado, ao final do tratamento do arquivo iremos ter um valor inicial que será usado para calcular as próximas expressões através da função de varição Δ_{bytes} = bytes - $bytes_{inicial}$, onde bytes é o último valor lido pela função sys_total_bytes() e $bytes_{inicial}$ é o valor inicial lido pela função, que em caso de já existir um registro em disco de bytes lidos, será esse valor, e Δ_{bytes} será o valor em bytes consumido de banda. A configuração do script se dá pelo trecho de código abaixo:

```
## SCRIPT CONFIGURATION ##
# interface name to be monitored
IFNAME="wwan0"
# threshold warning in KB example if franchise == daily and data usage
is more then THRESHOLD_WARNING (ex:300) then will send a warning to
zabbix server
THRESHOLD_WARNING=300
# threshold critical in KB example if franchise == daily and data
usage is more then THRESHOLD_CRITICAL (ex:500) then will send a
warning to zabbix server
THRESHOLD_CRITICAL=500
# check period time in seconds, default = 600, will refresh data usage
every CHECK_PERIOD seconds
CHECK_PERIOD=600
DAILY_FILE='/var/log/daily_data.log'
MONTHLY_FILE='/var/log/monthly_data.log'
```

Código 12. Configuração do Script de Leitura da Banda.

- IFNAME: Sequência de caracteres que define o nome da interface que será monitorada usando a função sys_total_bytes().
- THRESHOLD_WARNING: Variável inteira que define o limite inicial que será usado para determinar a partir de qual momento será enviado dados relacionados a banda usada para o ambiente central.
- THRESHOLD_CRITICAL: Variável inteira que define o limite crítico que será usado para determinar quando será enviado dados relacionados a banda usada para o ambiente central.
- CHECK_PERIOD: Variável inteira que define, em segundos, o tempo de checagem da variação de banda.
- DAILY_FILE: Sequência de caracteres que define o caminho do arquivo em disco onde serão feitos os registros durante o dia.
- MONTHLY_FILE: Sequência de caracteres que define o caminho do arquivo em disco onde serão registrados os dados diários para compilação mensal de banda trafegada.

4.3.5 MÓDULO *LED_CONTROLLER.PY*: CONTROLE DO LED OPCIONAL

O script led_controller.py foi um script desenvolvido em python que realiza a verificação do estado da conexão com a internet via rádio e em caso do modem estar com conectividade IoT o led ficar aceso, caso o modem não esteja conectado então o script irá apagar o LED externo. Conforme visualizamos no código abaixo, foi realizado um controle de exceções através do bloco *try*, *except* e *finally* da linguagem Python para realizar o cleanup da GPIO em situações inesperadas. O LED, por padrão, está conectado no pino 6 (GND) do GPIO e o polo positivo (ânodo) do LED no pino 12 (GPIO 18 - PCM Clock), a variável **GPIO_LED_PIN** é igual a 12 no caso que o ânodo do LED está conectado no pino 12. **GPIO.HIGH** indica que o pino do GPIO está no nível lógico alto, ou seja 3.3V com corrente máxima de 16mA. **GPIO.LOW** indica 0V ou nível lógico baixo. A publicação no website [14] explica mais informações sobre o uso de GPIO e seus níveis lógicos.

```
try:
    while True:
        try:
            path = get_modem_path()
            current_state = get_current_state(path)
            if current_state=='connected':
                # if status = connected, turn led on
                GPIO.output(GPIO_LED_PIN, GPIO.HIGH)
            else:
                # if status != connected, turn led off
                GPIO.output(GPIO_LED_PIN, GPIO.LOW)
            time.sleep(CHECK_PERIOD)
        except ModemNotFound as exc:
            print(exc)
            time.sleep(WAIT_TIME)
        except ParsingError as exc:
            print(exc)
            time.sleep(WAIT_TIME)
except Exception as e:
    print('EXCEPTION {}'.format(e))
finally:
    GPIO.cleanup()
```

Código 13. Controle do LED.

4.3.6 MÓDULO *PERIODIC_STATUS.PY*: SCRIPT DE CONTROLE DE CONEXÃO WAN E COMANDOS NO MODEM

O script periodic_status.py foi um script desenvolvido em python com o fim de enviar dados adicionais do ambiente para a central de monitoramento. Esse script foi modelado de forma diferente dos outros para ser um código de fácil manutenção, o objetivo disso é para em algum trabalho futuro, se for de interesse monitorar alguma variável além das monitoradas,

seja facilmente acrescentada neste Script. Hoje, ele monitora o uptime e a qualidade do sinal da rede WAN no local do raspberry pi.

Para alcançarmos essa fácil manutenção, foi usado uma biblioteca de terceiros chamada ***apscheduler (Advanced Python Scheduler)***. Essa biblioteca possui diversos tipos de schedulers ou escalonadores. Conforme a documentação oficial [2], citamos aqui alguns dos escalonadores:

- BlockingScheduler: usado quando o escalonador é a única coisa rodando no processo.
- BackgroundScheduler: usado quando não estiver sendo usando nenhum dos *frameworks* abaixo. E, deseja-se que o agendador seja executado em segundo plano dentro do aplicativo
- AsyncIOScheduler: Usado se estiver usando o módulo asyncio.

Como no nosso caso só queremos enviar dados para o Zabbix, a central de monitoramento, então escolhemos o BlockingScheduler. Então instanciamos o método que executa o escalonador e adicionamos os ***jobs*** que serão executados. Os ***jobs*** são as funções que serão periodicamente executadas. No exemplo abaixo estamos definindo que o uptime será enviado em intervalos periódicos definidos pela variável UPTIME_INTERVAL, que é uma variável inteira definindo a quantidade de segundos que irão contemplar o intervalo. O parâmetro **next_run_time** define quando será a próxima execução da função send_uptime() que é explicitada posteriormente.

```
scheduler.add_job(send_uptime, 'interval',
seconds=UPTIME_INTERVAL,next_run_time=datetime.now())
```

Código 14. Exemplo de escalonamento de envio periódico.

No código abaixo é explicitado a lógica principal do programa, podemos visualizar que para adicionar novos ***jobs*** é muito simples, basta adicionar a função que será executada periodicamente e o intervalo que ela será executada no escalonador:

```
def blocking_schedule():
    scheduler = BlockingScheduler()
    scheduler.add_job(send_uptime, 'interval',
seconds=UPTIME_INTERVAL,next_run_time=datetime.now())
    scheduler.add_job(send_signal_strength, 'interval',
seconds=SIGNAL_STRENGTH_INTERVAL,next_run_time=datetime.now())
    scheduler.start()
    try:
        while True:
            time.sleep(APP_ACTIVITY)
    except (KeyboardInterrupt, SystemExit):
        # Not strictly necessary if daemonic mode is enabled but
should be done if possible
        scheduler.shutdown()
```

```
blocking_schedule()
```

Código 15. Implementação do Escalonador de Tarefas.

Exemplo de implementação de *job*:

O trecho de código abaixo é o ***job*** responsável por enviar o uptime do equipamento para o Zabbix, o comportamento dele é o seguinte: o script roda um subprocesso em **shell** que executa o comando **"awk '{print $1}' /proc/uptime"**, esse comando retorna o uptime do equipamento em segundos e então o programa envia a conversão para inteiro desse valor ao Zabbix através da função ***zabbix_sender()***.

```
def send_uptime():
    cmd_run = subprocess.run('awk \'{print $1}\' /proc/uptime',
shell=True, stdout=subprocess.PIPE)
    uptime = cmd_run.stdout.decode('utf-8').rstrip()
    #log(uptime)
    sender_args_list = zabbix_cmd_uptime.copy()
    sender_args_list.append(str(int(float(uptime))))
    zabbix_sender(sender_args_list)
```

Código 16. Exemplo de *job* de envio periódico do uptime.

4.3.7 MMCLI (MODEMMANAGER CLI)

mmcli [2] é uma interface de linha de comando do ModemManager, que é um serviço para Linux baseado em um barramento de software chamado DBus. Esse mecanismo de comunicação entre processos, fornece uma interface de alto nível para comunicação com modems USB. Ele funciona em paralelo ao NetworkManager e os dois softwares possuem integração. O ModemManager possui suporte para comunicação com diversos tipos de modems para diferentes fabricantes. O software possui uma RIL (*Radio Interface Layer*) ou camada de interface de rádio, que é uma camada do sistema operacional que provê mecanismos para comunicação de *hardwares* via rádio. É possível ainda enviar comandos de baixo nível para os modems através desse módulo, chamamos de comandos AT ou conjunto de comandos de Hayes. Através desse módulo localizamos o modem e resetamos ele em casos excepcionais no script ***lte_dial.py***.

