UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

Flávio Garlet Reck

SISTEMA PARA INDICAÇÃO DE FALTAS NA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA ATRAVÉS DO MONITORAMENTO DE CHAVES FUSÍVEIS UTILIZANDO REDE DE COMUNICAÇÃO LORAWAN

Santa Maria, RS
2019

Flávio Garlet Reck

SISTEMA PARA INDICAÇÃO DE FALTAS NA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA ATRAVÉS DO MONITORAMENTO DE CHAVES FUSÍVEIS UTILIZANDO REDE DE COMUNICAÇÃO LORAWAN

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro Eletricista.**

ORIENTADOR: Prof. Mauricio Sperandio

Santa Maria, RS
2019

Flávio Garlet Reck

SISTEMA PARA INDICAÇÃO DE FALTAS NA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA ATRAVÉS DO MONITORAMENTO DE CHAVES FUSÍVEIS UTILIZANDO REDE DE COMUNICAÇÃO LORAWAN

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro Eletricista.**

Aprovado em 10 de dezembro de 2019:

Mauricio Sperandio, Dr. (UFSM)
(Presidente/Orientador)

Carlos Henrique Barriquello, Dr. (UFSM)

Lucas Maziero, Eng. (UFSM)

Santa Maria, RS
2019

AGRADECIMENTOS

Uma longa jornada se fez necessária para eu chegar aonde cheguei, e sem o apoio da minha família, principalmente minha mãe, Marcia Garlet Reck e meu pai, Carlos Renê Reck, certamente o caminho teria sido diferente. Nada das minhas conquistas é somente minha. Portanto, agradeço todos os dias por todo o apoio que vocês tiveram na minha vida.

Agradeço ao meu orientador, Dr. Mauricio Sperandio por ser um exemplo como profissional que pretendo levar para a vida, por todo o auxílio proporcionado durante o desenvolvimento deste trabalho.

Também agradeço imensamente todas as amizades que a UFSM me proporcionou, essenciais para um desenvolvimento pessoal e profissional. Em especial, a um irmão que a faculdade trouxe, Augusto Zanin Bertoletti, pelo companheirismo tanto em momentos bons, como ruins da graduação.

Aos professores e membros do GEDRE, grupo de pesquisa que fui membro durante parte da minha graduação, meu muito obrigado, principalmente ao Ms. Renan Rodrigo Duarte, pelo auxílio tanto no meu tempo de GEDRE, como fora dele, auxiliando no desenvolvimento do protótipo deste trabalho.

Aos amigos da Fox IoT, por todo companheirismo, dedicação e principalmente pela ajuda fornecida para a realização deste trabalho, principalmente pelo Eng. Lucas Maziero.

A Universidade Federal de Santa Maria, que mesmo com diversas restrições financeiras, consegue prestar um serviço de formação exemplar. Serei para sempre grato por ter uma educação de alto nível em uma instituição pública, gratuita e de qualidade.

Por fim, a todos que contribuíram de forma direta ou indiretamente na minha formação ao longo da graduação.

Este TCC eu só acredito vendo.
Eu vendo: "Não acredito"

(Autor desconhecido)

RESUMO

SISTEMA PARA INDICAÇÃO DE FALTAS NA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA ATRAVÉS DO MONITORAMENTO DE CHAVES FUSÍVEIS UTILIZANDO REDE DE COMUNICAÇÃO LORAWAN

AUTOR: Flávio Garlet Reck
ORIENTADOR: Mauricio Sperandio

Este trabalho apresenta uma solução para incertezas verificadas por parte de distribuidoras de energia em relação à localização de faltas ocorridas no sistema de distribuição da mesma. Através do monitoramento de chaves fusíveis, é possível enviar um dado no momento que o equipamento é desarmado por razão de uma falta. Para isso, é utilizada uma solução baseada em uma rede de comunicação LPWAN (do inglês Low Power Wide Area Network, uma rede de baixo consumo de potência). A utilização do protocolo LoRaWAN para comunicação garante uma grande área de cobertura, essencial para a operação na rede de distribuição de uma cidade. Com isso, é possível obter uma solução de baixo custo, grande abrangência e confiável. Através de um sistema supervisório, operadores de uma distribuidora de energia têm acesso à localização da chave fusível desarmada por decorrência de uma falta no momento que ela ocorre, tendo agilidade no despacho de equipes de manutenção, reduzindo tempo de operação, e consequentemente, reduzindo a possibilidade da ocorrência de multas por descumprimento de índices de qualidade por tempo de operação.

Palavras-chave: Indicação de falta. Rede de ditribuição. LoRaWAN

ABSTRACT

FAULT INDICATION SYSTEM IN DISTRIBUTION NETWORKS BY CUTOUT FUSE MONITORING USING LORAWAN COMMUNICATION

AUTHOR: Flávio Garlet Reck
ADVISOR: Mauricio Sperandio

This work presents a solution for the lack of information verified by power distribution companies regarding the location of faults that occurred in its distribution system. By the monitoring of cutout fuses, it is possible to send data the moment that the equipment is opened as a result of a fault. To achieve this solution, a Low Power Wide Area Network is used. The LoRaWAN protocol makes possible to achieve long-range communication, which is essential to the operation in a power distribution network of a city. With this solution, it is possible to obtain a low-cost solution, with a long-range operation and trustworthiness. A supervisory system is developed, so that the maintenance team can have access to the location of an opened cutout fuse, because of a fault in the moment that the fault occurs, providing speed to dispatch maintenance teams, reducing the operation time, and consequently, the possibility of fines because of low quality indexes.

Keywords: Fault indication. Distribution network. LoRaWAN

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

$SDEE$	Sistema de Distribuição de Energia Elétrica
$ANEEL$	Agência Nacional de Energia Elétrica
QEE	Qualidade de Energia Elétrica
DEC	Duração Equivalente de Interrupção por Unidade Consumidora
FEC	Frequência Equivalente de Interrupção por Unidade Consumidora
DIC	Frequência de Interrupção Individual por Unidade Consumidora
FIC	Frequência de Interrupção Individual por Unidade Consumidora
$DMIC$	Duração Máxima de Interrupção Contínua por Unidade Consumidora
SEL	Schweitzer Engineering Laboratories
$IEEE$	Instituto de Engenheiros Eletricistas e Eletrônicos
IDE	Ambiente Integral de Desenvolvimento
ISR	Rotina de Interpretação de Interrupção
$LPWAN$	Rede de Baixo Consumo de Potência
$Anatel$	Agência Nacional de Telecomunicações
SF	Fator de Espalhamento
IoT	Internet das Coisas
SPI	Interface Serial de Periféricos
TTN	The Things Network

SUMÁRIO

1 INTRODUÇÃO

1.1 FALTAS NO SISTEMA DE DISTRIBUIÇÃO

Os Sistemas de Distribuição de Energia Elétrica (SDEE) estão sujeitos constantemente a diversas causas de faltas, como por exemplo condições climáticas, falhas em equipamentos, acidentes de trânsito, etc. (LEE et al., 2004). Em estudo realizado pela UK Power Networks, operadora das redes de distribuição do Reino Unido, e apresentado em (PLET, 2012), foram avaliados as causas de faltas no SDEE. Esses dados estão disponíveis na Figura 1.1.

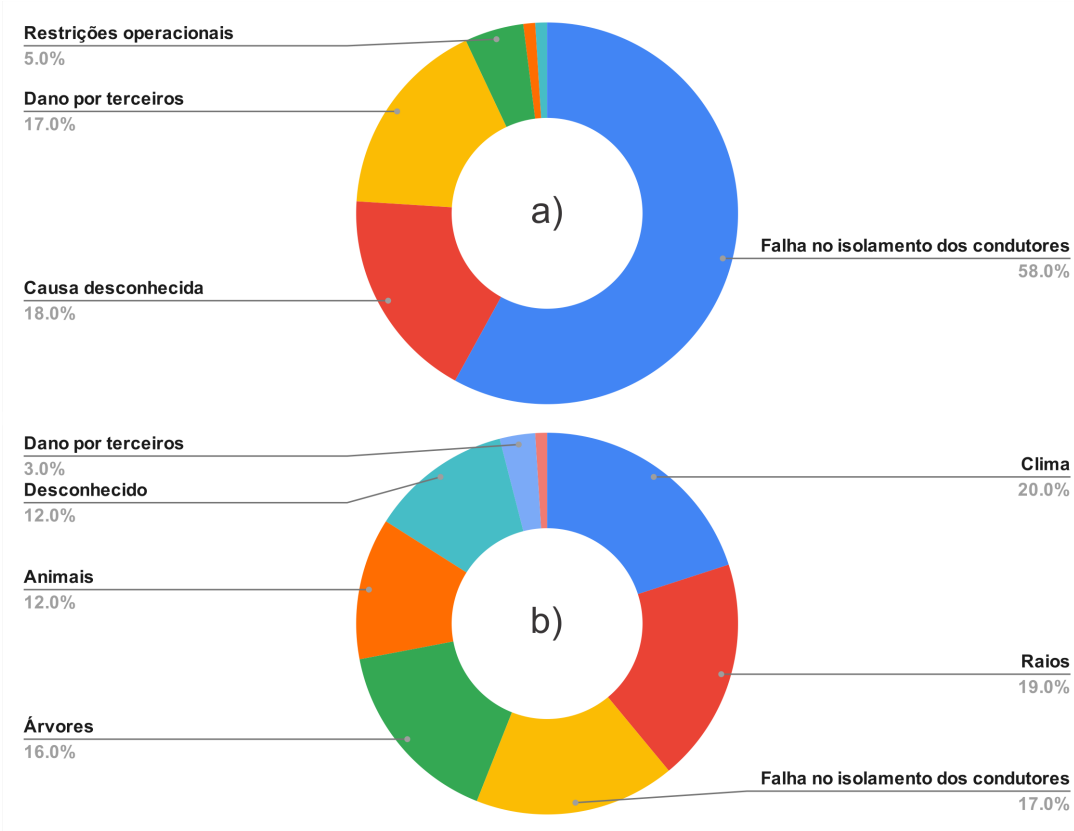


Figura 1.1 – Causas de faltas no sistema de distribuição a) Subterrânero e b) Aéreo

Fonte: Adaptado de Plet (2012).

Pode-se perceber que as causas de interrupções em sistemas aéreos, que representam a maioria dos sistemas no Brasil, pelo alto custo associado com uma rede subterrânea (VELASCO, 2003), são distribuídas e variadas. Além do mais, grande parte dessas faltas não pode ser evitada. Resta apenas com que a distribuidora de energia realize a manutenção corretiva, através de uma equipe de manutenção para o local da falta.

As faltas no SDEE podem ocorrer em uma, duas ou nas três fases. Faltas monofásicas são mais comumente verificadas, com quase 80% das faltas verificadas sendo monofásicas, entre o neutro ou entre o condutor de proteção (SHORT, 2018).

Na Tabela 1.1, é apresentado o número de fases envolvidas em faltas no SDEE.

Tabela 1.1 – Quantidade de fases nas faltas do SDEE.

Tipo de falta	**Porcentagem**
Uma fase - neutro	63%
Fase - fase	11%
Duas fases - neutro	2%
Três fases	2%
Uma fase - proteção	15%
Duas fases - proteção	2%
Três fases - proteção	1%
Outros	4%

Fonte: Adaptado de Short (2018).

Estas faltas geram a interrupção do fornecimento de energia elétrica para diversos consumidores. Como forma de regulamentação do setor, a Agência Nacional de Energia Elétrica (ANEEL), determina níveis para a Qualidade de Energia Elétrica (QEE). Os principais indicadores são (ANEEL, 2018):

- DEC (Duração Equivalente de Interrupção por Unidade Consumidora): Indica a média do número de horas que o consumidor de determinado conjunto esteve sem o fornecimento de energia;
- FEC (Frequência Equivalente de Interrupção por Unidade Consumidora): Indica a média da quantidade de interrupções do fornecimento em um determinado período;
- DIC (Duração de Interrupção Individual por Unidade Consumidora): Duração da interrupção;
- FIC (Frequência de Interrupção Individual por Unidade Consumidora): Quantidade de interrupções verificadas;
- DMIC (Duração Máxima de Interrupção Contínua por Unidade Consumidora): Duração da maior interrupção durante um período.

Desde 2010, a ANEEL disponibiliza dados de compensação por ultrapassagem dos índices de qualidade. Este valor já ultrapassou a marca de R$ 4 Bi. Na Figura 1.2, está representada a evolução da compensação por ultrapassagem dos índices de qualidade no período de 2010 até Outubro de 2019 (ANEEL, 2019).

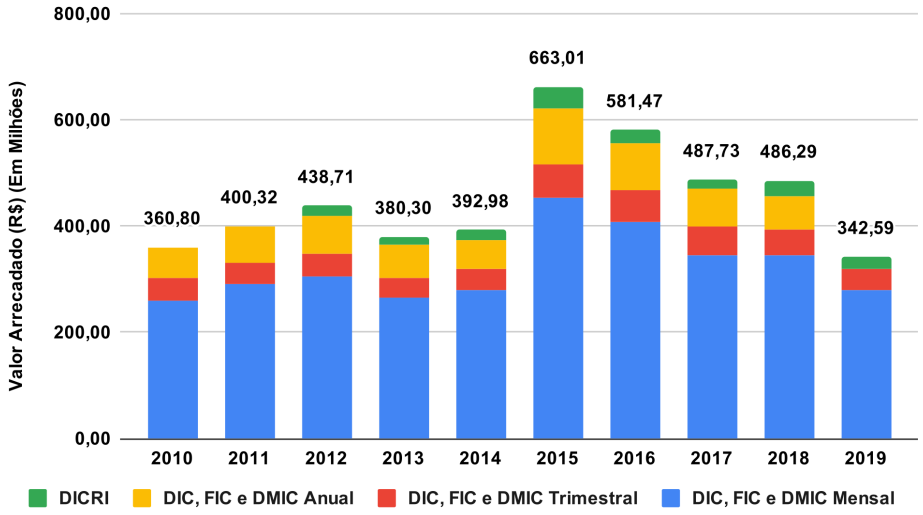


Figura 1.2 – Compensação por ultrapassagem de indicadores de continuidade para o período de 2010 a 2019

Fonte: Adaptado de ANEEL (2019).

Portanto, no desenvolvimento de um sistema de indicação de faltas, é possível melhorar os índices de qualidade referentes à duração das interrupções no fornecimento, apresentando ganhos tanto para a concessionária como para o consumidor final. Em se tratando da concessionária, a redução na interrupção do fornecimento acarreta em um uma redução nas multas referentes à ultrapassagem. Já para o consumidor, é possível verificar um melhor fornecimento de energia, com a redução no tempo das interrupções.

1.2 MOTIVAÇÃO

Conforme discutido na Seção 1.1, o não cumprimento com os índices de continuidade estabelecidos pela ANEEL são responsáveis por grande montante por parte de concessionárias e permissionárias. Como exemplo, é analisado o caso da RGE Sul, que atua na região de Santa Maria - RS. Em 2018, para a região sul, nas cidades onde possuem atendimento pela RGE Sul, o número total de compensações por ultrapassagem de DIC, FIC e DMIC foi de 2.240.888 ocorrências, totalizando R$ 22.127.676,75 (ANEEL, 2019).

Atualmente, o processo de despacho de manutenção quando uma interrupção do fornecimento está representada na Figura 1.3. Quando o consumidor tem seu fornecimento de energia interrompido, ele deve recorrer aos meios de comunicação da concessionária para informar o ocorrido, seja por telefone, site ou aplicativo. Porém a concessionária deve esperar a reclamação de diversos consumidores para despachar sua equipe de manutenção, que ainda assim não possui a localização exata do problema (CAPELINI et

al., 2016). Cada etapa desse processo representa um tempo gasto, contabilizado pelo DIC e consequentemente, sendo possível a ultrapassagem dos limites impostos pela ANEEL, resultando em multas para a concessionária.

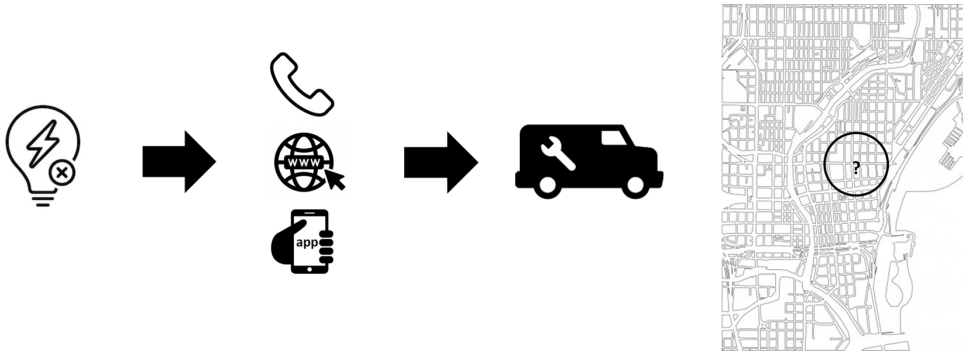


Figura 1.3 – Sequência de eventos quando uma interrupção no fornecimento ocorre.

Fonte: Autor.

Com isso, a motivação principal do trabalho é reduzir o tempo necessário para a concessionária despachar sua equipe de manutenção no caso de ocorrência de uma interrupção de serviço. Através de um sistema supervisório, é possível verificar no momento em que uma chave fusível abriu, assim como sua localização, permitindo assim um deslocamento correto da equipe de manutenção para o local exato, reduzindo assim a duração das interrupções, e consequentemente a multa por ultrapassagem de valores limites, definidos pela ANEEL.

1.3 OBJETIVOS

1.3.1 Objetivos Gerais

Tem-se como objetivo geral o desenvolvimento de uma solução para a indicação de faltas no SDEE. Devido ao tamanho e ao número de usuários do sistema, a solução deve possibilitar a implementação:

- com grande área de cobertura;
- com baixo consumo de potência, para possibilitar o uso com baterias;
- de baixo custo.

1.3.2 Objetivos Específicos

- Propor uma solução para o monitoramento de chaves fusíveis através de um sensor magnético;
- Integração do sensor magnético com o microprocessador ATmega328p;
- Envio dos dados através de uma rede de baixo consumo (LPWAN) através do módulo RFM95;
- Implementação de técnicas de sono profundo do microprocessador e do módulo de comunicação para operação com bateria;
- Manutenção da vida útil das baterias através de carregamento solar por mini painéis fotovoltaicos;
- Desenvolvimento de um sistema supervisório para o monitoramento remoto.

1.4 ESTRUTURA DO TRABALHO

O trabalho se dará da seguinte maneira: No Capítulo 2, será abordado o desenvolvimento teórico do trabalho, primeiramente através de uma revisão bibliográfica sobre os sistemas indicadores de faltas verificados atualmente no mercado. Após, um resumo do funcionamento das chaves fusíveis, através do monitoramento delas que será possível realizar a indicação de faltas. Em seguida, será abordado o microcontrolador utilizado no trabalho, em especial como ele lida com interrupções e gerenciamento de energia. Também será feita uma revisão sobre a rede de comunicação utilizada para o desenvolvimento de um protótipo, assim como o sistema supervisório para acompanhamento dos dados.

No Capítulo 3, serão abordados os passos práticos tomados para a construção de um protótipo, pela descrição do *hardware* utilizado no sistema, como por exemplo o microcontrolador, sensor e módulo de comunicação, assim como o carregamento de baterias através de uma placa solar. O sistema como um todo, assim como o supervisório é descrito em funcionamento.

No Capítulo 4, serão apresentados os resultados através do desenvolvimento de um protótipo de dispositivo, assim como um sistema supervisório. Por fim, no Capítulo 5 são apresentadas as conclusões do trabalho.

2 DESENVOLVIMENTO TEÓRICO

2.1 SISTEMAS INDICADORES DE FALTAS

Os sistemas indicadores de faltas no SDEE foram inicialmente verificados na Alemanha, em 1946. Inicialmente, esses dispositivos acionavam uma chave mecânica quando a corrente medida ultrapassava um valor definido. Assim, era necessário com que a equipe de manutenção acompanhasse o circuito, o ponto entre o último indicador com a chave mecânica ativa e o primeiro com a mesma desligada seria o ponto em que a falta ocorreu (ANGERER, 2008). A Figura 2.1 apresenta um desses dispositivos.

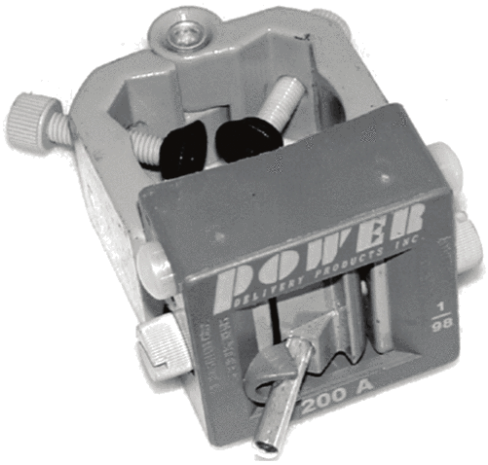


Figura 2.1 – Indicador de falta da primeira geração

Fonte: (ANGERER, 2008)

O princípio de funcionamento de indicadores de falta está em aplicação até o presente momento. A Figura 2.2 apresenta um indicador de falta desenvolvido pela SEL - Schweitzer Engineering Laboratories. Nota-se que a principal diferença está na forma de identificação, que alterou-se de uma chave mecânica para um conjunto de lâmpadas. Porém seu preço de catálogo é de US$ 212,00, o que inviabiliza a aplicação em grande escala numa cidade para uma solução completa de monitoramento. Também, ainda recorre na questão da dependência da informação do consumidor que ele se encontra sem energia na sua residência.

O Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) possui normas em relação a maneira de que dispositivos indicadores de faltas devem apresentar comportamento, através do padrão IEEE Std 1610-2016 (IEEE, 2016), dentre as normativas, estão os modos com que os indicadores devem voltar ao estado normal.

- De forma manual: um operador deve manualmente resetar o dispositivo;

Figura 2.2 – Indicador de falta aéreo AR360

Fonte: (SEL, 2019)

- De forma automática: podendo ser por corrente, tensão ou tempo. Esta forma de configuração teve seu início na década de 70 (ANGERER, 2008);

  Corrente: Assim que a corrente de carga é verificada, o dispositivo volta ao estado normal;

  Tensão: Estes dispositivos monitoram a presença de um campo eletrostático, para indicar quando o sistema retornou a operação normal;

  Tempo: O dispositivo voltará ao estado inicial após um determinado tempo independente do estado de tensão e corrente.

Uma possível solução para a indicação de faltas é o monitoramento de chaves fusíveis. Assim que uma chave é aberta, um sinal é enviado para a concessionária, com isso, a equipe de manutenção tem acesso ao ramo de onde uma interrupção do serviço foi verificada, limitando o trecho que deve ser percorrido para localizar a falha.

Antes, é necessário compreender o funcionamento das chaves fusíveis, afim de propor uma solução para o seu monitoramento. O funcionamento de tais dispositivos é tratado na seção 2.2.

2.2 CHAVES FUSÍVEIS

A proteção de sobrecorrentes é de fundamental importância, não sendo exceção no SDEE. Como exemplos de dispositivos usados para a proteção, podemos citar disjuntores, religadores e chaves fusíveis (SHORT, 2018).

As chaves fusíveis são dispositivos de proteção utilizados em linhas aéreas, em sua maioria, para fornecer proteção para transformadores de distribuição quando é verificado um surto de potência. Seu funcionamento se dá através de uma seção de cabo em espessura menor (fusível). Assim, quando é verificada uma sobrecorrente, o calor gerado pela mesma derrete o fusível, protegendo assim os dispositivos seguintes do circuito (MOLLET, 1995). Uma chave fusível tradicional pode ser vista na Figura 2.3, este modelo, em específico, conforme catálogo do fabricante, possui variações de isolação de 15, 27 e 38 kV com correntes nominais de 100 ou 200 A.

Figura 2.3 – Chave fusível genérica

Fonte: (MAURIZIO, 2016)

Quando uma chave fusível é aberta, devido à interrupção do caminho da corrente, há a possibilidade de criação de arco elétrico. Dentro do porta fusível, o mesmo é preenchido com um material arenoso, que ajuda a extinguir o arco, tornando assim uma opção viável de proteção (SHORT, 2018).

A filosofia de proteção para distribuição difere para a proteção de sistemas de transmissão ou industriais. A coordenação dos elementos em série deve ser feita (SHORT, 2018):

- elo a elo;
- religador com elo;
- disjuntor com religador ou elo

Uma correta coordenação é recomendada para evitar a abertura incorreta de chaves, o que resulta na interrupção do fornecimento de mais consumidores, o que apresenta um desafio para a equipe de manutenção para localizar a chave aberta.

2.3 PLATAFORMA ARDUINO

Com o objetivo do trabalho definido: o monitoramento da abertura de chaves fusíveis no SDEE. O próximo passo do trabalho é a definição do microcontrolador utilizado para a realização do monitoramento.

Conforme discutido na seção 1.3.1, é necessário uma solução simples, para manter um baixo custo de produção. Para isso, escolheu-se uma solução através do microcontrolador ATmega328p, em sua placa de desenvolvimento Arduino Pro Mini (ARDUINO, 2019).

Arduino é uma plataforma de código aberto com o intuito de prover soluções de fácil uso tanto para *hardware* como para *software*, usado na construção e programação de dispositivos eletrônicos. Com ele, é possível receber e mandar informações para uma grande variedade de dispositivos ou plataformas (BANZI; SHILOH, 2014).

Sistemas baseados com placas Arduino têm a possibilidade de ler entradas de sensores, cliques de botão, transformar sinais em saídas, etc. (ARDUINO, 2019). Sua estrutura é composta de duas partes distintas: a placa de desenvolvimento, ou seja, o *hardware* no qual os sensores/atuadores serão adicionados e também o Ambiente Integral de Desenvolvimento (IDE, do inglês *Integrated Development Environment*), que é o *software* utilizado para realizar a programação da placa. Essa programação dita como o *hardware* deve se comportar (BANZI; SHILOH, 2014). A programação através da IDE é via linguagem de programação Arduino, baseada na linguagem *Processing* (BANZI; SHILOH, 2014).

2.3.1 Hardware

A placa escolhida para o desenvolvimento do trabalho é a versão Pro Mini, principalmente por seu tamanho reduzido e baixo consumo de potência. Ela conta com um microprocessador ATmega 328p, o mesmo presente em sua versão Uno. A Figura 2.4 apresenta a placa utilizada.

Na Tabela 2.1, é possível verificar as especificações do módulo. Percebe-se que para a aplicação desejada, que será usado apenas um sensor magnético para a detecção da abertura da chave, usa-se apenas um pino de interrupção. Além disso, é possível o uso de bibliotecas, que são extensões de funcionalidade para aplicações específicas (ARDUINO, 2019), para a aplicação deste trabalho, será utilizada a biblioteca *Low-Power* (ROCKETSCREAM, 2019), que aplica técnicas de gerenciamento de energia, possibilitando o sistema a consumir micro-amperes quando hibernando. O uso dessa biblioteca será explicado durante o desenvolvimento prático do dispositivo.

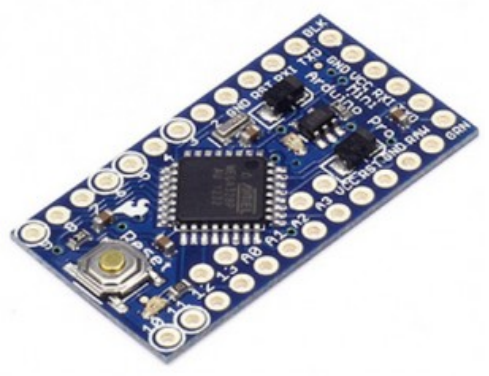


Figura 2.4 – Arduino Pro Mini

Fonte: (ARDUINO, 2019)

2.3.2 Software

A IDE do Arduino é um *software* que deve ser instalado em um computador que permite a escrita de programas em uma linguagem de programação simples. Quando o programa é enviado para a placa através da IDE, o código escrito é traduzido para a linguagem C, que é passada para o compilador *avr-gcc*, peça importante de *software* livre, responsável pela tradução final do código para linguagem de máquina (BANZI; SHILOH, 2014).

Na Figura 2.5, é apresentada a tela inicial da IDE do Arduino. Seu uso é muito simples após a configuração da placa utilizada e a porta serial na qual a comunicação será feita. Na barra superior, o primeiro botão (*Verify*) realiza a verificação do código para checar se há erros, e o segundo (*Upload*) envia o código para a placa.

Uma peça chave no desenvolvimento desse trabalho é a forma com que o dado referente à abertura da chave será enviado. Ou seja, como realizar o monitoramento de uma forma simples e que consiga retirar o microprocessador do estado de sono. Isso é realizado através de uma interrupção, que será abordada na seção 2.3.2.1, a seguir.

2.3.2.1 Interrupções

As interrupções são uma seção de *hardware* e *software* de um microcontrolador que é capaz de monitorar eventos externos em um pino. Quando são verificados, a execução do programa é interrompida, que dá lugar para uma rotina de interpretação da interrupção (ISR, do inglês *Interrupt Service Routine*). Após, o programa volta a ser executado a partir do ponto onde foi interrompido (MICROCONTROLLERSLAB, 2018). Seu princípio está demonstrado na Figura 2.6.

A única maneira de retirar o microcontrolador do estado de sono por eventos ex-

Tabela 2.1 – Especificações do módulo Arduino Pro Mini

Microcontrolador	ATmega328p
Alimentação da placa	5 - 12 V
Tensão de operação	5V
Pinos de entrada/saída digitais	14
Pinos de PWM	6
UART	1
SPI	1
I2C	1
Pinos de entrada analógicos	6
Interrupções externas	2
Máxima corrente por pino de entrada/saída digital	40 mA
Memória flash	32KB, das quais 2KB são usados pelo bootloader
Memória SRAM	2 KB
Memória EEPROM	1 KB
Velocidade de clock	16 MHz

Fonte: Adaptado de Arduino (2019).

ternos é através de interrupções (ROCKETSCREAM, 2019), portanto, seu uso será de extrema importância para o desenvolvimento do trabalho.