Abaixo podemos visualizar alguns comandos úteis do módulo para análise de troubleshooting ou aplicações específicas:

- Para enviar um comando AT para um modem digite ***`mmcli -m {modem_path/modem_index} --command=COMMAND`***
- Para fazer um reset no modem digite ***`mmcli -m {modem_path/modem_index} -r`***.
- Para mostrar informações do modem use ***`mmcli -m {modem_path/modem_index}`***. exemplo: ***`mmcli -m 0`*** para mostrar informações do modem de índice 0.

- Para mostrar os modems reconhecidos pelo serviço ModemManager use ***`mmcli -L`***.

De acordo com a especificação de cada modem, é possível definir a preferência da tecnologia de conexão à rede móvel (2G, 3G, 4G). Através do comando abaixo, por exemplo, está sendo definido como preferencial a conexão por 2G para o modem de índice 0:

```
mmcli -m 0 --set-allowed-modes='2g|3g' --set-preferred-mode=2g
```

Código 17. Definindo perfil de conexão do modem.

4.3.8 NMCLI (NETWORKMANAGER CLI)

O nmcli [15] é uma interface de linha de comando para o programa utilitário do Linux NetworkManager, o módulo serve para realizar operações de redes de computadores em sistemas operacionais baseados em Linux. Com o nmcli podemos visualizar status gerais das conexões de rede nas interfaces, gerenciar conexões de diferentes tipos, como wi-fi, GSM e ethernet, dispositivos de rede, também é possível monitorar o estado das conexões. Um uso típico do nmcli é automação de scripts que não podem acessar uma interface gráfica para configurar a rede. O nmcli possui um formato de saída que pode ser facilmente tratado em scripts.

Para criar um perfil de conexão móvel no nmcli basta editar na seção de configuração do script ***lte_dial.py***. Toda vez que o script é executado, é deletado os perfis de conexão e criado um novo de acordo com os parâmetros configurados CONNECTION_NAME, CONNECTION_APN,CONNECTION_USER e CONNECTION_PWD. Essas variáveis mudam de acordo com a operadora utilizada para prover a conexão móvel. Abaixo podemos visualizar um exemplo do comando executado para configurar um perfil:

```
nmcli c add connection.type gsm connection.id "gsm-con"
connection.interface-name "" gsm.apn nlt.com.br connection.autoconnect
no gsm.username nlt gsm.password nlt
```

Código 18. Exemplo de criação de perfil de conexão para o Modem.

Abaixo podemos visualizar alguns comandos úteis de troubleshooting para conexão móvel usando o módulo nmcli.

- Para derrubar a conexão LTE manualmente digite ***`nmcli con down gsm-con`***.
- Para subir a conexão LTE manualmente digite ***`nmcli con up gsm-con`***.
- Para editar configurações no perfil de conexão use ***`nmcli con edit gsm-con`***.
- Para mostrar as conexões existentes digite ***`nmcli con show`***.
- Para mostrar os devices reconhecidos digite ***`nmcli device status`***.
- Para mostrar informações sobre os devices digite ***`nmcli device show`***.

4.4 MONITORAMENTO (*ZABBIX*)

O Zabbix é uma ferramenta de gerência de redes de código aberto, é uma das ferramentas de monitoramento usadas pelo PoP-SC/RNP e REMEP-FLN e nesse contexto, é a ferramenta usada para o monitoramento da energia elétrica desses ambientes remotos nesse trabalho, onde cada dispositivo IoT irá ter um *host* correspondente no Zabbix onde suas métricas de monitoramento serão compiladas e apresentadas ao usuário final.

Foi criado um template com o fim de tornar a solução escalável. Templates no Zabbix é uma maneira de automatizar entidades como items de monitoramento, triggers que são alarmes condicionais, gráficos de visualização, telas, regras de descoberta e cenários web. O template "Pimon Zabbix Trapper Discovery" é o nome do template criado. Esse template possui apenas um macro:

- {$NODATA.TIMEOUT}: Valor de tempo no qual será disparado um alarme caso algum dispositivo de monitoramento não envie nenhum dado nesse intervalo de tempo, o valor padrão é 5h.

É criado também um mapeamento de valores para o item de monitoramento "Pimon AC Status", esse item é controlado pelos valores enviados por meio do script ***check_ac_in.py*** cujo funcionamento foi explicado na seção de programas e software. O mapeamento é o seguinte:

- Valor 0 ➢ AC Fault
- Valor 1 ➢ AC OK

Isso significa que se o valor recebido é zero, será feito um mapeamento indicando que ocorreu indisponibilidade de energia no host monitorado e caso o valor seja 1, será mapeado para uma sequência de caracteres que indica que há energia disponível no local.

Visando a escalabilidade da solução, foi criada uma regra de descoberta do tipo *Zabbix trapper*, ou seja um monitoramento feito através de dados enviados de forma ativa pelo host monitorado ao invés do processo gerente pedir dados para o processo agente. No caso desse trabalho usamos o utilitário ***zabbix_sender*** para efetuar o envio de dados, o comportamento dessa ferramenta foi descrito na seção de programas e software. Essa regra de descoberta automatiza os 6 itens de monitoramento da Tabela 3:

NOME DO PROTÓTIPO DE ITEM	CHAVE DE MONITORAMENTO
Pimon {#SITE}: AC Status	pimon.ac.status[{#KEYNAME}]
Pimon {#SITE}: Battery Level	pimon.battery.level[{#KEYNAME}]

Pimon {#SITE}: Mobile Carrier Signal	pimon.carrier.signal[{#KEYNAME}]
Pimon {#SITE}: System Uptime	pimon.uptime[{#KEYNAME}]
Pimon {#SITE}: Traffic Daily	pimon.traffic.daily[{#KEYNAME}]
Pimon {#SITE}: Traffic Monthly	pimon.traffic.monthly[{#KEYNAME}]

Tabela 3. Itens de Monitoramento do Dispositivo.

1. Item **Pimon {#SITE}: AC Status**: Esse item é responsável por indicar o estado da energia no local, é atualizado periodicamente ou em caso de troca de estado, o comportamento desse item é controlado pelo script ***check_ac_in.py***. É um item do tipo numeric (unsigned).
2. Item **Pimon {#SITE}: Battery Level**: Esse item é responsável por indicar o nível de carga da bateria acoplada no dispositivo, o seu controle é feito pelo script ***check_ac_in.py***. É um item do tipo numeric (unsigned) com unidade em %.
3. Item **Pimon {#SITE}: Mobile Carrier Signal**: Esse item é responsável por indicar a intensidade do sinal de telefonia móvel no local do dispositivo, o script que controla seus valores é o ***periodic_status.py***. É um item do tipo numeric (unsigned) com unidade em %.
4. Item **Pimon {#SITE}: System Uptime**: Esse item é responsável por indicar o tempo que o dispositivo está ligado desde o seu processo de *boot*, ou seja, desde que foi ligado pela última vez. O script que controla seus valores é o ***periodic_status.py***. É um item do tipo numeric (unsigned) de unidade uptime.
5. Item **Pimon {#SITE}: Traffic Daily**: Esse item é responsável por indicar o consumo de banda trafegada (soma dos bytes que entram e saem da interface do modem USB) diariamente no dispositivo. O envio de dados é controlado pelo script ***monitor_net.py***. É um item do tipo numeric (float) com unidade B (bytes).
6. Item **Pimon {#SITE}: Traffic Monthly**: Esse item é responsável por indicar o consumo de banda trafegada (soma dos bytes que entram e saem da interface do modem USB) mensalmente no dispositivo. O envio de dados é controlado pelo script ***monitor_net.py***. É um item do tipo numeric (float) com unidade B (bytes).

Além da automatização da configuração desses itens de monitoramento foi feito também a automação de 2 triggers ou alarmes condicionais. A primeira trigger dispara em caso do dispositivo de monitoramento não enviar dados sobre a energia elétrica do local após o tempo definido pela macro, que tem o valor padrão de 5h. A segunda trigger dispara quando

ocorre uma queda de energia no local monitorado, podemos visualizar mais detalhes dos alarmes na Tabela 4.

SEVERIDADE	NOME	EXPRESSÃO
High	Pimon {#SITE} not available (or nodata for {$NODATA.TIMEOUT})	nodata(/PoP-SC Pimon Zabbix Trapper Discovery/pimon.ac.status[{#KEYNAME}],{$NODATA.TIMEOUT})=1
Warning	{#SITE} AC Fault (Pimon)	last(/PoP-SC Pimon Zabbix Trapper Discovery/pimon.ac.status[{#KEYNAME}])=0

Tabela 4. Alarmes/Triggers de monitoramento.

Uma terceira automação de configuração foi feita que é um gráfico de informações gerais de energia do local. É um *graph prototype* com nome **{#SITE} Pimon Information**, comprimento 900x200 que compila as médias das métricas dos itens **Pimon {#SITE}: AC Status** e **Pimon {#SITE}: Battery Level**, ou seja, da disponibilidade de energia e da bateria.

Para facilitar a visualização do comportamento desses macros e regras de descoberta iremos mostrar a configuração de um dispositivo que está rodando em produção, o Pimon IFSC Palhoça. Ao entrar na tela de configuração desse host podemos ver os seus itens de monitoramento configurados na Figura 19 e as triggers configuradas na Figura 20:

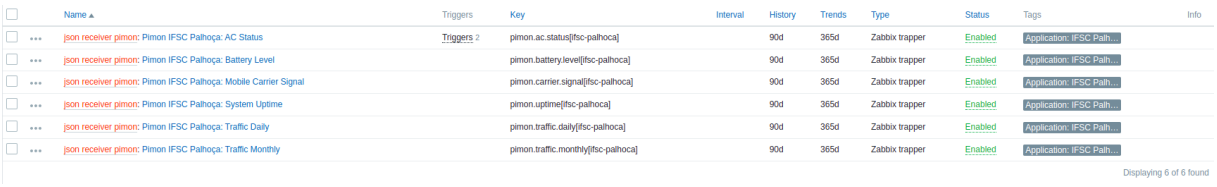

Figura 19. Itens de Monitoramento do dispositivo IFSC Palhoça.

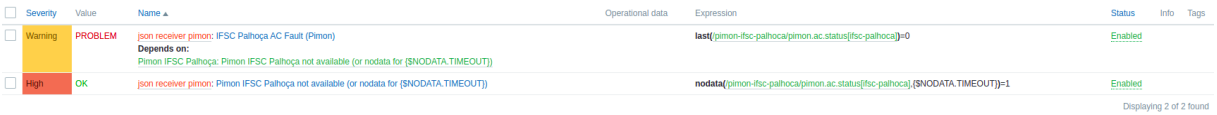

Figura 20. Triggers do dispositivo IFSC Palhoça.

4.5 SEGURANÇA (VPN E CRIPTOGRAFIA)

Sabe-se que dispositivos IoT apresentam diversas vulnerabilidades e riscos relacionados à segurança da informação, para mitigar esses riscos, criou-se um ambiente onde os dados enviados trafegam por uma VPN (*Virtual Private Network*) e os dados são criptografados. A ferramenta usada para implementar essa solução foi o WireGuard. Assim,

cria-se uma conexão segura ponto-a-ponto entre cada nó remoto com o proxy central que recebe as informações

O NAT Traversal é usado nessa solução, ele gerencia problemas relacionados a tradução de endereços em ambientes com dados protegidos com VPN e IPsec, qualquer alteração no endereço IP faz com que o *Internet Key Exchange* descarte os pacotes. O NAT Traversal adiciona uma camada de encapsulamento do UDP aos pacotes IPsec para que eles não sejam descartados na tradução de endereços [17].

5. RESULTADOS

5.1 DISPOSITIVO FINAL

Na Figura 21, podemos visualizar a aparência do dispositivo final agregado e rodando o software desenvolvido dentro da caixa protetora de plástico, no caso deste dispositivo foi optado pelo uso do LED Opcional, indicativo da conectividade WAN. Dentro da caixa protetora estão os demais componentes explicitados na seção de Hardware deste trabalho.

Figura 21. ***PiMon*** - Dispositivo final, rodando a solução de monitoramento.

5.2 CONTRIBUIÇÕES

Após a montagem, instalação dos pacotes de software em um dispositivo, sua posterior instalação física no site remoto (Figura 22) e a configuração do monitoramento no Zabbix será possível de visualizar as métricas coletadas pelos dispositivos de maneira integral no frontend do Zabbix ou com uma configuração posterior em um *dashboard* do **grafana**, visto que o grafana e o Zabbix possuem uma integração de dados, facilitando assim a criação de *dashboards*. Na Figura 23, podemos visualizar 3 dispositivos que estão rodando em produção na tela do Zabbix, junto de suas métricas de monitoramento, os dispositivos estão

localizados na faculdade Estácio São José, UDESC Campus CEFID e UFSC Campus Abelhas, nos 3 casos há energia. Na Figura 24, podemos visualizar o dashboard no grafana que compila os dados de todos os dispositivos monitoramentos instalados e que estão configurados no Zabbix.

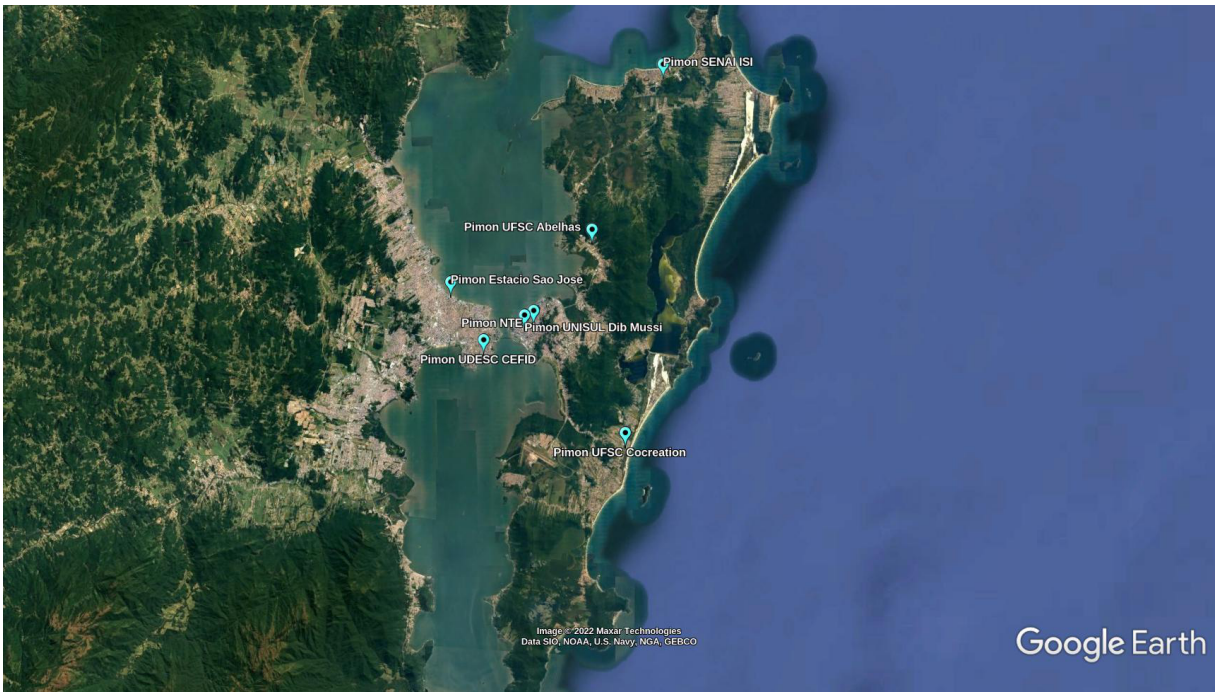

Figura 22. Instalação Física dos Dispositivos nos Sites remotos na região de Florianópolis.