Após, é necessário definir a rede de comunicação que será utilizada para o envio dos dados pelo monitoramento de chaves remotas, através de um microcontrolador baseado na plataforma Arduino, através de uma interrupção. Isso será abordado na Seção 2.4, a seguir.

2.4 REDE DE COMUNICAÇÃO

A escolha da rede de comunicação a ser utilizada é de fundamental importância, devido principalmente a grande área coberta pelas redes de distribuição em cidades. Além do mais, também é necessário uma opção de baixo consumo de potência, para viabilizar o uso de baterias, e por último, não há a necessidade de uma alta taxa de transmissão de dados, pois só é necessário enviar um bit referente ao estado da chave e outro para identificar o funcionamento pleno do sistema.

Afim de realizar a comunicação entre o dispositivo indicador de faltas e uma central (nesse caso, um concentrador - *gateway*), uma rede de baixo consumo (LPWAN, do inglês Low Power Wide Area Network) foi escolhida. Em comparação com outras redes, como por exemplo *Zigbee*, *Wi-Fi* e *Bluetooth*, as redes LPWAN apresentam vantagens necessárias para o contexto desse trabalho (MUTHANNA et al., 2018). As redes LPWAN

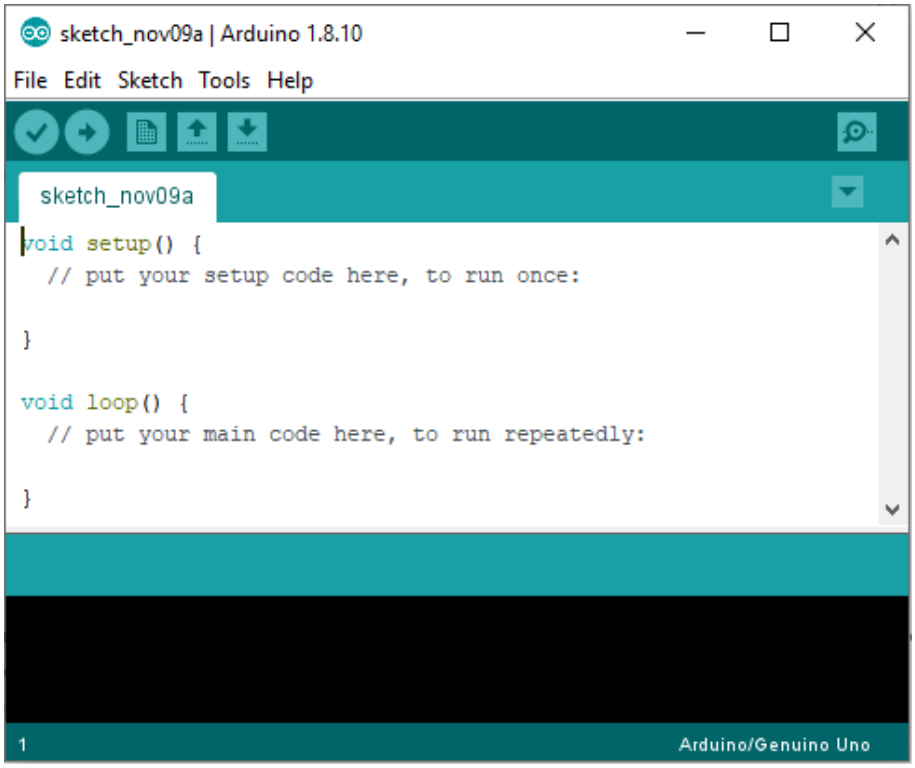


Figura 2.5 – IDE do Arduino

Fonte: Autor.

apresentam comprometimentos diferentes do que outras tecnologias sem fio tradicionais, principalmente referente à área de cobertura, tecnologias tradicionais se estendem geralmente questão de dezenas de metros nos melhores dos casos (RAZA; KULKARNI; SOORIYABANDARA, 2017). Pelo tamanho que uma rede de distribuição de uma cidade pode apresentar, consistindo de várias transformadores de distribuição e chaves fusíveis em uma grande área de atuação, uma operação de longa distância se faz necessária. Soluções baseadas em redes de celular apresentam ótima cobertura em ambientes urbanos, porém seu consumo de potência é elevado. Como exemplo, temos o módulo SIM800L, que tem um consumo máximo de 2 A quando em modo de transmissão (NADA, a), além de que é uma opção custosa, já que cada dispositivo deverá ter um cartão SIM com um plano de internet. Na Figura 2.7 é apresentado uma comparação entre as redes LPWAN e as tecnologias de comunicação tradicionais.

Dentro do contexto das redes LPWAN, existem duas grandes tecnologias de rede: LoRa e Sigfox, ambas apresentam similaridades, porém com uma diferença fundamental em respeito da conexão do nó para a aplicação: o uso de uma rede LoRa permite uma conexão inteiramente proprietária, o que significa que o desenvolvedor possui completo acesso e responsabilidade pelo tráfego. Sigfox, por outro lado, utiliza protocolos pertencentes à empresa, e portanto, a conexão não é proprietária do desenvolvedor. Em um primeiro momento, esse fator não parece representar um grande problema em uma ope-

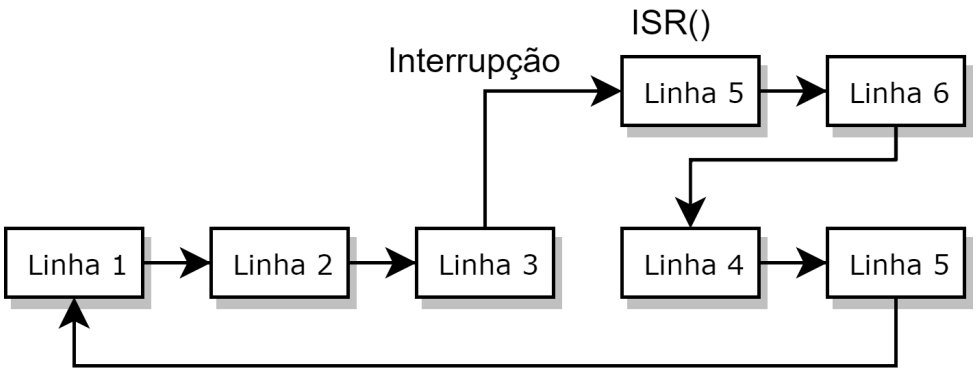


Figura 2.6 – IDE do Arduino

Fonte: Adaptado de microcontrollerslab (2018).

ração normal. Porém, vários relatos de usuários no fórum oficial (SIGFOX, 2015) relatam uma falta de responsabilidade por parte da empresa, o que significa que o desenvolvedor não tem o total controle da solução desenvolvida, pelo tráfego da rede, dificultando no processo de manutenção (ANUPRIYA et al., 2016). Por isso, uma rede LoRa com o protocolo LoRaWAN foi escolhida para o desenvolvimento deste trabalho. Um ponto que merece destaque é referente à diferença entre LoRa e LoRaWAN, o que costuma gerar equívocos. LoRa é a camada física de rádio frequência. Nela, um dos protocolos mais comumente utilizado é o LoRaWAN, que rege parâmetros de comunicação, encriptação, etc.

As redes LPWAN apresentam algumas características em comum:

- Longo alcance: As redes operam nas frequências sub-GHz. No Brasil, o uso de redes LoRa não é licenciado, mas sim regulamentado pela Agência Nacional de Telecomunicações (Anatel) através da resolução 680/17, que regulamenta frequências para a transmissão de dados. O uso do plano australiano para o protocolo LoRaWAN é compatível com essa regulamentação, o protocolo AU915 com frequência centrada em 915 MHz (ANATEL, 2017). Essa modulação de sinal, provida pela camada física LoRa, baseada em modulação por *chirps*, permite operação de até 12 km em ambientes abertos (VANGELISTA, 2017);

- Baixa transmissão de dados: Sistemas tradicionais sem fio, como por exemplo o *Wi-Fi* apresentam taxas de transmissão de centenas de Mbps. Já as redes LPWAN apresentam taxas limitadas de até 100 kbps (CHEN et al., 2017). Através do protocolo LoRaWAN, por exemplo, é possível a transmissão de dados em no máximo 27 kbps, valor razoável para aplicações que não necessitam grande tráfego de dados;

- Baixo consumo energético: O protocolo LoRaWAN define classes nas quais os dispositivos podem operar, geralmente a operação se dá na Classe A, o que significa que a transmissão só ocorre quando há um evento agendado para um determinado

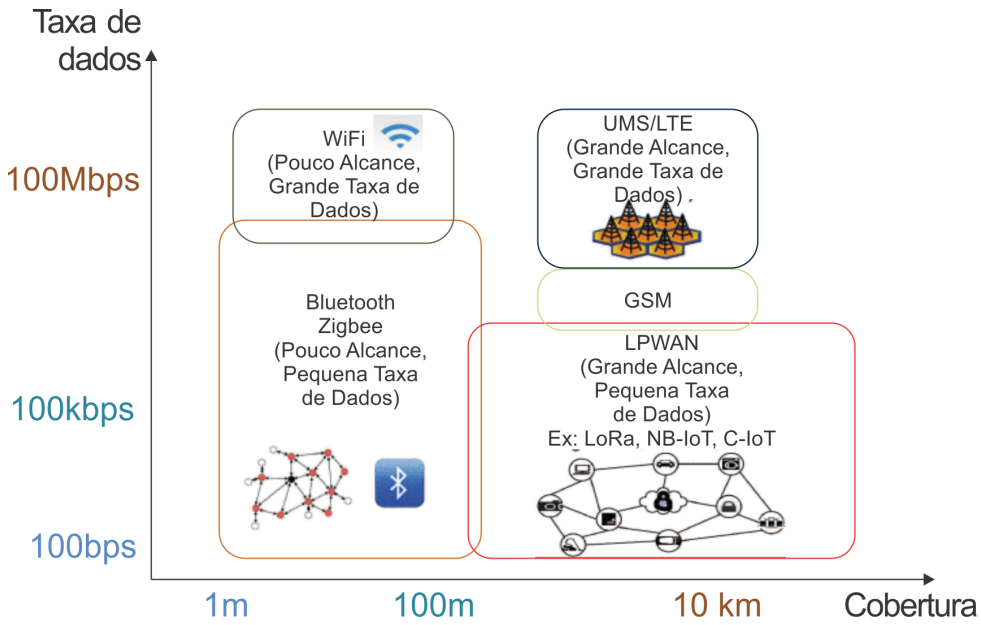


Figura 2.7 – Comparação de redes de comunicação

Fonte: Adaptado de Chen et al. (2017).

momento. A recepção (Rx) ocorre apenas após uma transmissão (Tx), essa operação gera uma baixíssimo consumo de energia, já que quando o dispositivo não está enviando, o módulo de comunicação entra em estado de sono automaticamente. Ao contrário de outras arquiteturas, como por exemplo GSMA/LTE, não há a necessidade de sincronização dos dispositivos frequentemente com a rede, o que seria responsável por um grande consumo de potência (ALLIANCE, 2015).

Na Seção 2.4.1, é apresentada como a rede é organizada, assim como é detalhado a função de cada ponto responsável na transmissão.

2.4.1 Estrutura da Rede

A organização de uma rede LoRa se dá através de uma topologia de estrela. Nessa estrutura, os nós se conectam com um servidor central (chamado de *gateway*) através do protocolo LoRaWAN. Os *gateways*, por sua vez, estão conectados a um servidor de rede através de uma rede não-LoRaWAN, geralmente *Wi-Fi*, *Ethernet*, *3G* ou *4G* (ADELANTADO et al., 2017). Todos os dados que são enviados através do protocolo LoRaWAN possuem encriptação AES-128, o que significa que apenas a aplicação que possui as chaves corretas conseguem decifrar os dados. A arquitetura de uma rede LoRa com protocolo LoRaWAN está apresentada na Figura 2.8.

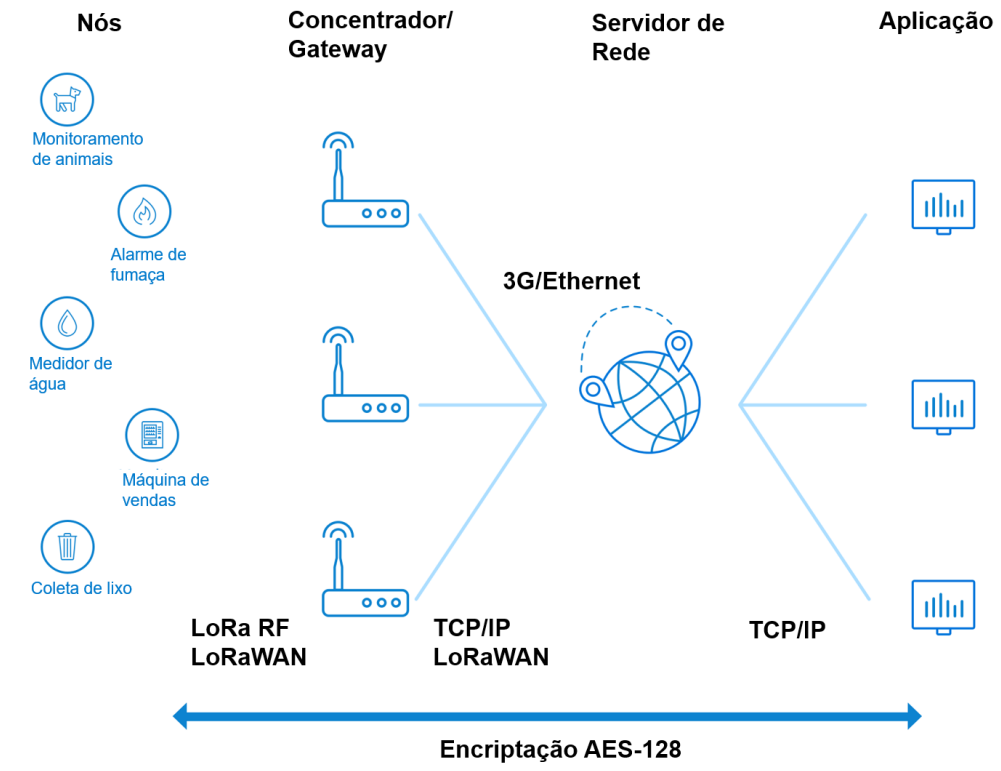


Figura 2.8 – Estrutura de uma rede LoRaWAN

Fonte: Adaptado de TTN (2019).

2.4.1.1 Nós

Em uma solução baseada em LoRaWAN, nós são os dispositivos finais da aplicação. São objetos que contam com um dispositivo de comunicação de baixo consumo (TTN, 2019). Estes dispositivos podem ser para as mais variadas aplicações, como por exemplo o monitoramento de animais, alarmes de incêndio, medidores de água, máquinas de vendas, verificação de coleta de lixo, etc.