Pimon Estácio São José	Pimon Estácio São José: AC Status	1h 24m 54s	AC OK (1)		Application: Estácio S...	Graph
Pimon Estácio São José	Pimon Estácio São José: Battery Level	1h 24m 54s	88 %		Application: Estácio S...	Graph
Pimon Estácio São José	Pimon Estácio São José: Mobile Carrier Signal	12m 15s	51 %	+3 %	Application: Estácio S...	Graph
Pimon Estácio São José	Pimon Estácio São José: System Uptime	12m 15s	12 days, 00:01:28	+2 days, 00:00:00	Application: Estácio S...	Graph
Pimon Estácio São José	Pimon Estácio São José: Traffic Daily	15h 58m 35s	115.93 KB	+22.31 KB	Application: Estácio S...	Graph
Pimon Estácio São José	Pimon Estácio São José: Traffic Monthly	15h 58m 34s	2.88 MB	+115.93 KB	Application: Estácio S...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: AC Status	49m 27s	AC OK (1)		Application: UDESC C...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: Battery Level	49m 26s	94 %	-3 %	Application: UDESC C...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: Mobile Carrier Signal	53m 9s	41 %	-7 %	Application: UDESC C...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: System Uptime	12h 53m 9s	1 day, 00:01:21	+23:59:59	Application: UDESC C...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: Traffic Daily	16h 3m 6s	132.43 KB	+36.11 KB	Application: UDESC C...	Graph
Pimon UDESC CEFID	Pimon UDESC CEFID: Traffic Monthly	16h 3m 5s	2.72 MB	+132.43 KB	Application: UDESC C...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: AC Status	1h 36m 35s	AC OK (1)		Application: UFSC Ab...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: Battery Level	13m 32s	93 %	-1 %	Application: UFSC Ab...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: Mobile Carrier Signal	4h 15m 27s	0 %	-38 %	Application: UFSC Ab...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: System Uptime	4h 15m 27s	00:01:21	-00:00:07	Application: UFSC Ab...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: Traffic Daily	15h 58m 43s	93.58 KB	+2.92 KB	Application: UFSC Ab...	Graph
Pimon UFSC Abelhas	Pimon UFSC Abelhas: Traffic Monthly	15h 58m 43s	2.3 MB	+93.58 KB	Application: UFSC Ab...	Graph

Figura 23. Visualização do Monitoramento no Zabbix.

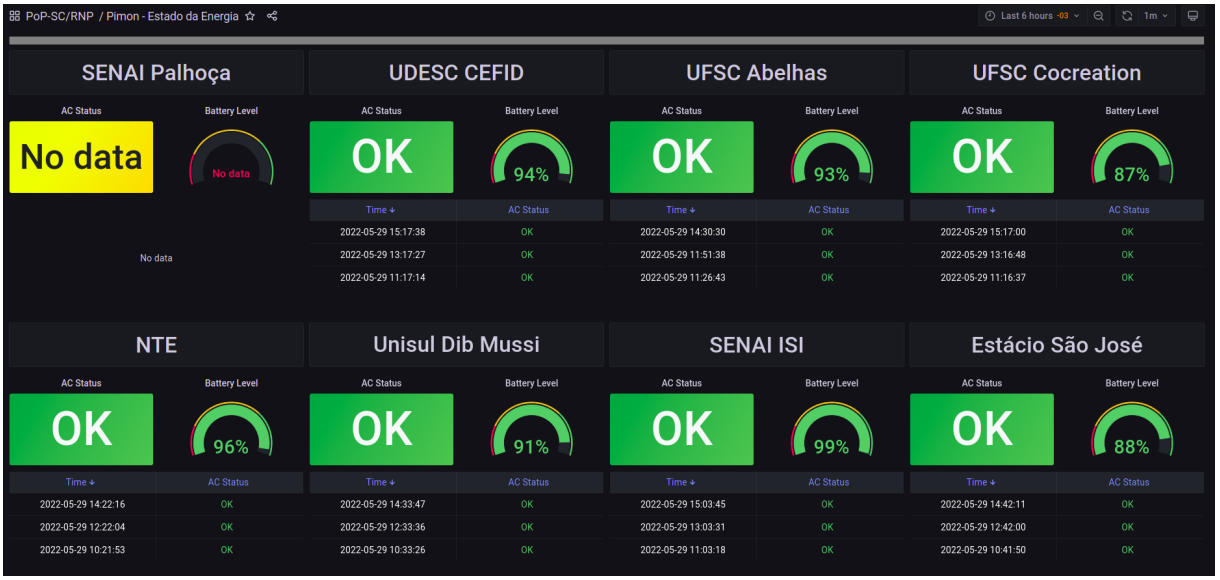

Figura 24. Visualização do Monitoramento no dashboard integrado de monitoramento de energia do grafana.

Com esse projeto foi possível validar a proposta de uma solução escalável para o monitoramento do estado da corrente elétrica das redes em ambientes remotos, esses campus universitários e localidades explicitados nas figuras acima são parceiros da iniciativa da Rede Metropolitana Comunitária de Educação e Pesquisa da Região de Florianópolis (REMEP-FLN) que está incluída na iniciativa da Rede Nacional de Ensino e Pesquisa (RNP) de implantação de redes metropolitanas e as ações relacionadas a falta de energia elétrica nesses localidades hoje são mais assertivas do que outras localidades que não possuem o dispositivo, assim facilitando o trabalho dos operadores da equipe do PoP-SC/RNP e da REMEP-FLN na atuação de chamados de indisponibilidade desses locais ou apenas na validação de energia elétrica do local, por qualquer motivo que seja. Existem alguns pontos de aprimoramento na solução para torná-la mais escalável, uma vez que o projeto de montagem de cada dispositivo é artesanal e precisa de trabalho braçal. Também o aprovisionamento de instalações e configurações de cada dispositivo individual envolve uma quantia significativa de trabalho, portanto, se fosse viabilizado um método de automação da configuração e instalação de alguns pacotes em cada dispositivo, tornaria a solução mais escalável ainda.

5.3 LEVANTAMENTO DA BANDA TRAFEGADA PELOS DISPOSITIVOS

Foram elaborados cálculos relacionados ao intervalo de envio de mensagens e tamanho da carga útil de dados nessas mensagens que estimam o consumo gerado por cada dispositivo. A tabela 5, abaixo, explicita esses cálculos que influenciaram na escolha do plano de tráfego com a provedora do serviço de comunicação. O consumo estimado por mês foi

considerando que existem 12 mensagens por dia do tipo Pimon (alteração de estado) e 72 mensagens por dia do tipo VPN, que incluem Handshake Initiation (190 Bytes), Handshake Response (134 Bytes) e Keepalive (74 Bytes). Com o resultado dos cálculos conclui-se que a banda estimada através desses parâmetros será de 3.7MB e um plano de 5MB com cobrança por excedente de banda seria a melhor alternativa para esse trabalho.

TIPO DA MENSAGEM	TAMANHO DA MENSAGEM (kB)	INTERVALO (minutos)	CONSUMO ESTIMADO NO MÊS (kB)
Pimon - Alteração de Estado (Fault + OK)	8	120	2880
VPN - Handshake Initiation + Hanshake Response + Keepalive	0.398	20	859.68
TOTAL (kB)			**3739**

Tabela 5. Cálculo estimativo de banda por dispositivo.

Foi realizado um levantamento de informações para compilar a Tabela 6. que mostra a quantidade de dados consumidos no mês de maio de 2022 por cada um dos 7 dispositivos em operação. Esses dados foram coletados pelo script ***monitor_net.py*** que calcula a variação de bytes recebidos e enviados na interface de rede do Modem USB e traduz esses dados de modo a informar a banda trafegada total diariamente e mensalmente, assim teremos um controle da banda utilizada e poderemos estimar o custo dos planos IoT para cada dispositivo. Podemos visualizar que a média de consumo é igual a 2.96 MB e o desvio padrão dos dados amostrados é aproximadamente 0,285 MB, ou seja, existe um grau de dispersão baixo entre os dados amostrados.

DISPOSITIVO	BANDA UTILIZADA NO MÊS
Pimon Estácio São José	3.15 MB
Pimon PMF/NTE	2.89 MB
Pimon SENAI Inovação	3.33 MB
Pimon UDESC CEFID	3.03 MB
Pimon UFSC Abelhas	2.41 MB
Pimon UFSC Cocreation	2.91 MB

Pimon Unisul Dib Mussi	3 MB
TOTAL	**20.72 MB**

Tabela 6. Banda Utilizada pela conexão IoT.

5.4 EXEMPLO PRÁTICO - EXEMPLO DE ALARME DE ENERGIA JUNTO DE QUEDA

A Figura 25, é uma fotografia do painel de alarmes do Zabbix, sistema de monitoramento, durante a operação do dia 08/07/2022 no turno da tarde (14h-18h). Note que houve um alarme enviado pelo Pimon de Falta de Energia, o nome do alarme disparado é "UFSC Cocreation AC Fault (Pimon)", e logo depois dispararam outros monitoramentos, de circuitos L2VC e de ICMP Ping, por exemplo, esses alarmes são gerados quando a CPE está indisponível. Portanto, sabe-se que houve uma queda de energia elétrica no local, viabilizando esses outros alarmes. Com isso, obteve-se a informação necessária para atuação ou não em cima dos alarmes e não foi necessário realizar contato com a instituição abrigo dos equipamentos da rede.

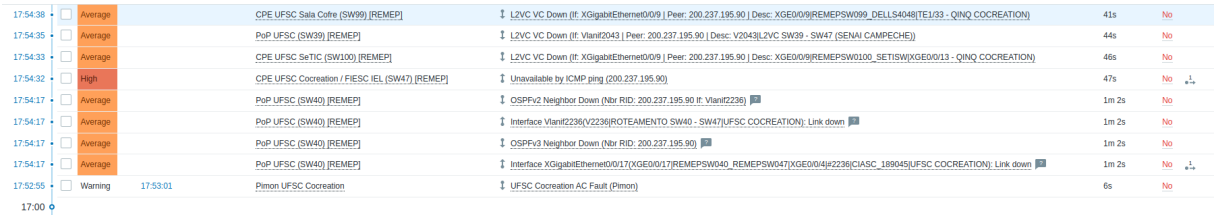

Figura 25. Painel de Alarmes.

Para confirmar o reset do equipamento de rede, foi levantado o seu Uptime via ferramenta de gerência Zabbix, que coleta entre outras métricas o Uptime do equipamento via protocolo SNMP. Ao analisar o valor, visualizamos que o Uptime foi zerado, isso significa que a CPE efetuou o processo de *reboot*. Ao efetuar a subtração de 18h01m32s pelo valor de 400s, obtemos o valor de 17h54m52s, ou seja esse é o horário que a CPE efetuou o processo de *boot*, coincidindo com o tempo de volta de energia apontado pela coleta de métricas e envio de dados do dispositivo pimon, cujo valor é 17h53m01s.

DATA E HORA	VALOR DO UPTIME (SEGUNDOS)
2022-07-08 18:01:32	400

Tabela 7. Valor do Uptime do dispositivo.

5.5 ANÁLISE DO TEMPO

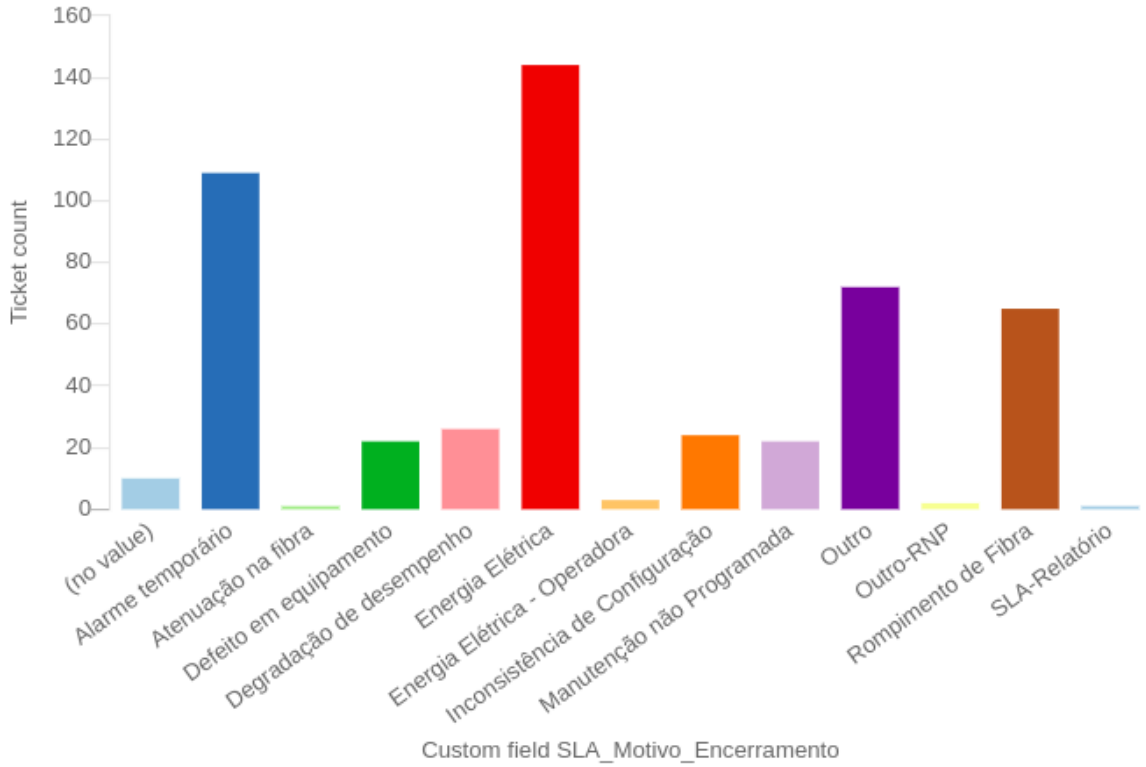

Figura 26. Gráfico de chamados do RT.

Usando a ferramenta RT, levantamos alguns dados da fila de SLA. A partir da seguinte consulta **Query:Queue = 'SLA' AND Created > '2022-01-01' AND Created < '2022-06-02' AND Owner != 'csla'**, ou seja, tickets que são da fila SLA, foram criados a partir de 01 de jan de 2022 até 01 de jun de 2022 (5 meses) e que não foram gerados pela ferramenta csla (ferramenta que gera tickets e encerra automaticamente ao visualizar o reset do *uptime*), podemos visualizar na Figura 26 o resumo dessa consulta. Visualizamos que houveram um total de 501 chamados, e que os principais chamados foram alarmes temporários ou quedas de energia elétrica no local da CPE, nos dois casos o Pimon seria de notável ajuda para a assertividade da verificação de energia. Podemos gerar o resumo de tempo trabalhado desses chamados que está explicitado na tabela 8. É importante ressaltar que o tempo trabalhado dos chamados envolvem outras atividades além do tempo de verificação de energia elétrica, mas a validação de energia é o primeiro passo na atuação desses tickets.

Motivo de Encerramento	Tempo Mínimo	Tempo Médio	Tempo Máximo	Total
(no value)	0s	5m 36s	20m	56m
Alarme temporário	0s	9m 36s	24m	17h 28m

Atenuação na fibra	**5m**	**5m**	**5m**	**5m**
Defeito em equipamento	**9m**	**42m 49s**	**2h 5m**	**15h 42m**
Degradação de desempenho	**0s**	**15m 13s**	**45m**	**6h 36m**
Energia Elétrica	**0s**	**16m 41s**	**2h**	**40h 3m**
Energia Elétrica - Operadora	**5m**	**20m**	**40m**	**60m**
Inconsistência de Configuração	**0s**	**96m 7s**	**12h 10m**	**38h 27m**
Manutenção não Programada	**0s**	**16m 19s**	**2h**	**5h 59m**
Outro	**0s**	**14m 17s**	**2h 35m**	**17h 9m**
Outro-RNP	**12m**	**16m**	**20m**	**32m**
Rompimento de Fibra	**5m**	**31m 35s**	**3h 21m**	**34h 13m**
SLA-Relatório	**5m**	**5m**	**5m**	**5m**
Total	**41m**	**4h 54m 16s**	**26h 50m**	**7d 10h 15m**

Tabela 8. Resumo de Tempo gasto em chamados.