2.4.1.2 Concentradores

São os dispositivos que realizam uma ponte entre os nós e o servidor de rede, recebendo os dados enviados pelos nós e os repassando para o servidor de rede, para futuramente serem interpretados. Esses dispositivos, ao contrário dos nós, possuem comunicação com protocolos de altas taxas de transferência, como por exemplo *Wi-Fi*, *Ethernet* ou *4G* para a comunicação com servidores de rede devido ao grande número de dispositivos que enviam informações para o mesmo (TTN, 2019).

A presença de diversos concentradores em uma área traz uma melhor confiabi-

lidade para o sistema, já que se mais de um *concentrador* receber o mesmo dado, o servidor de rede realiza a escolha do dado que foi recebido com o melhor sinal para ser decodificado. Um único concentrador pode atender milhares de dispositivos (TTN, 2019).

2.4.1.3 Servidor de Rede

Uma cobertura LoRa em uma cidade permite o desenvolvimento de várias soluções, o servidor de rede é o responsável por realizar a separação dos dados para as aplicações corretas. Por exemplo, repassar somente os dados de medidores específicos para a aplicação correta (TTN, 2019).

2.4.1.4 Aplicação

Por fim, a aplicação é um *software* desenvolvido para decodificar o dado e tratá-lo como desejado, seja realizando o armazenamento, o exibindo em um gráfico ou até mesmo usando o dado para controlar algum dispositivo (TTN, 2019). No caso deste trabalho, a aplicação é responsável por identificar a localização do medidor que detectou a abertura de uma chave fusível.

2.4.2 Cobertura

A utilização de uma rede de comunicação utilizando LoRaWAN permite a operação com uma grande cobertura. Em ambientes abertos, verifica-se uma cobertura de até 12km (VANGELISTA, 2017), enquanto em ambientes urbanos, com apenas um *Gateway*, já foi verificada uma operação de 2,2 km (WIXTED et al., 2016). Com a presença de diversos *gateways* em uma cidade, é possível prover uma solução completa com cobertura plena em toda cidade.

Uma rede com o protocolo LoRaWAN permite a transmissão de dados entre 0,3 e 27 kbps. Pode ser considerado uma pequena taxa, mas para o contexto deste trabalho, onde apenas alguns *bytes* necessitam ser enviados, é uma boa opção considerando a cobertura oferecida. Outro parâmetro determinado pelo protocolo que tem impacto direto na área de cobertura é o Fator de Espalhamento (SF, do inglês *Spreading Factor*), que é um conjunto de parâmetros que especificam a potência transmitida, frequências secundárias e tempo de dado no ar. Existe um comprometimento direto entre SF e a área de cobertura. Quanto maior o SF, maior a área de transmissão e menor a taxa de bits transmitidos (SORNIN et al., 2015). Valores de SF podem variar entre 7 e 12, representados na Figura 2.9.

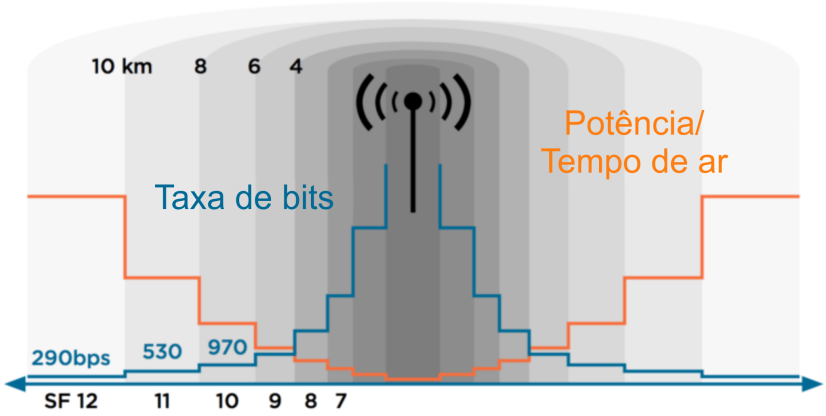


Figura 2.9 – Fatores de Espalhamento LoRaWAN

Fonte: Adaptado de Sornin et al. (2015).

2.5 SISTEMA SUPERVISÓRIO

Para realizar o monitoramento do estado das chaves fusíveis em uma rede de distribuição de uma cidade, é necessário desenvolver um sistema para que, de forma gráfica, a equipe de manutenção consiga ter acesso a localização do equipamento aberto. Para isso, diversos serviços são integrados para que uma plataforma seja desenvolvida. Primeiramente, O *software Telegraf* é responsável por repassar os dados enviados pelos dispositivos identificadores de falta para um banco de dados, que é gerido pelo *software* InfluxDB. Por fim, uma interface gráfica é desenvolvida através do *software* Grafana, responsável por acessar os dados do banco e identificar através de alertas visuais que uma chave fusível encontra-se aberta.

Todas as ferramentas aqui utilizadas são de *software* livre. A seguir, será feita uma breve descrição e sobre a aplicação de cada elemento responsável pela construção de um sistema supervisório final.

2.5.1 Telegraf

Telegraf é um módulo para a implementação da coleta de dados de dispositivos, arquivos ou até outros bancos de dados, usado pra transferir seus dados para o banco de dados temporais InfluxDB (KYCHKIN et al., 2019).

É usado para a coleta e interpretação dos seguintes dados (TELEGRAF, 2019):

- Bancos de dados: Conecta fontes de dados como por exemplo *MongoDB*, *MySQL*, *Redis*, entre outros, para a coleta e envio de métricas;

- Sistemas: Coleta de métricas de plataformas em nuvem, contâineres e outros orquestradores;
- Sensores IoT: Coleta de dados críticos (níveis de pressão, temperatura, etc.) de sensores e dispositivos da Internet das Coisas (IoT, do inglês Internet of Things).

2.5.2 InfluxDB

O InfluxDB é uma solução de código aberto para bancos de dados, fornecendo o armazenamento de medidas automaticamente (KYCHKIN et al., 2019). Com ele, é possível gerenciar um banco de dados pois o mesmo insere o instante de tempo no qual o dado foi recebido pelo banco, sendo possível a interpretação em formato de gráficos.

InfluxDB e Telegraf pertencem ao mesmo desenvolvedor, possuindo então uma compatibilidade plena. Ambos fazem parte de um conjunto de soluções para interpretação de dados.

2.5.3 Grafana

Grafana é um servidor de internet que implementa a relação homem-máquina, gerando interface gráfica para a visualização de dados temporais, os dados exibidos podem ser operados, como por exemplo a realizações de cálculos, e até mesmo o uso de alarmes, com notificações por email, por exemplo (KYCHKIN et al., 2019).

O objetivo do uso do Grafana é a criação de uma página contendo um mapa com todos os pontos sendo monitorados, e no momento que uma chave fusível encontrar-se aberta, o envio do dado de abertura provoca um alarme no sistema, indicando a localização do dispositivo.

Com isso, um resumo dos *softwares* utilizados para as métricas está apresentado na Figura 2.10.

2.6 SISTEMA COMPLETO

Com a definição dos materiais e métodos que serão utilizados, é possível resumir o funcionamento completo do sistema. No total, serão duas situações em que o dispositivo identificador de abertura de chave fusível enviará um dado:

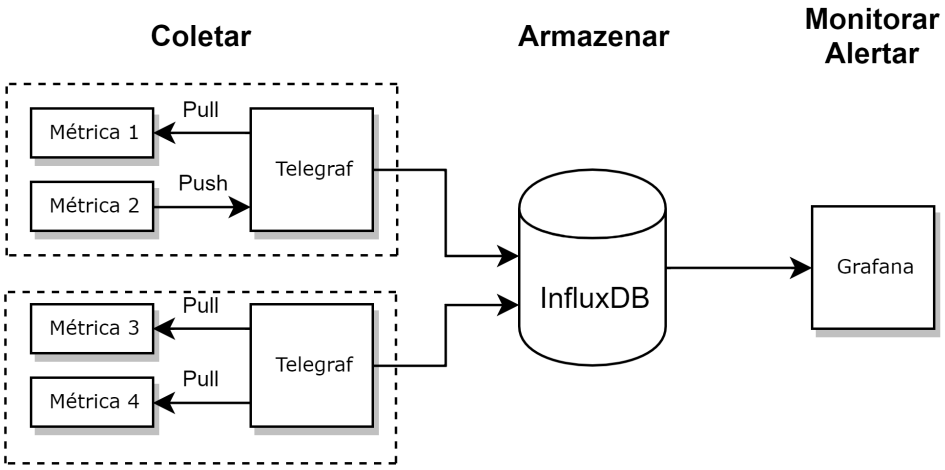


Figura 2.10 – Resumo dos *softwares* utilizados.

Fonte: Adaptado de Bordier (2017).

- Quando a chave fusível encontrar-se aberta (com repetição por um intervalo programado de envios até a chave ser fechada);
- Regularmente em um intervalo de tempo, como forma de aumentar a confiabilidade do sistema, assim, é possível certificar-se de que as baterias estão em pleno funcionamento.

Portanto, quando um dado é enviado, através de uma rede de comunicação LoRa, através do protocolo LoRaWAN, esse dado é enviado do nó até um *gateway*, que através do servidor de rede e aplicação, é decodificado e repassado esse dado pelo *Telegraf* para o *InfluxDB*. O dado é então armazenado em um banco de dados, e graficamente, um alerta é criado em uma interface gráfica através do *Grafana*. Assim, a equipe de manutenção tem agilidade e certeza da localização da chave fusível que foi aberta. Essa sequência de eventos é esquematizada na Figura 2.11.

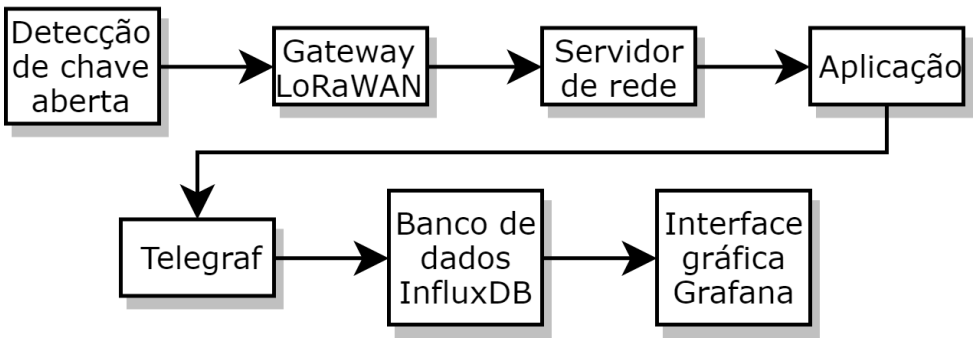


Figura 2.11 – Esquema de eventos durante uma transmissão de dados

Fonte: Autor.

No Capítulo 3, será abordado o desenvolvimento prático do trabalho, como por exemplo a integração dos eventos descritos na Figura 2.11 em um protótipo.

3 DESENVOLVIMENTO PRÁTICO

Neste capítulo, serão apresentados os materiais e a forma como o sistema foi desenvolvido e implementado, em todo momento tendo como meta o desenvolvimento de uma solução de baixo custo, afim de atingir o objetivo do trabalho, que é o desenvolvimento de uma solução para identificação de abertura de chaves fusíveis.

3.1 DESCRIÇÃO DO *HARDWARE* UTILIZADO NO SISTEMA

De início, foi desenvolvido um diagrama de blocos contendo os elementos de *hardware* da solução proposta para melhor visualização. Este está disponível na Figura 3.1.

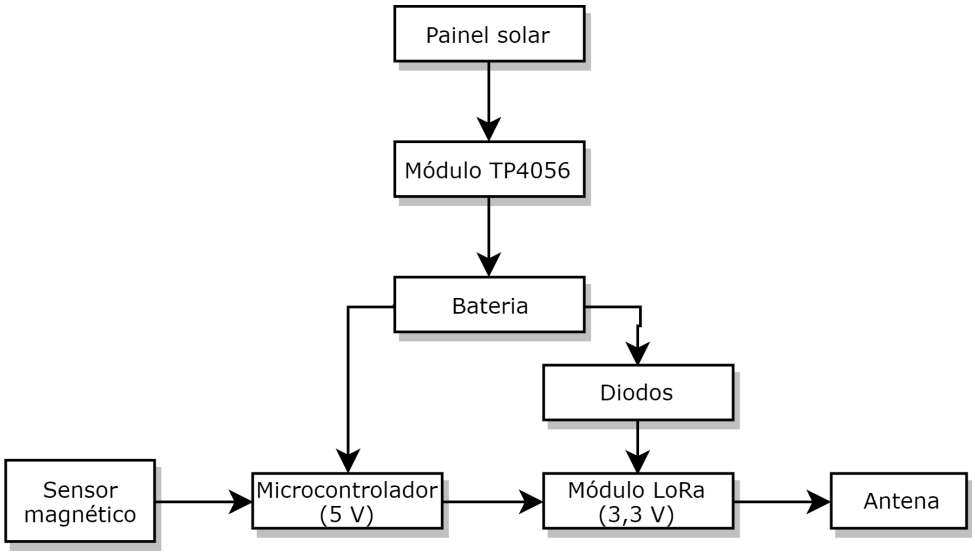


Figura 3.1 – Diagrama de blocos do *hadrware* utilizado

Fonte: Autor.

Para o desenvolvimento do sistema, foi utilizado o microcontrolador ATmega328p, através de sua placa de desenvolvimento Arduino Pro Mini, a identificação de abertura de chave fusível se dará através de um sensor magnético (*Reed switch*), que enviará um dado através de uma rede LoRa, pelo módulo RFM95. O dispositivo conta com carregamento de bateria através de energia solar, cujo funcionamento também será descrito nas próximas seções.

3.1.1 Arduino 328p

A placa Arduino Pro Mini é uma placa de microcontrolador baseado no ATmega328p. Possui 14 pinos digitais de entrada/saída (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, 1 UART (porta serial de *hardware*), um cristal oscilador de 16 MHz, a conexão USB é dada através de um conversor FTDI para realizar a programação do mesmo. O mais importante deste processador é a possibilidade de modificações para a implementação de técnicas de hibernação, reduzindo o consumo de energia de forma significativa. A placa retorna em seu estado normal através de interrupção externa, em um dos dois pinos que possuem essa função.

A placa Arduino será responsável pelo processamento dos eventos, pois a biblioteca LMIC, responsável pelo envio de dados através de LoRaWAN, é baseada em agendamento de eventos, portanto, o Arduino é responsável pelos mesmos. Uma representação da placa foi apresentada na Figura 2.4.

3.1.2 Reed Switch

Um sensor magnético (*Reed switch*) é uma chave controlada por um campo magnético aplicado na proximidade. É composto de 2 placas metálicas encapsuladas em um recipiente hermeticamente selado. Uma das placas é fixa, invariante a mudanças no campo magnético presentes, já a outra é composta de material magnético. Portanto, quando um ímã é posicionado próximo de um *Reed swicth*, o contato que é normalmente aberto, se fecha.