A partir da análise da tabela 8, concluímos que o tempo médio trabalhado em um chamado dessa busca (Total do Tempo médio dividido pelo número de categorias de chamados) é de cerca de 22 minutos, na grande maioria dos casos esse tempo poderia ser reduzido se existisse um Pimon monitorando o local e informando de maneira assertiva se há energia elétrica no local que ocorreu uma indisponibilidade, visto que o procedimento padrão de atuação nos chamados que entram na fila é sempre verificar se o local está com energia elétrica antes de tomar qualquer ação. Analisando apenas casos em que o motivo de encerramento do chamado foi queda de energia elétrica no lado da CPE, visualizamos que o tempo médio trabalhado foi de 16m e o tempo máximo foi de 2h. A análise padrão de energia elétrica, hoje, é lenta e envolve diversas variáveis, supondo que o tempo médio para identificar quedas elétricas é de cerca de 15 minutos, o dispositivo já reduziria muito esse tempo, se supormos que o tempo para verificação com o dispositivo abaixa para 1 minuto

com o dispositivo isso implica em uma redução de aproximadamente 93% do tempo de verificação.

Vale notar que independente do tempo médio, em alguns casos específicos, a indisponibilidade dura horas e o contato com a instituição (ambiente remoto) não é realizado durante todo o tempo da indisponibilidade, por conta de contatos de telefone desatualizados, trabalhadores que não estão no local onde ocorreu a indisponibilidade no momento da ligação ou e-mails que não foram lidos ou entregues, por exemplo, houve um caso em que ocorreu uma queda elétrica no bairro onde está localizado o IFSC Câmpus Gaspar e levou 2h para o chamado ser resolvido, esse tempo poderia ter sido resolvido na escala de minutos se houvesse um dispositivo Pimon no local, porque ele informaria a indisponibilidade de energia elétrica instantâneamente.

Além disso, essa operação de redes automatizada pode ser realizada 24 horas por dia, ou seja, ações podem ser tomadas fora do horário comercial e em finais de semana ou feriados, onde normalmente não é possível obter métricas relacionadas a disponibilidade de energia elétrica porque a interação humana é, com frequência, impossibilitada.

O dispositivo Pimon demora no máximo 2 segundos para identificar uma indisponibilidade elétrica e ao detectar essa queda ele instantâneamente envia os dados para a rede IoT, com direção à central de monitoramento, então o tempo para identificar indisponibilidade de energia elétrica seriam apenas o tempo do operador associar o dado de indisponibilidade com a CPE indisponível. Isso ajuda em casos que houve rompimento da fibra no caminho do tráfego da instituição, queda de energia elétrica no campus, queda de energia elétrica no lado da operadora, erros de configuração, manutenções não programadas, erros de configuração, quedas de PoP's/PoA's da operadora, problemas em equipamentos, entre outros.

6. CONCLUSÃO E TRABALHOS FUTUROS

6.1 RESUMO DOS RESULTADOS E CONCLUSÃO

Diante das informações acima, podemos concluir que os principais resultados e ganhos são na visualização do operador e/ou engenheiro de redes, que facilita o seu trabalho no dia a dia, na automação de um processo oneroso que envolve buscar contatos, atualizar base de dados e entrar em contato com diferentes instituições, a não dependência de horário comercial e disponibilidade de terceiros em finais de semanas ou feriados para obter informações precisas que dizem respeito a disponibilidade de energia elétrica e a agilização

no processo de abertura de chamados, de modo que o tempo de indisponibilidade das organizações usuárias tendem a reduzir.

6.2 TRABALHOS FUTUROS

- **Automação da Configuração e Instalação dos Softwares nos dispositivos:**

Um script instalador e auto configurador de novos nós pode ser desenvolvido de modo que o processo de alocação de um novo módulo IoT seja rápido e fácil. Algumas prováveis ferramentas que podem ser utilizadas para isso são a combinação de *Bash* e/ou Ansible.

- **Implementação de caminhos e tecnologias alternativas para conexão WAN:**

Validar a adição de tecnologias WAN alternativas a telefonia móvel, de modo que em ambientes onde o sinal das antenas de rádio é desfavorável para o estabelecimento de uma conexão a solução não seja inviabilizada. Também verificar a possibilidade de conectar o dispositivo via cabo UTP ou Wi-fi em redes locais, de modo que a banda da operadora possa ser economizada quando seu uso não for necessário.

- **Adição de outras métricas de monitoramento:**

A adição de novos sensores, como de temperatura e umidade pode ser útil em casos que deseja-se monitorar outras métricas além da disponibilidade de energia elétrica. Existem portas GPIO disponíveis no raspberry pi 0 e esses dispositivos podem ser facilmente adicionados e integrados na solução, devido ao modo que foi implementado.

- **Alimentação PoE (*Power over Ethernet*):**

A adição de uma alimentação PoE pode ser vantajosa porque o dispositivo poderia ser diretamente conectado em CPE's com entradas alimentadas RJ-45. Com isso, dispensa-se o uso de tomadas e a validação da energia elétrica seria diretamente no equipamento de rede e não na distribuidora de energia elétrica.

- **Acesso Serial a CPE's:**

Se for possível validar o uso de acesso serial, as CPE's poderiam ser configuradas remotamente via interface serial conectada diretamente no dispositivo IoT. Isso seria uma maneira de acesso *out-of-band* nos equipamentos de rede e de configuração remota em casos que perde-se a configuração do dispositivo, grandemente vantajoso em ambientes remotos, de longa distância ou difícil acesso.

REFERÊNCIAS

[1] FREECODECAMP. Controlling an External LED using a Raspberry Pi and GPIO pins. 2018. Disponível em

<https://www.freecodecamp.org/news/hello-gpio-blinking-led-using-raspberry-pi-zero-wh-65af81718c14/>. Acesso em 21 mai. 2022.

[1] GRONHOLM, Alex. Documentação apscheduler. 2021. Disponível em: <https://apscheduler.readthedocs.io/en/3.x/userguide.html>. Acesso em 14 jun. 2022.

[2] RUSSEL, M; MORGADO, A. Manual MMCLI. 2022. Disponível em: <https://www.freedesktop.org/software/ModemManager/man/latest/mmcli.1.html>. Acesso em: 14 jun. 2022.

[3] HUURDEMAN, A. The Worldwide History of Telecommunications. Local de publicação: John Wiley & Sons, 31 de julho de 2003.

[4] ZABBIX. Documentação Zabbix Trapper. 2022. Disponível em: <https://www.zabbix.com/documentation/current/pt/manual/config/items/itemtypes/trapper>. Acesso em: 14 jun. 2022.

[5] ABOUBAKAR, Moussa; KELLIL, Mounir; ROUX, Pierre. A review of IoT network management: Current status and perspectives. Journal of King Saud University-Computer and Information Sciences, 2021.

[6] KUMAR, R.; RAJASEKARAN, M. Pallikonda. An IoT based patient monitoring system using raspberry Pi. In: 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16). IEEE, 2016. p. 1-4.

[7] KASHANI, Mostafa Haghi et al. A systematic review of IoT in healthcare: Applications, techniques, and trends. Journal of Network and Computer Applications, v. 192, p. 103164, 2021.

[8] REMEP-FLN. Boletim Informativo de Ações REMEP-FLN 2020-2022. 2022. Disponível em <https://remep.pop-sc.rnp.br/home/noticias/20220513/1436>. Acesso em: 10 jun. 2022.

[9] NEWMAN, P. IoT Report: How Internet of Things Technology Is Now Reaching Mainstream Companies and Consumers. 2020. Disponível em: <https://www.businessinsider.com/internet-of-things-report>. Acesso em: 09 jun. 2022.

[10] SANTOS, Diogo; FERREIRA, João C. IoT power monitoring system for smart environments. Sustainability, v. 11, n. 19, p. 5355, 2019.

[11] RAGNOLI, Mattia et al. An autonomous low-power LoRa-based flood-monitoring system. Journal of Low Power Electronics and Applications, v. 10, n. 2, p. 15, 2020.

[12] REDHAT. What is edge computing? 2021. Disponível em: <https://www.redhat.com/en/topics/edge-computing/what-is-edge-computing>. Acesso em: 14 jun. 2022.

[13] SHUFIAN, Abu et al. Smart irrigation system with solar power and GSM technology. In: 2019 5th International Conference on Advances in Electrical Engineering (ICAEE). IEEE, 2019. p. 301-305.