A aplicação no monitoramento de abertura de chaves fusíveis se dará da seguinte maneira: em uma caixa, junto com a placa Arduino, estará um tubo com um ímã, no final deste tubo, um Reed switch conectado ao Arduino. Quando a chave fusível for aberta, pela gravidade o ímã percorrerá o tubo (este, fixo) até a extremidade que contém o Reed switch, que provocará uma interrupção no Arduino, indicando para enviar um dado através do protocolo LoRaWAN. Os envios se repetirão a cada minuto até que a chave seja fechada novamente pela equipe de manutenção. Com isso, o ímã retornará para sua posição inicial, abrindo o circuito do *Reed switch*. O Arduino entrará em modo de hibernação, e o processo se repetirá quando a chave fusível for aberta novamente.

Na Figura 3.2 é apresentado um *Reed switch*. Um esquema da forma que o sistema identificará a abertura da chave é apresentado na Figura 3.3.

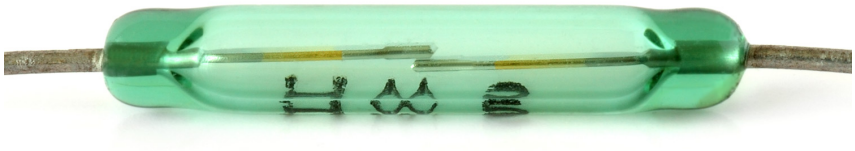


Figura 3.2 – Reed switch

Fonte: Karwath (2005).

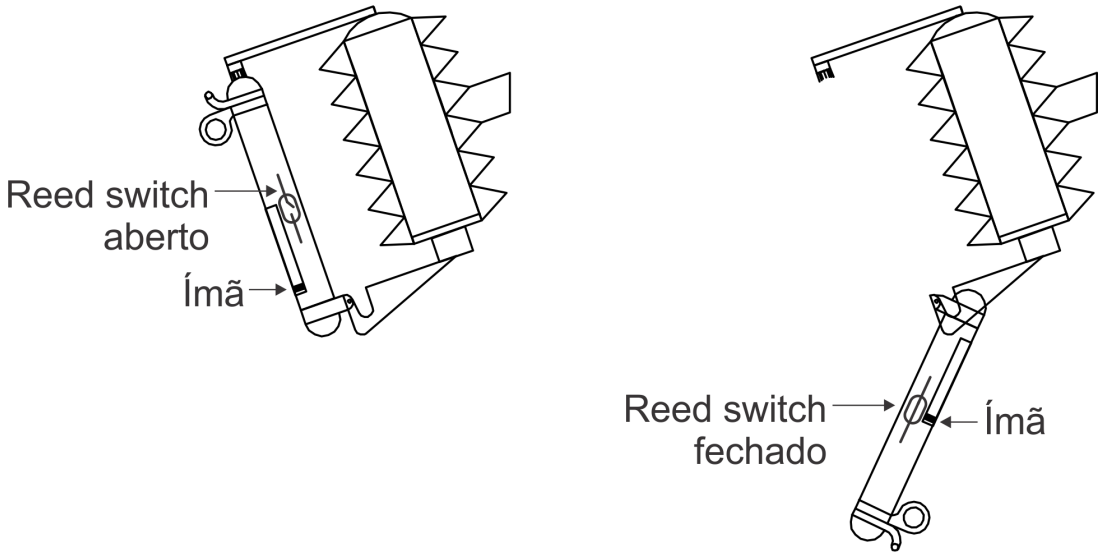


Figura 3.3 – Funcionamento do monitoramento pelo *Reed switch*.

Fonte: Autor.

3.1.3 Módulo LoRa RFM95

O envio dos dados por LoRa se dá através do Módulo RFM95, que possui as seguintes características:

- Faixa de Frequências: 137-1020 MHz;
- Fator de Espalhamento: 6-12;
- Largura de Banda: 7.8-500kHz;
- Taxa de Bits Efetiva: 0.18-37.5 kbps;
- Sensibilidade Estimada: -111 a -148 dBm.

O módulo está apresentado na Figura 3.4.

O módulo utilizado é em sua versão 915 MHz, para o funcionamento com redes brasileiras. Como destaque, temos o consumo de potência de apenas 0,2 μA em modo de hibernação. Quando enviando dados, o módulo consome 120 mA, porém por um curto

Figura 3.4 – Módulo LoRa RFM96

Fonte: Hoperf (2016).

período de tempo, possibilitando a operação com bateria, já que os envios são limitados a um longo período de tempo. Esses envios programados são necessários para certificação de que o sistema está em pleno funcionamento, com o envio da tensão da bateria assim como o estado da chave fusível.

A comunicação do módulo com o Arduino se dá por uma interface serial de periféricos (SPI, do inglês *Serial Peripheral Interface*). Utilizando uma conexão SPI, sempre há um dispositivo mestre (neste caso o Arduino), que controle o dispositivo periférico (neste caso o módulo RFM95). Tipicamente existem 4 conexões comuns a todos os dispositivos (ARDUINO, 2019):

- MISO: (Entrada do Mestre, Saída do Escravo, do inglês *Master In Slave Out*) Linha em que o Escravo manda informações para o Mestre;
- MOSI: (Saída do Mestre, Entrada do Escravo, do inglês *Master Out Slave In*) Linha em que o Mestre manda informações para os Escravos;
- SCK: (*Clock* do sistema, do inglês *System Clock*) Pulsos de *clock* que sincronizam a transmissão de dados gerada pelo Mestre;
- SS: (Seleção de Escravo, do inglês *Slave Select*) Pino no qual o Mestre habilita a comunicação. Com isso, é possível utilizar diversos dispositivos compartilhando os mesmos pinos de MISO, MOSI e SCK, cada um atuando por vez.

3.1.4 Caixa

Todos os componentes do dispositivo necessitam estarem abrigados em uma caixa, resistente à entrada de água, pois a caixa ficará exposta ao tempo. Para isso, a caixa

escolhida para o protótipo possui as seguintes dimensões: 3,5 cm de altura, 5 cm de largura e 12,5 cm de comprimento, podendo ser vista na Figura 3.5.

Figura 3.5 – Caixa do dispositivo

Fonte: Autor.

3.1.5 Biblioteca Low-Power

Para o desenvolvimento deste trabalho, foi utilizada uma biblioteca customizada e otimizada para o microcontrolador ATmega328p, chamada de *Low-Power*, desenvolvida pelo usuário do *Github* rocketscream (ROCKETSCREAM, 2019).

Quando o ATmega328p está em modo ativo, sempre estará executando continuamente milhares de instruções por segundo. Seus periféricos também estão ativos, como por exemplo o conversor analógico-digital, SPI, *timers*, I2C, USART, *Watchdog Timer* e a detecção de subtensão (BOD), consumindo potência.

Para reduzir o consumo de potência, o ATMega328p suporta alguns modos de sono, no qual periféricos não utilizados podem ser desativados. Esses modos diferem em quais partes permanecem ativas, pela duração e pelo tempo necessário para retornar ao modo normal. Com isso, foram desenvolvidas diversas customizações desses modos de sono pelo usuário *rocketscream*, através da biblioteca *Low-Power*.

A biblioteca é simples de usar, porém com muito poder, utilizando o comando:

```
LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF)
```

põe o microcontrolador em sono entre 16 ms e 8s, dependendo do primeiro argumento. Esse modo desativa o conversor AD e o BOD. Esse modo de sono significa que todas as funções do chip estão desativadas até a próxima interrupção. Somente interrupções em INT1 e INT2, interrupções de mudança de pinos ou o *Watchdog timer* conseguem trazer o microcontrolador de volta ao estado normal.

A seguir, na Tabela 3.1, é realizada uma comparação entre os modos de sono com e sem o uso da biblioteca, para um microcontrolador ATmega 328p.

Tabela 3.1 – Consumo em diferentes modos de sono

ATmega328P Pro Mini	**Fonte de tensão**	**Estado**	**5.0 V @ 16 MHz**	**3.3 V @ 8 MHz**
Sem modificações	RAW	ACT	19.9 mA	4.74 mA
Sem modificações	RAW	PDS	3.14 mA	0.90 mA
Sem LED	RAW	ACT	16.9 mA	3.90 mA
Sem LED	RAW	PDS	0,0232 mA	0.0541 mA
Sem LED, sem regulador	VCC	ACT	12,7 mA	3,58 mA
Sem LED, sem regulador	VCC	PDS	0,0058 mA	0,0045 mA

Fonte: Adaptado de Neuy (2017).

3.1.6 Bateria

É necessário definir uma bateria capaz de garantir a operação plena do dispositivo por uma longa vida útil. No total, o sistema possui dois agentes responsáveis por um consumo significativo de potência: o microcontrolador e o módulo LoRa. Durante o desenvolvimento prático, foram utilizadas bibliotecas customizadas para o microcontrolador, que desativam diversas funções não utilizadas, o microcontrolador entra em um estado de sono profundo. Com a utilização dessa biblioteca, é possível manter o microcontrolador com um consumo de 7 μA (ROCKETSCREAM, 2019).

Analisando o *datasheet* do fabricante do módulo LoRa, o mesmo também apresenta estado de sono profundo automático. Portanto, a operação do módulo se divide em duas situações: Envio e Não envio. Durante o envio, que dura milissegundos (dependendo do fator de espalhamento utilizado), o módulo consome entre 20 e 120 mA. No estado de sono automático, o módulo consome entre 0,2 e 1 μA (HOPERF, 2016).

O envio dos dados se dará de duas formas: periódico (1 vez ao dia), para garantir que o sistema está em pleno funcionamento, e também quando a chave fusível é aberta,

repetindo o envio a cada minuto até que o problema seja resolvido. Portanto, em situações normais, em quase a totalidade do dia, o consumo do sistema no geral é reduzido. Para isso, foi utilizada uma bateria com a capacidade de 530 mAh, de tensão nominal de 3,7 V e código LB530LP1. A bateria está apresentada na Figura 3.6.

Figura 3.6 – Bateria utilizada para o dispositivo.

Fonte: Autor.

Para o cálculo da autonomia do sensor e para verificar a necessidade de carregamento da bateria através de painel solar, são analisadas duas situações: operação normal e chave aberta.

Em operação normal, temos um consumo de 7 μA do microcontrolador e 1 μA do módulo LoRa, portanto com um envio (utilizando um total de 5 segundos entre o microcontrolador despertar e enviar, para dormir novamente), temos:

$$(7\mu A + 1\mu A) * 1000mA/A * (23h + \frac{59min}{60min/h} + \frac{55s}{3600s/h}) = 0,192mAh/dia \quad (3.1)$$

Já durante o envio, temos o consumo do microcontrolador e do módulo de 19,9 mA e 120 mA, respectivamente, portanto:

$$(19,9mA + 120mA) * \frac{5s}{3600s/h} = 0,0276mAh/dia \quad (3.2)$$

Com isso, temos um consumo diário de 0,2196 mAh. Neste cenário, levando em conta um fator de 15 $\%$ de auto descarga, temos uma vida útil de 2051 dias, ou 5,69 anos. Porém este cenário apenas leva em conta um funcionamento onde a chave fusível nunca é aberta.

Para a próxima estimativa, é suposto que a chave fusível é aberta uma vez a cada semana, e em cada vez que é aberta, leva-se 1 hora para solucionar o problema. Utilizando os mesmos consumos, temos para o sono:

$$(7\mu A + 1\mu A) * 1000mA/A * [(24h * 6d) + 23h + \frac{54min}{60min/h} + \frac{25s}{3600s/h}] = 1,34mAh/semana \quad (3.3)$$

E para os envios:

$$(19,9mA + 120mA) * (\frac{5s * 60}{3600s/h} + 7 * 5s) = 13mAh/semana \quad (3.4)$$

Este caso, que em muitas vezes retrata mais a realidade, temos um consumo de 31,4 mAh por semana, com a duração da bateria de 220 dias, ou seja, menos de um ano. Para isso, a alternativa de carregamento solar se mostra uma boa ideia, que será tratada a seguir.

3.1.7 Carregamento Solar

Analisando os componentes utilizados, é possível ter uma estimativa do consumo total do sistema, com a bateria escolhida, deve-se determinar o carregamento da mesma através de um painel solar, visto que o dispositivo ficará exposto ao tempo, colocado junto a chaves fusíveis, geralmente com grande exposição solar. Vale ressaltar que o sistema é desenvolvido para apresentar um baixíssimo consumo de energia, então, o carregamento solar é opcional, apenas para aumentar a vida útil do sistema.

Para isso, nessa seção será abordado a parte do carregamento das baterias através de energia solar, serão abordados os módulos e bateria utilizados.

3.1.7.1 Painel Solar

Para o projeto do painel solar, é necessário avaliar o tamanho da caixa na qual o dispositivo será instalado e também o consumo de todos os integrantes do dispositivo.

A caixa, apresentada na Seção 3.1.4, possui as seguintes dimensões: 3,5 cm de altura, 5 cm de largura e 12,5 cm de comprimento. Por isso, foi utilizado um painel solar de 5,3 cm x 3 cm, com o propósito de ser instalado no topo da caixa. O painel solar está apresentado na Figura 3.7.

Este módulo possui uma tensão de operação de 5 V e corrente nominal de 30 mA. Como descrito na Seção 3.1.6, temos que o consumo total do módulo é bem baixo, portanto um painel de 30 mA consegue manter a operação do sistema. Após, é necessário escolher o módulo responsável pelo carregamento da bateria, para regular a tensão de carregamento quando há disponibilidade de energia solar.

Figura 3.7 – Painel solar utilizado

Fonte: Autor.

3.1.7.2 Módulo TP4056

Para o carregamento da bateria, foi utilizado o módulo TP4056, responsável por regular a tensão fornecida para a bateria em 4,2 V. É um carregador linear de corrente constante/tensão constante para baterias de íon de lítio. O módulo com o TP4056 possui apenas um elemento variável, para definir a corrente limite de carga. A Figura 3.8 apresenta a ligação do módulo.

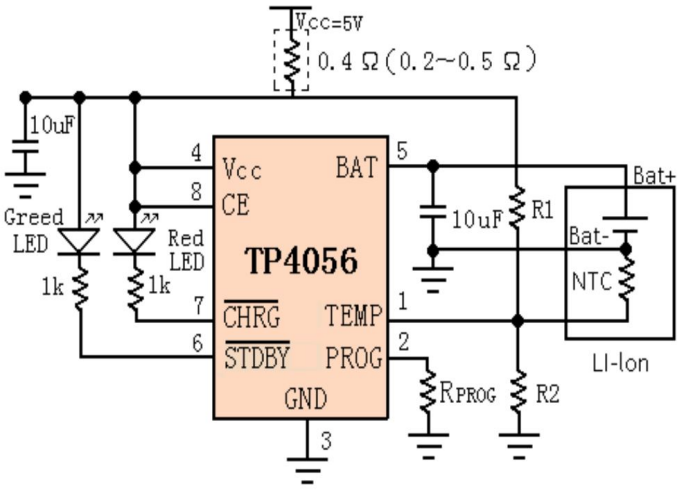


Figura 3.8 – Ligação do módulo TP4056

Fonte: NanJing (2017).

A configuração do resistor R_{PROG} se dá de acordo com a Tabela 3.2.

Portanto, como o painel solar adotado, quando em funcionamento pleno, tem uma tensão de 5V e uma corrente de 30mA, é escolhido o maior valor para R_{PROG}, neste caso, 10kΩ, de modo que a corrente limitada pelo módulo TP4056 será de 130mA.

Tabela 3.2 – Relação entre o resistor R_{PROG} e a corrente fornecida para a bateria.

R_{PROG}(k)	I_{BAT}(mA)
10	130
5	250
4	300
3	400
2	580
1,66	690
1,5	780
1,33	900
1,2	1000

Fonte: Adaptado de NanJing (2017).

3.2 FUNCIONAMENTO DO SISTEMA

Com todos os módulos pertencentes ao desenvolvimento prático do sistema já abordados, parte-se para uma descrição do funcionamento do sistema como um todo, primeiramente, será abordada a implementação prática para integrar cada serviço, *hardware* e *software* com outros, afim de desenvolver um protótipo operante.

3.2.1 Diagrama de Blocos

Em resumo, temos que o dispositivo deverá enviar informações em duas situações: quando a chave fusível abrir, repetindo o envio em um pequeno intervalo de tempo, como por exemplo, 1 minuto, para o sistema possuir uma redundância, ou também em um intervalo maior de tempo, como por exemplo, a cada dia, para se ter certeza de que a bateria está em funcionamento pleno.

Portanto, o funcionamento se dará da seguinte forma: em qualquer momento que a chave fusível abrir, é verificada uma interrupção pelo *reed switch*, com isso, as interrupções são desabilitadas, para a interpretação inicial, agenda-se um envio para o mesmo instante. O código responsável pelo agendamento de envios então verifica se de fato, a chave fusível está aberta, para evitar falsos positivos. Se sim, um dado indicando a abertura é enviado, e o próximo agendamento é feito para depois de 1 minuto. Se a chave não estiver aberta, é enviado um dado indicando que a mesma encontra-se fechada e o próximo agendamento é realizado para o próximo dia, nesse caso, as interrupções são habilitadas novamente e nesse ponto, o código de ambas as situações se encontra na mesma linha, que indica o dispositivo para entrar em sono profundo. O funcionamento detalhado, através de um diagrama de blocos, é mostrado na Figura 3.9.

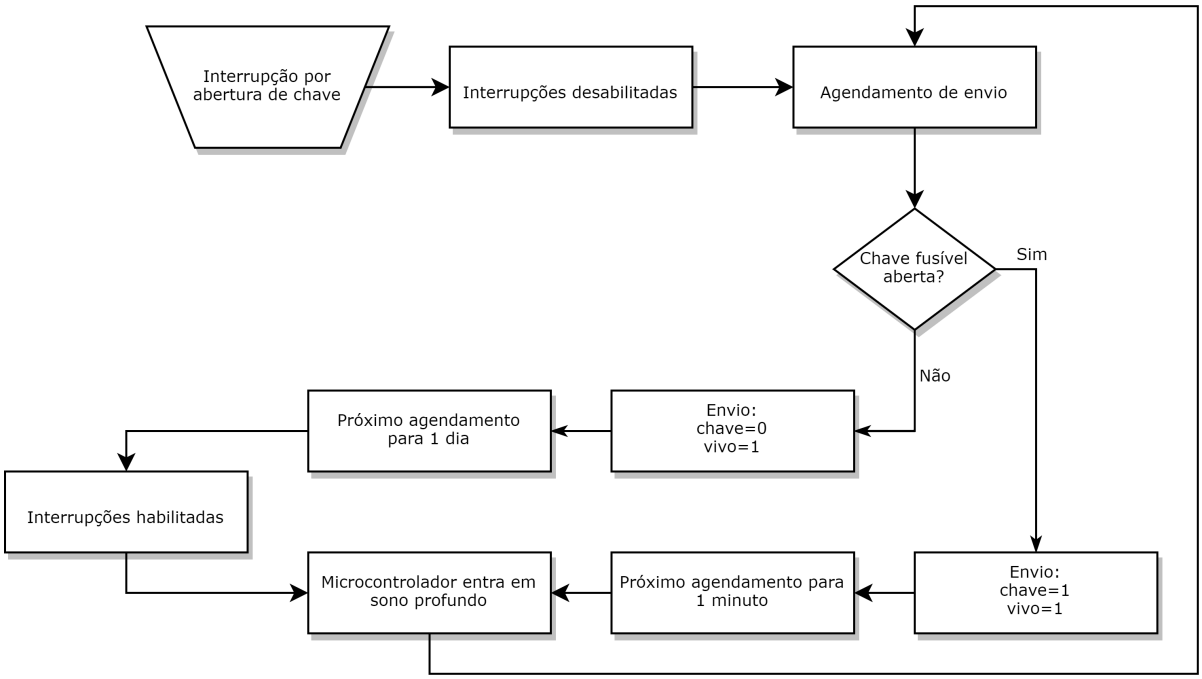


Figura 3.9 – Diagrama de blocos projetado para o sistema.

Fonte: Autor.

3.2.2 Console The Things Network

Para o desenvolvimento prático deste trabalho, é necessário fazer a escolha do servidor de rede a ser utilizado. Diversas soluções são verificadas atualmente. Em um primeiro momento, uma opção viável seria a utilização dos serviços da ChipStack (CHIRPSTACK, 2019), principalmente por ser uma solução de código aberto. Foram realizados testes com a plataforma, que se mostrou estável, porém, era necessário manter um *gateway*, neste caso uma placa de desenvolvimento *Raspberry Pi* para receber os dados enviados.

Como os testes foram realizados na Universidade Federal de Santa Maria, a mesma já possui uma rede LoRaWAN implementada através do projeto de pesquisa intitulado "Pesquisa e desenvolvimento de uma rede LoRaWAN na UFSM"(BARRIQUELLO, 2017). Esta rede está integrada com um servidor de rede hospedado na plataforma do The Things Network (TTN), portanto a mesma foi utilizada para o desenvolvimento deste trabalho.

Esta solução possui código fonte disponível no Github e Possibilita implementações privadas ou também através de uma plataforma online, chamada de *console*. O TTN, realiza a separação de duas frentes: *Aplicações* e *Gateways*, conforme pode ser visto na tela inicial do *console*, apresentada na Figura 3.10.

Nas próximas seções, serão abordados os passos necessários para o registro de um *Gateway*, a criação de uma aplicação e o registro de um dispositivo na mesma. Após, os dados deste dispositivos serão registrados no código do protótipo deste trabalho.

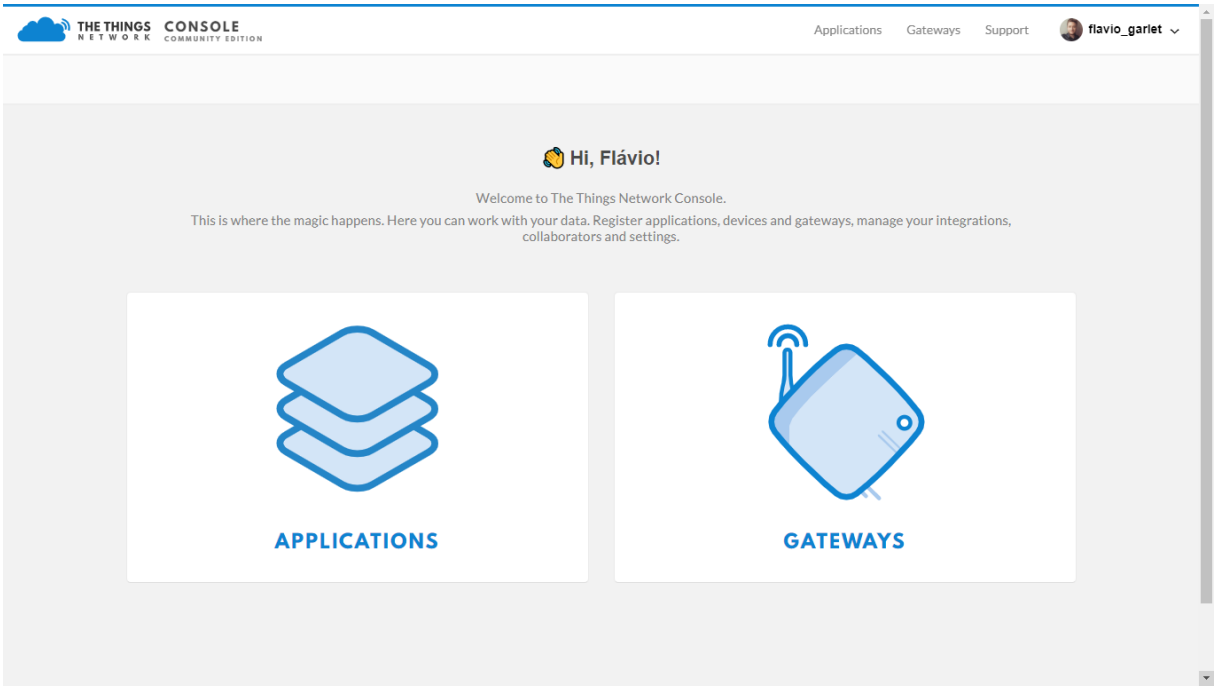


Figura 3.10 – Console da plataforma do The Things Network.

Fonte: Autor.

3.2.2.1 Gateway

O registro de um *Gateway* no console do TTN é simples. O processo é o mesmo para diversos dispositivos, que utilizam o concentrador SX1301, configurado para frequências australianas. O arquivo de configuração para os padrões brasileiros está disponível no Apêndice A.

No console do TTN, o registro de um *gateway* é através da página "*Gateways*", e em seguida, "*Register gateway*", ou através do link <https://console.thethingsnetwork.org/gateways/register>. A tela desta página está apresentada na Figura 3.11.

A caixa de seleção *I'm using the legacy packet forwarder* deve ser selecionada, com isso, a opção *Gateway EUI* estará disponível para edição. Neste campo, deverá ser inserido o número identificador do *Gateway*, disponível na configuração do mesmo, através do arquivo "global_conf.json".

Em *Description*, deverá ser inserida uma descrição do *Gateway*, como por exemplo onde está localizado, fabricante, etc. Em *Frequency Plan*, para uso no Brasil, deve ser escolhido a banda australiana, em 915 MHz. Já em *Router*, é recomendada a seleção do router brasileiro, por apresentar uma menor distância, e portanto, menos redirecionamentos. Em *Location*, a localização que o *Gateway* será instalado, e por fim, em *Antenna Placement*, se o *Gateway* será interno ou externo.

Com estas configurações, basta confirmar em *Register Gateway*, o que redirecionará para a tela com os *Gateways* registrados na conta do usuário. Caso o mesmo deseje

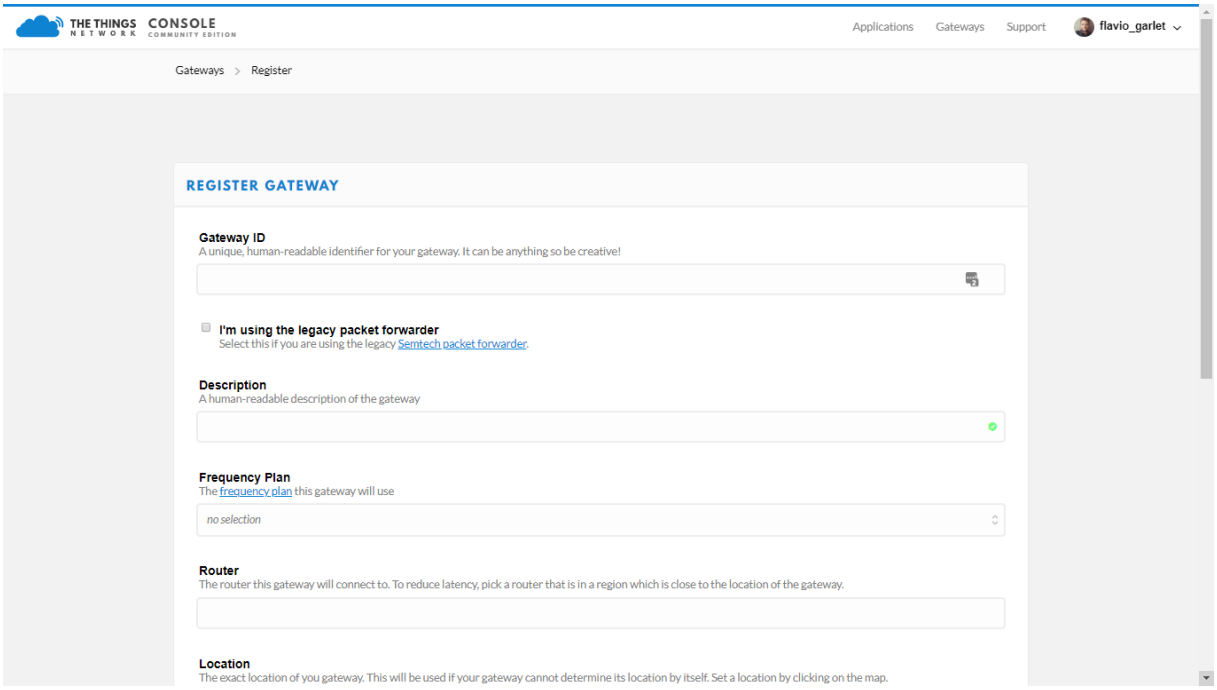
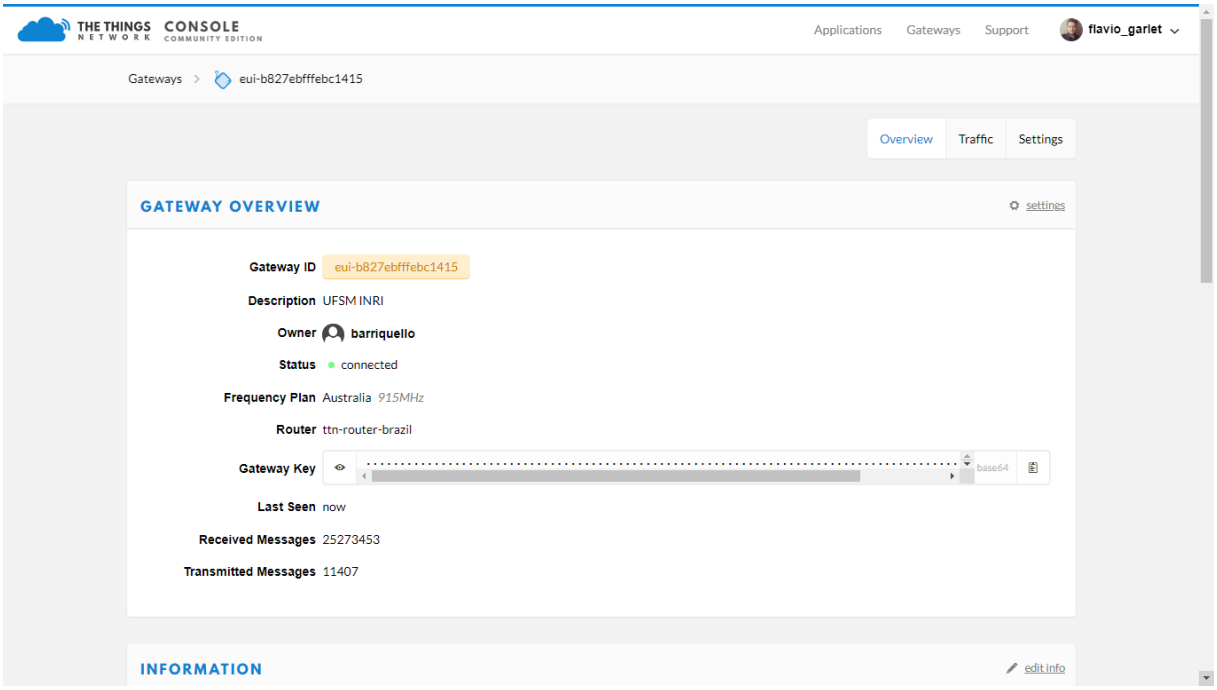