[14] BUENO, Cleiton. Raspberry PI – GPIO input com Python. 2014. Disponível em: <https://www.embarcados.com.br/raspberry-pi-gpio-modo-input-python/>. Acesso em: 27. Jun. 2022.

[15] THE GNOME PROJECT. nmcli - command-line tool for controlling NetworkManager. 2014. Disponível em: <https://developer-old.gnome.org/NetworkManager/stable/nmcli.html>. Acesso em: 05. Jul. 2022.

[16] RAO, R. Nageswara; SRIDHAR, B. IoT based smart crop-field monitoring and automation irrigation system. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, 2018. p. 478-483.

[17] JUNIPER NETWORKS. VPNs baseadas em rotas e baseadas em políticas com NAT-T. 2022. Disponível em: <https://www.juniper.net/documentation/br/pt/software/junos/vpn-ipsec/topics/topic-map/security-route-based-and-policy-based-vpns-with-nat-t.html>. Acesso em: 16 set. 2022.

APÊNDICE A – ARTIGO SBC

Solução Escalável para Monitoramento do estado da Corrente Elétrica das Redes em Ambientes Remoto

Caetano C. Torres[1], Rodrigo B. Pescador[2], Prof. Dr. Carlos Becker Westphall[1]

[1]Departamento de Informática e Estatística (INE) - UFSC

[2]Rede Nacional de Ensino e Pesquisa - RNP
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil

caetanocolintorres@gmail.com

Abstract. *The solution encompasses the development of a stable, low cost, scalable, easy to install and maintain IoT device that will monitor the state of the electrical grid at remote sites. A single board hardware (Raspberry Pi Zero) will be coupled with a battery module with electrical power supply, along with a USB modem that will use mobile phone technology for data transmission in the absence of connection to the environment's local network. This hardware will read information from the GPIO pins and tell if there is electrical current available on the network or not. In addition, the developed system will be integrated with the current monitoring system of the PoP-SC/RNP, Zabbix, and will send periodic additional information from the device (uptime, modem signal strength, battery level and consumption of the data) to the central environment. With this information sent by the monitoring device, it will be possible to make more assertive decisions, facilitating and speeding up the process of making requests with the connectivity transport providers hired by the company.*

Resumo. *A solução engloba o desenvolvimento de um dispositivo IoT estável, de baixo custo, escalável, de fácil instalação e manutenção que irá monitorar o estado da rede elétrica em sites remotos. Um hardware de placa única (Raspberry Pi Zero) será acoplado com um módulo de bateria com fornecimento de energia elétrica ininterrupto, junto a um modem USB que usará tecnologia de telefonia móvel para a transmissão de dados na falta de conexão com a rede local do ambiente remoto. Esse hardware irá ler informações dos pinos GPIO e informar se há corrente elétrica disponível na rede. Além disso, o sistema desenvolvido será integrado com o sistema de monitoramento atual do PoP-SC/RNP, cujo nome é Zabbix, e enviará informações adicionais periódicas do dispositivo (uptime, intensidade do sinal do modem, nível da bateria e consumo da franquia de dados) para o ambiente central. Com essas informações enviadas pelo dispositivo de monitoramento será possível tomar decisões mais assertivas, facilitando e agilizando o processo de abertura de chamados com as operadoras de transporte contratadas pela empresa.*

1. Introdução

Esse projeto é um projeto IoT desenvolvido em conjunto com o PoP-SC/RNP e REMEP-FLN. Atualmente, quando ocorre indisponibilidade de energia elétrica em um site remoto, é necessário estabelecer contato com a instituição ou organização cliente via telefone ou e-mail para confirmar a disponibilidade, acontece que na maior parte das vezes a instituição que está sem energia elétrica não está de prontidão para dar o retorno em relação a disponibilidade. Por isso, o sistema a ser desenvolvido será elaborado com o fim de agilizar o processo de verificação de disponibilidade de energia elétrica em sites remotos, de modo que seja assertivo e praticamente instantâneo, facilitando a abertura de chamados com as provedoras dos serviços de comunicação quanto a indisponibilidade da conexão em sites remotos e melhorando os tempos de respostas nos SLA's.

A Internet das Coisas (IoT) é um ecossistema em constante expansão que integra software, hardware, objetos físicos e dispositivos de computação para comunicar, coletar e trocar dados.

A IoT fornece uma plataforma perfeita para facilitar as interações entre humanos e uma variedade de coisas físicas e virtuais (KASHANI, 2021)[7]. Peter Newman prevê que haverá mais de 41 bilhões de dispositivos IoT conectados até 2027. Ele também estima, que o mercado de IoT está a caminho de crescer para mais de US$2,4 trilhões anualmente até 2027 [9].

Como objetivo geral, espera-se ao final do trabalho, um hardware agregado e funcional, montado num computador de placa única e acoplado em um módulo de bateria com fornecimento ininterrupto, capaz de enviar informações de indisponibilidade elétrica via modem USB e uma integração nos sistemas de monitoramento do PoP-SC/RNP e REMEP-FLN.

2. Conceitos Básicos

O termo IoT (Internet of Things) ou Internet das Coisas, em português, vêm recebendo notoriedade ao longo da última década. O desenvolvimento de dispositivos IoT conta com tecnologias de estado da arte e a sua gerência entra como destaque na área de gerência de redes. O uso de dispositivos inteligentes como sensores inteligentes ou agentes atuadores causa diversos problemas de performance na rede, e a gerência de redes eficiente para esses dispositivos é fundamental para uma boa performance (ABOUBAKAR, 2021)[5].

O processo de administração e gerência de redes é necessário para manutenção das redes de computadores. Inclui a provisão de infraestrutura de rede para conectividade dos usuários, análise e monitoramento de incidentes, gerenciamento de segurança e manutenção da qualidade de serviço (QoS). Existem diversos softwares de gerenciamento de rede no mercado e geralmente são administrados por times de operações de redes ou engenheiros de redes.

No caso deste trabalho usaremos o Zabbix como ferramenta de gerência de redes. O Zabbix, é uma ferramenta de software de código aberto para monitorar a infraestrutura de TI, como redes, servidores, máquinas virtuais e serviços em nuvem. Ele será usado para coletar as métricas que serão analisadas. Vale destacar que será utilizado o utilitário Zabbix Sender, um utilitário comumente usado em scripts que desejam enviar dados periódicos, aplicando perfeitamente para o caso deste trabalho de monitoramento de ambientes remotos.

3. Implementação

O modelo inicial pensado para a montagem do dispositivo físico está demonstrado na figura 1, que mostra um diagrama onde temos o computador de placa única acoplado em um módulo de bateria, sendo alimentado por um cabo de energia elétrica e com um modem USB ligado ao computador, tudo isso dentro de uma caixa protetora de plástico, também chamado de patola.

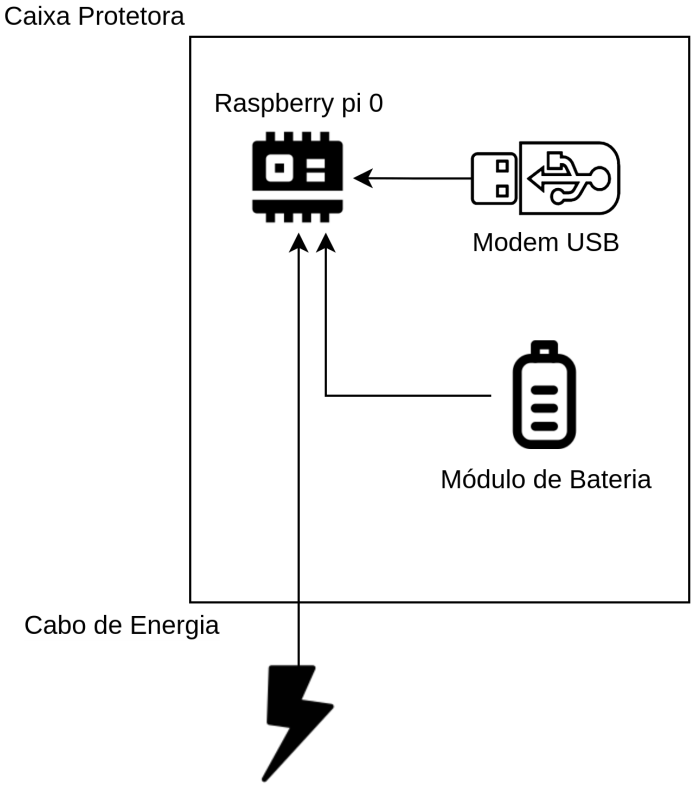

Figura 1. Modelagem Inicial de Montagem.