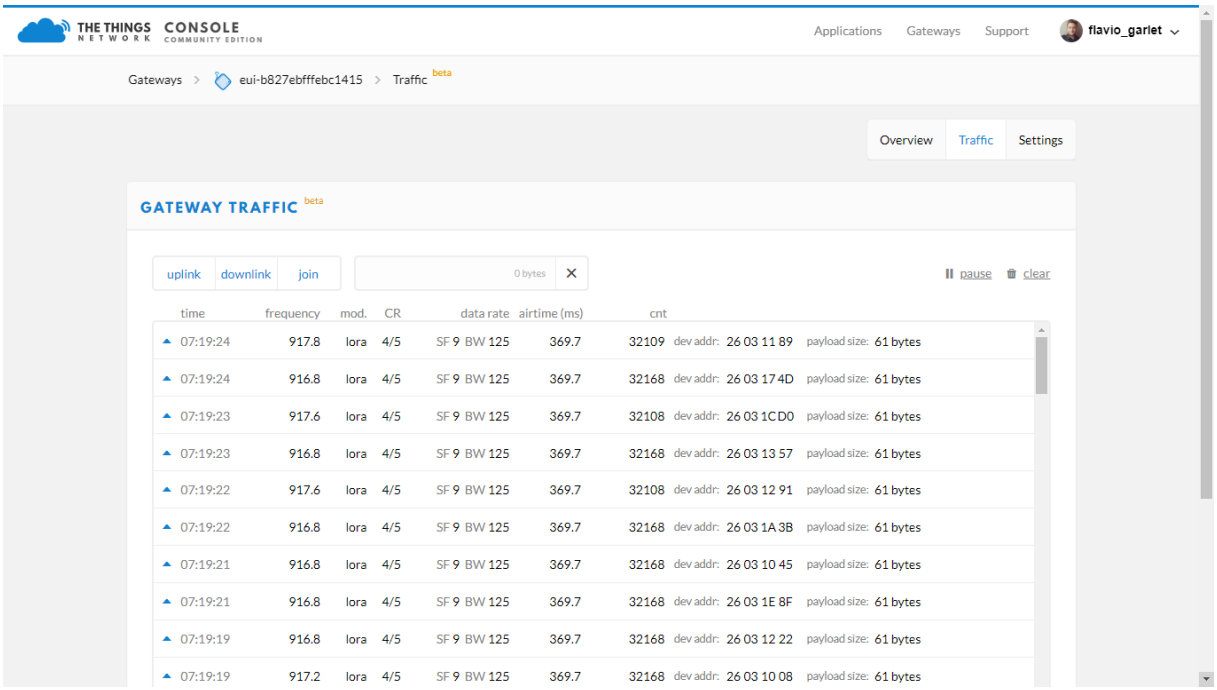
Figura 3.11 – Tela de Registro de *Gateway* no TTN.

Fonte: Autor.

verificar os dados que o *Gateway* está recebendo, basta clicar sobre o *Gateway* desejado, que redirecionará para uma tela de configuração geral, apresentada na Figura 3.12a. Em *Traffic* é posível verificar os pacotes que o *Gateway* está recebendo, como mostrado na Figura 3.12b.



(a)



(b)

Figura 3.12 – (a)*Gateway* recém criado no TTN e (b)Pacotes recebidos pelo *Gateway* no TTN.

3.2.2.2 Aplicação

Na tela de Aplicações do console do TTN, é possível adicionar, editar ou remover aplicações. Para cada uma, é possível registrar dispositivos. Primeiramente, deve-se criar uma aplicação para interpretar os dados e em seguida, adicionar os dispositivos identificadores de abertura de chave fusível.

Também nos mesmos moldes da criação de um *Gateway*, é necessário clicar em *add application*, ou através do link <https://console.thethingsnetwork.org/applications/add>. A página resultante é apresentada na Figura 3.13.

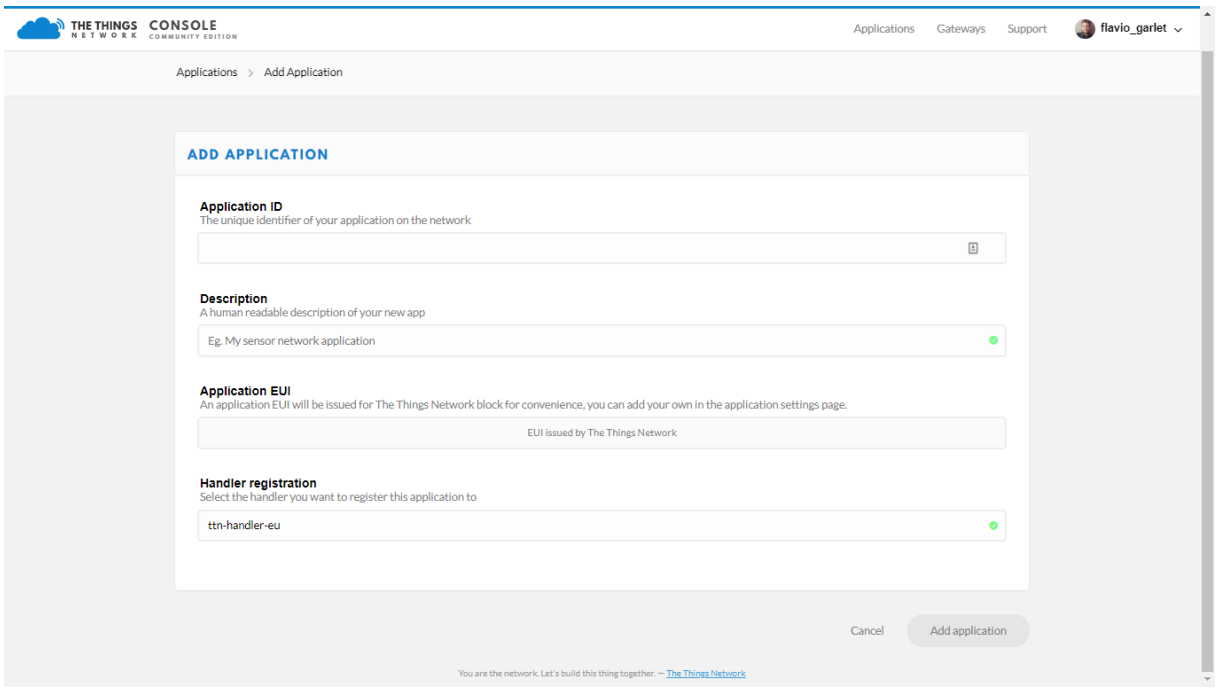


Figura 3.13 – Tela de Registro de aplicação no TTN.

Fonte: Autor.

Em *Application ID*, é necessário fornecer um identificador para a aplicação, que será utilizado para a integração com a interface gráfica. Em *Description*, uma descrição do propósito da aplicação, e em *Handler registration*, selecionar o *handler* brasileiro.

A aplicação é responsável por receber os dados e separá-los, isto é implementado através da guia *Payload Formats*. O código utilizado para a decodificação dos dados é apresentado a seguir:

```
function Decoder(bytes, port) {
  var alive  = bytes[0]; // OK
  var trip   = bytes[1]; // Cutout fuse is opened
  return{
    alive:alive,
    trip:trip
  };
}
```

Por fim, é necessário adicionar os dispositivos à aplicação, o que é possível através da aba *Devices*, e em *register device*, no canto superior direito. A tela de registro de dispositivo está apresentada na Figura 3.14.

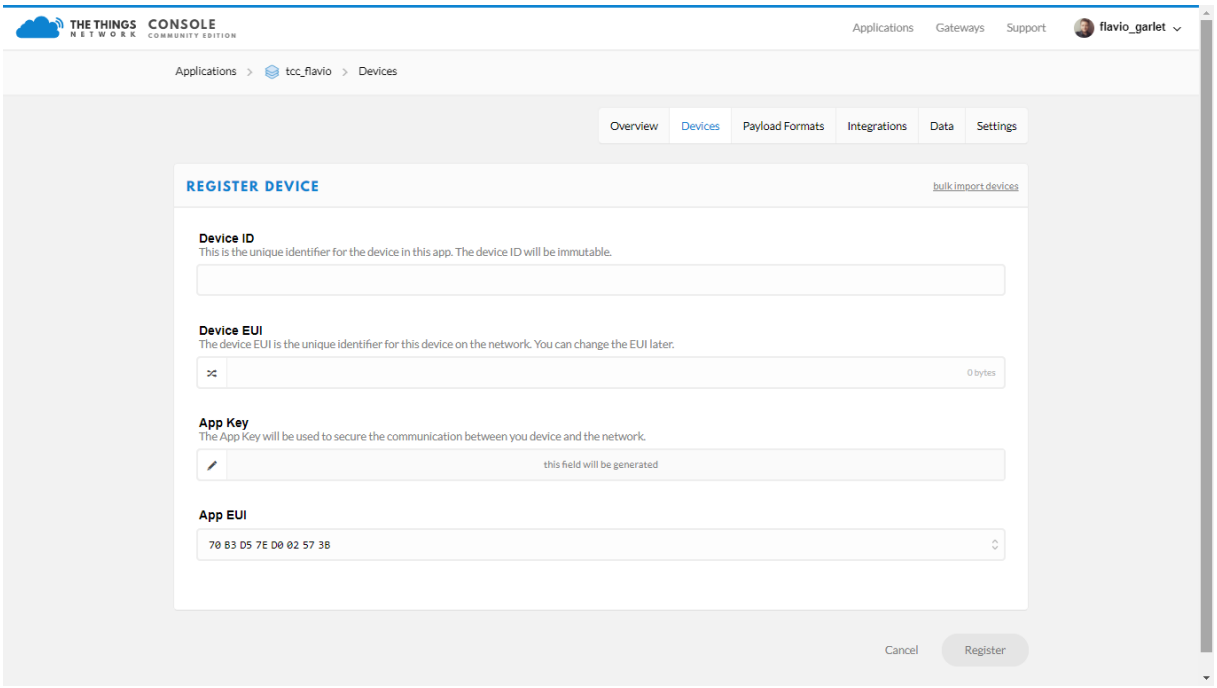


Figura 3.14 – Tela de Registro de dispositivo no TTN.

Fonte: Autor.

Em *Device ID*, é necessário inserir um identificador único para o dispositivo em questão. Em *Device EUI*, deve-se inserir o endereço do dispositivo, onde pode ser definido um valor arbitrário. Após, é necessário finalizar o registro, pela aba *Settings*, cuja tela está apresentada na Figura 3.15.

Em *Activation Method*, é necessária a alteração de OTAA para ABP e a desabilitação da opção *Frame Counter Checks*. Retornando à página principal, em *Overview*, os valores dos campos *Device Address*, *Network Session Key* e *App Session Key* deverão ser inseridos no código do dispositivo.

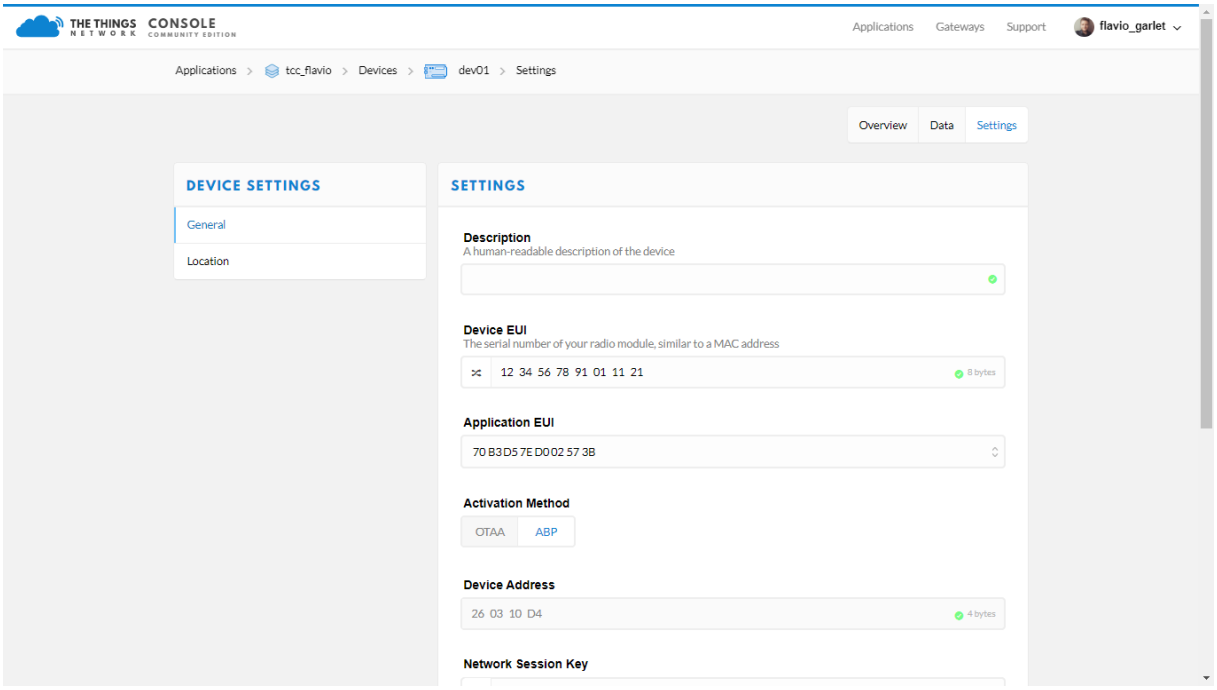


Figura 3.15 – Tela de configuração de dispositivo no TTN.

Fonte: Autor.

3.2.3 Descrição do Código Desenvolvido

Nesta seção, serão abordados as bibliotecas e os códigos necessários para a correta operação do sistema, em Arduino.

3.2.3.1 Biblioteca LMIC

A biblioteca LMIC para o Arduino é uma modificação da biblioteca original da IBM, para operação em um abiente Arduino, permitindo o uso com o módulo HopeRF RFM95.

Sua instalação é simples, através do repositório oficial do Github, no seguinte link: <https://github.com/matthijskooijman/arduino-lmic>

A instalação pode ser feita através de:

- Gerenciador de bibliotecas do Arduino ("Sketch-> "Incluir Biblioteca-> "Gerenciar Bibliotecas..."), ou
- Download da biblioteca através do Github ("Download ZIP") e após, na IDE do Arduino, ("Sketch-> "Incluir Biblioteca-> "Adicionar Biblioteca ZIP..."), ou
- Clonar o repositório do Github para a pasta de bibliotecas do Arduino.