Abaixo são apresentados os principais módulos de software desenvolvidos e/ou utilizados, principalmente na linguagem Python, com o fim de executar as tarefas necessárias e abranger o escopo do trabalho definido anteriormente, compreendendo comentários sobre configuração, implementação e testes realizados. A tabela abaixo especifica quais são os módulos de implementação própria e quais são os módulos de terceiros, junto da finalidade de cada módulo.

Tabela 1. Principais Módulos de Software Envolvidos

Nome do Módulo	Implementação	Finalidade
rebind.sh	Própria	Utilitário que simula a reconexão lógica do modem.
check_ac_in.py	Própria	Script de verificação do estado da corrente elétrica.
lte_dial.py	Própria	Script de controle de conexão WAN e gerência do Modem.
monitor_net.py	Própria	Utilitário de rede que monitora o consumo da banda trafegada.
led_controller.py	Própria	Controle do LED indicativo do estado da conectividade IoT.

periodic_status.py	Própria	Envio periódico de informações do ambiente através de um escalonador.
mmcli (ModemManager CLI)	Terceiros	Interface de linha de comando para gerência de dispositivos com comunicação via rádio, i.e. Modem USB.
nmcli (NetworkManager CLI)	Terceiros	Interface de linha de comando para gerência de redes no sistema operacional.

4. **Resultados e Conclusão**

Após a montagem, instalação dos pacotes de software em um dispositivo, sua posterior instalação física no site remoto e a configuração do monitoramento no Zabbix será possível de visualizar as métricas coletadas pelos dispositivos de maneira integral no frontend do Zabbix ou com uma configuração posterior em um dashboard do grafana, visto que o grafana e o Zabbix possuem uma integração de dados, facilitando assim a criação de dashboards.

Os principais resultados e ganhos são na visualização do operador de redes, que facilita o seu trabalho no dia a dia, na automação de um processo oneroso que envolve buscar contatos, atualizar base de dados e entrar em contato com diferentes instituições, a não dependência de horário comercial e disponibilidade de terceiros em finais de semanas ou feriados para obter informações precisas que dizem respeito a disponibilidade de energia elétrica e a agilização no processo de abertura de chamados, de modo que o tempo de indisponibilidade das organizações usuárias tendem a redução.

5. **Trabalhos Futuros**

Automação da Configuração e Instalação dos Softwares nos dispositivos:

Um script instalador e auto configurador de novos nós pode ser desenvolvido de modo que o processo de alocação de um novo módulo IoT seja rápido e fácil. Algumas prováveis ferramentas que podem ser utilizadas para isso são a combinação de Bash e/ou Ansible.

Implementação de caminhos e tecnologias alternativas para conexão WAN:

Validar a adição de tecnologias WAN alternativas a telefonia móvel, de modo que em ambientes onde o sinal das antenas de rádio é desfavorável para o estabelecimento de uma conexão a solução não seja inviabilizada. Também verificar a possibilidade de conectar o dispositivo via cabo UTP ou Wi-fi em redes locais, de modo que a banda da operadora possa ser economizada quando seu uso não for necessário.

Adição de outras métricas de monitoramento:

A adição de novos sensores, como de temperatura e umidade pode ser útil em casos que deseja-se monitorar outras métricas além da disponibilidade de energia elétrica. Existem portas GPIO disponíveis no raspberry pi 0 e esses dispositivos podem ser facilmente adicionados e integrados na solução, devido ao modo que foi implementado.

Alimentação PoE (Power over Ethernet):

A adição de uma alimentação PoE pode ser vantajosa porque o dispositivo poderia ser diretamente conectado em CPE's com entradas alimentadas RJ-45. Com isso, dispensa-se o uso de tomadas e a validação da energia elétrica seria diretamente no equipamento de rede e não na distribuidora de energia elétrica.

Acesso Serial a CPE's:

Se for possível validar o uso de acesso serial, as CPE's poderiam ser configuradas remotamente via interface serial conectada diretamente no dispositivo IoT. Isso seria uma maneira de acesso out-of-band nos equipamentos de rede e de configuração remota em casos que perde-se a configuração do dispositivo, grandemente vantajoso em ambientes remotos, de longa distância ou difícil acesso.

Referências

[Freecodecamp]. (2018) Controlling an External LED using a Raspberry Pi and GPIO pins. Disponível em <https://www.freecodecamp.org/news/hello-gpio-blinking-led-using-raspberry-pi-zero-wh-65af81718c14/>. Acesso em 21 mai. 2022.

[Gronholm, Alex]. (2021) Documentação apscheduler. Disponível em: <https://apscheduler.readthedocs.io/en/3.x/userguide.html>. Acesso em 14 jun. 2022.

[Russel, M; Morgado, A.] (2022) Manual MMCLI. Disponível em: <https://www.freedesktop.org/software/ModemManager/man/latest/mmcli.1.html>. Acesso em: 14 jun. 2022.

[Huurdeman, A.] (2003) The Worldwide History of Telecommunications. Local de publicação: John Wiley & Sons, 31 de julho de 2003.

[Aboubakar, Moussa; Kellil, Mounir; Roux, Pierre.] (2021) A review of IoT network management: Current status and perspectives. Journal of King Saud University-Computer and Information Sciences,

[Kumar, R.; Rajasekaran, M. Pallikonda.] (2016) An IoT based patient monitoring system using raspberry Pi. In: 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16). IEEE, 2016. p. 1-4.

[Kashani, Mostafa Haghi et al.] (2021) A systematic review of IoT in healthcare: Applications, techniques, and trends. Journal of Network and Computer Applications, v. 192, p. 103164, 2021.

[Remep-Fln.] (2022) Boletim Informativo de Ações REMEP-FLN 2020-2022. Disponível em <https://remep.pop-sc.rnp.br/home/noticias/20220513/1436>. Acesso em: 10 jun. 2022.

[Newman, P.] (2020) IoT Report: How Internet of Things Technology Is Now Reaching Mainstream Companies and Consumers. Disponível em: <https://www.businessinsider.com/internet-of-things-report>. Acesso em: 09 jun. 2022.

[Santos, Diogo; Ferreira, João C.] (2019) IoT power monitoring system for smart environments. Sustainability, v. 11, n. 19, p. 5355, 2019.

[Ragnoli, Mattia et al.] (2020) An autonomous low-power LoRa-based flood-monitoring system. Journal of Low Power Electronics and Applications, v. 10, n. 2, p. 15, 2020.

[Redhat.] (2021) What is edge computing? Disponível em:

<https://www.redhat.com/en/topics/edge-computing/what-is-edge-computing>. Acesso em: 14 jun. 2022.

[Shufian, Abu et al.] (2019) Smart irrigation system with solar power and GSM technology. In: 2019 5th International Conference on Advances in Electrical Engineering (ICAEE). IEEE, 2019. p. 301-305.

[Bueno, Cleiton.] (2014) Raspberry PI – GPIO input com Python. Disponível em: <https://www.embarcados.com.br/raspberry-pi-gpio-modo-input-python/>. Acesso em: 27. Jun. 2022.

[The Gnome Project.] (2014) nmcli - command-line tool for controlling NetworkManager. Disponível em: <https://developer-old.gnome.org/NetworkManager/stable/nmcli.html>. Acesso em: 05. Jul. 2022.

[Rao, R. Nageswara; Sridhar, B.] (2018) IoT based smart crop-field monitoring and automation irrigation system. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, 2018. p. 478-483.

[Juniper Networks.] (2022) VPNs baseadas em rotas e baseadas em políticas com NAT-T. Disponível em: <https://www.juniper.net/documentation/br/pt/software/junos/vpn-ipsec/topics/topic-map/security-route-based-and-policy-based-vpns-with-nat-t.html>. Acesso em: 16 set. 2022.

[Zabbix.] (2022) Documentação Zabbix Trapper. Disponível em: <https://www.zabbix.com/documentation/current/pt/manual/config/items/itemtypes/trapper>. Acesso em: 14 jun. 2022.