Após, é necessário realizar a conexão entre o módulo RFM95 e o Arduino Pro Mini, essa conexão é feita através da interface SPI. Nessa interface, temos 4 conexões: MOSI,

MISO, SCK e SS, as três primeiras possuem pinos equivalentes no Arduino, já a última, pelo Arduino ser o mestre, pode ser conectado em qualquer pino digital, sendo necessário apenas configurar na biblioteca qual foi o escolhido. Também é necessária a conexão dos pinos digitais DIO0 e DIO1, que são utilizados para informação de *status* instantâneo. DIO0 é usado para verificar quando uma transmissão ou recepção foi concluída e DIO1 é utilizado para verificar quando se esgotou o tempo limite para uma recepção de dado.

A conexão dos pinos entre o módulo RFM95 e o Arduino Pro Mini é apresentada na Tabela 3.3. A alimentação do módulo RFM95, que é de até 3,9V, é fornecida através da mesma almentação do Arduino, que é de 5V, com 2 diodos em série, resultando em 3,6V.

Tabela 3.3 – Conexão do módulo RFM95 e o Arduino Pro Mini

HopeRF RFM95	**Arduino Pro Mini**	**HopeRF RFM95**	**Arduino Pro Mini**
ANT	-	GND	-
GND	GND	DIO5	-
DIO3	-	RESET	5
DIO4	-	NSS	10
3.3V	VCC*	SCK	13
DIO0	4	MOSI	11
DIO1	3	MISO	12
DIO2	-		

Fonte: Autor.

A inicialização do módulo no Arduino se dá indicando os pinos, pelas seguintes linhas de código:

```
1 #define LORAWAN_RFM95_PIN_NSS    9
2 #define LORAWAN_RFM95_PIN_RST    5
3 #define LORAWAN_RFM95_PIN_D0     4
4 #define LORAWAN_RFM95_PIN_D1     3
5 const lmic_pinmap lmic_pins = {
6   .nss = LORAWAN_RFM95_PIN_NSS,
7   .rxtx = LMIC_UNUSED_PIN,
8   .rst = LORAWAN_RFM95_PIN_RST,
9   .dio = {LORAWAN_RFM95_PIN_D0, LORAWAN_RFM95_PIN_D1,
      LMIC_UNUSED_PIN},
10 };
```

Também é necessário configurar o endereço do dispositivo e as chaves, da aplicação e da rede, vindas da configuração do dispositivo no *The Things Network*. No caso do dispositivo desenvolvido e utilizado para testes, o endereço e chaves são os seguintes:

- Endereço do dispositivo: 26 03 10 D4;

- Chave da rede: D4 BA 79 33 DB 78 AB 6D 7C 1F EA F0 B2 BE 77 0F;
- Chave da aplicação: 6B F2 95 BF E7 46 1C 68 A7 AB 84 32 A4 95 F8 15.

Estes valores são registrados no Arduino com o seguinte código:

```
1 uint8_t LORAWAN_DEVADDR[4]={0x26, 0x03, 0x10, 0xD4};
2 uint8_t LORAWAN_NWSKEY[16]={0xD4, 0xBA, 0x79, 0x33, 0xDB, 0x78,
     0xAB, 0x6D, 0x7C, 0x1F, 0xEA, 0xF0, 0xB2, 0xBE, 0x77, 0x0F
    };
3 uint8_t LORAWAN_APPKEY[16]={0x6B, 0xF2, 0x95, 0xBF, 0xE7, 0x46,
     0x1C, 0x68, 0xA7, 0xAB, 0x84, 0x32, 0xA4, 0x95, 0xF8, 0x15
    };
```

Através da função init_LoRaWAN, as chaves e outros parâmetros de transmissão são definidos:

```
1 init_LoRaWAN(LORAWAN_SESSION_PORT, LORAWAN_DEVADDR,
    LORAWAN_NWSKEY,
2 LORAWAN_APPKEY,     // Set port and keys
3 LORAWAN_START_CH, LORAWAN_END_CH, LORAWAN_CH_500,// Select
    channel enable
4 LORAWAN_ATTEMPTS,   // Attempts for confirmed frames
5 LORAWAN_DR,         // Data rate TX for uplink
6 LORAWAN_DN2DR);     // Data rate RX2 for downlink
```

Basicamente, com os parâmetros registrados pela biblioteca, a mesma é responsável por fazer todo o gerenciamento de instruções para o envio automaticamente. Um evento de envio pode ser agendado através do seguinte comando:

```
do_send(&sendjob);
```

Foram realizadas modificações apenas em 2 itens. Primeiro, quando um evento é agendado, é realizada uma verificação do estado da chave fusível, através das seguintes linhas de código:

```
1 void do_send(osjob_t* j)
2 {
3   detachInterrupt(digitalPinToInterrupt(REED));
4   wdt_enable(WDTO_8S);
5   lpp.reset();                          // clear the buffer
```

```
6   uint8_t alive = 1;
7   uint8_t trip;                        // trip value
8   if (!digitalRead(REED)) trip  = 1; // cutout fuse check
9   else trip  = 0;
10  lpp.addUint8(alive);                 // add data to payload
11  lpp.addUint8(trip);
12  // Auto select DR
13  if (COUNT_CH_500khz)
14  {
15    if (COUNT_CH_125khz) LMIC.datarate = (++COUNT_DATA %
16    (COUNT_CH_125khz + COUNT_CH_500khz)) ? LORAWAN_DR :
        LORAWAN_DR_500;
17    else LMIC.datarate = LORAWAN_DR_500;
18  }
19  // Prepare upstream data transmission at the next possible
      time.
20  if (!sendTX_LoRaWAN(LORAWAN_SESSION_PORT, lpp.getBuffer(),
21  lpp.getSize(), LORAWAN_ACK)) AVRreset(); // Reset by watchdog
22 }
```

A outra modificação foi o gerenciamento do próximo envio, que é realizado quando um evento de envio é completado. A maneira desenvolvida foi através laços *for*, que repetem a instrução de colocar o microcontrolador em sono profundo em seu intervalo máximo, que é de 8 segundos. O laço for repete em intervalos de 8 s até o tempo para o próximo envio, que é de 1 dia ou 1 minuto, se a chave fusível estiver fechada e aberta, respectivamente. Esse funcionamento é desenvolvido através das seguintes linhas de código.

```
1 void onEvent (ev_t ev)
2 {
3   switch (ev)
4   {
5     case EV_TXCOMPLETE:
6       wdt_reset();
7       if (!digitalRead(REED)) { // Check if cutout fuse is
          still opened
8         sleep((uint32_t)(LORAWAN_INTERVAL * 1000)); // Sleep
            now
9         do_send(&sendjob); // Start transmission LoRaWAN
10      }
```

```
11        else {
12          attachInterrupt(digitalPinToInterrupt(REED), wakeUp,
               FALLING);
13          for (int i = 0; i < 10801; i++) {// 1 day / 8s = 10800s
14            if (!flag) { // Interrupt flag
15              LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
16            }
17            else {
18              break; // Break and send data if flag = true
19            }
20          }
21          do_send(&sendjob); // Start transmission LoRaWAN
22        }
23        flag = false;
24        break;
25    }
26 }
```

3.2.3.2 Interrupção pelo Reed Switch

O ponto principal deste trabalho é a identificação de abertura de chaves fusíveis, o que é realizado através de um *reed switch*. Para isso, primeiramente deve-se conectar o sensor entre o GND e um pino que suporte interrupções, neste caso, foi utilizado o pino 2.

Deve-se inicializar o pino utilizando o *pull-up* interno do Arduino, através de:

```
#define REED                2
pinMode(REED, INPUT_PULLUP);
```

Por fim da configuração, deve-se configurar a interrupção, através de:

```
attachInterrupt(digitalPinToInterrupt(REED), wakeUp, FALLING);
```

Com isso, quando o pino 2 for para nível lógico 0, a função *wakeUp* será executada, esta função deve ser simples, e consiste de apenas dois argumentos: Desabilitar as interrupções e definir o valor de uma variável de 0 para 1. Como visto na Seção 3.2.3.1, quando esta mesma variável é verdadeira, o laço *for* é interrompido, pulando para a próxima instrução, que é o agendamento imediato de um envio.

```
1  void wakeUp() {
2    if (!digitalRead(REED)) {
3      detachInterrupt(digitalPinToInterrupt(REED));
4      flag = true;
5    }
6  }
```

Esta forma pode parecer confusa e desnecessária, porém a biblioteca LMIC se difere das demais por se basear unicamente em eventos agendados, portanto não seria possível agendar outro evento pois o anterior ainda está pendente.

3.2.4 Telegraf

Com o programa em execução e a aplicação criada no TTN, é necessário enviar os dados para um banco, essa integração é dada através do Telegraf. Após sua instalação, é necessário editar o arquivo *telegraf.conf*, realizando apenas duas alterações. A primeira é em [[outputs.influxdb]], sendo necessário retirar o comentário (#) dos parâmetros "**urls**"e "**database**", em "**urls**", modificar para ["http://localhost:8086"].

A segunda alteração é a adição das seguintes linhas de código ao final:

```
1 [[inputs.mqtt_consumer]]
2    name_override = "NOME_DA_APLICACAO"
3    servers = ["tcp://brazil.thethings.network:1883"]
4    qos = 0
5    persistent_session = false
6    client_id = ""
7    connection_timeout = "15s"
8    topics = [ "+/devices/+/up" ]
9    username = "USUARIO"
10   password = "SENHA"
11   data_format = "json"
```

São necessários editar os seguintes parâmetros conforme a aplicação:

- name_override;
- username;
- password.

Com estas alterações realizadas, basta executar o *Telegraf*.

3.2.5 InfluxDB

É necessária a criação de um banco de dados para que as informações referentes à abertura e fechamento de chaves fusíveis sejam armazenadas. Após a instalação do InfluxDB, a criação de um banco de dados é simples, através de linha de comando, se inicia o InfluxDB, que deverá retornar algo no seguinte padrão:

```
Connected to http://localhost:8086 version v1.7.9
InfluxDB shell version: v1.7.9
```

A criação de um banco de dados se dá por:

```
CREATE DATABASE "telegraf"
```

E a de um usuário e senha, por:

```
CREATE USER "user" WITH PASSWORD 'passwd'
```

3.2.6 Grafana

Por fim, é desenvolvida uma interface gráfica para mostrar a localização das chaves fusíveis no mapa, assim como seu estado. Isto é feito através do *software Grafana*. Após instalado, é necessário adicionar uma fonte de dados, e após, criar uma tela, ou como é chamada pelo *software*, *dashboard*. A tela inicial após a instalação é apresentada na Figura 3.16.

Primeiramente, a adição do banco de dados previamente configurado se dá através da opção *Add data source*. Basta procurar por *InfluxDB* e selecionar esta fonte de dados. Na página resultante, apenas é necessária a alteração de 4 campos:

- *URL*: Inserir o IP e a porta do servidor InfluxDB;
- *Database*: Nome do banco de dados cadastrado previamente;
- *User*: Usuário;
- *Password*: Senha.

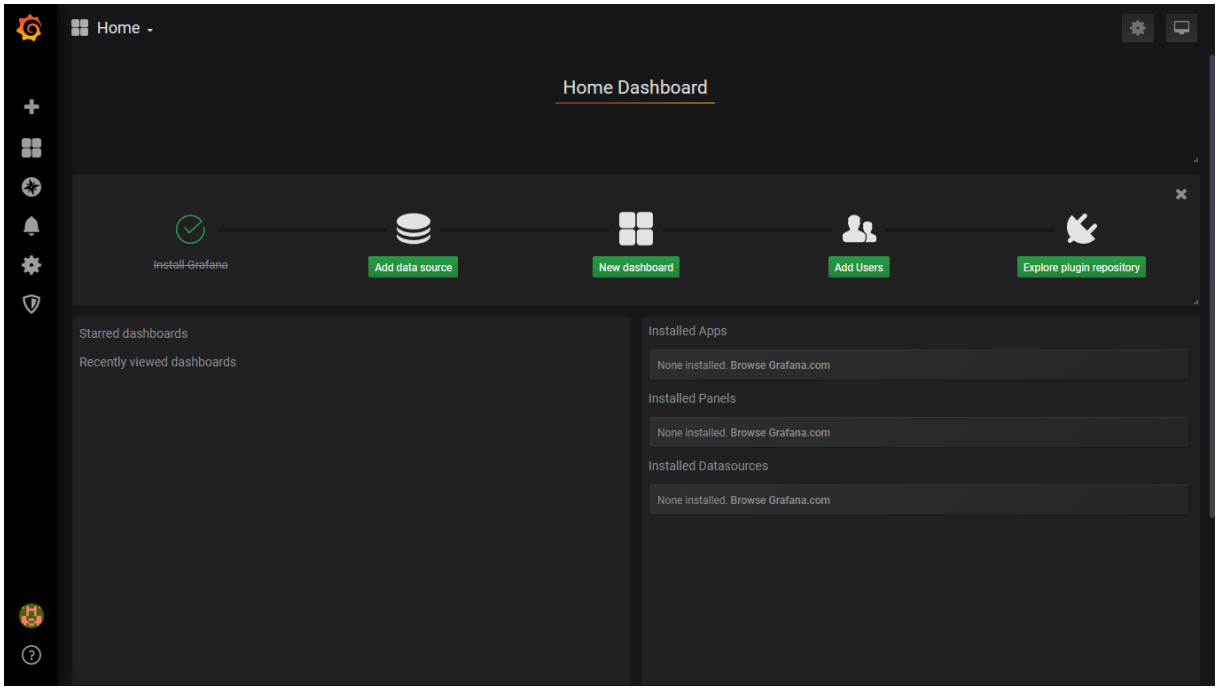


Figura 3.16 – Tela inicial após a instalação do *Grafana*.

Fonte: Autor.

Em *Save & Test*, é possível testar se a conexão está funcionando corretamente, para então partir para a próxima etapa que é a criação do *dashboard*.

A identificação da localização do dispositivo identificador de abertura de chave fusível se dá através do parâmetro *Device Address*, único do dispositivo. Para importar estes dados no *dashboard*, é necessário tratá-los como variáveis. Em *Dashboard settings*, no canto superior direito, e em *Variables*. É necessário adicionar 2 variáveis, conforme relacionado nas Tabelas 3.4 e 3.5.

Tabela 3.4 – Parâmetros da primeira variável - *Grafana*

Parâmetro	Valor
Name	*var_topic_dev_name*
Data source	*InfluxDB*
Refresh	*On Dashboard Load*
Query	*SHOW TAG VALUES ON "telegraf" FROM "banco" WITH KEY=topic*
Regex	*/.*devices/([^/]*).*/*

Fonte: Autor.

Onde "banco" e "ttn_ application" são respectivamente o nome do banco de dados do InfluxDB e o *Application ID* do TTN, ambos configurados previamente.

Por fim, o mapa é adicionado ao *dashboard* através do *Leaflet*, para a criação de mapas com marcadores interativos inteiramente de código aberto. São duas opções de marcadores utilizados: um vermelho e um verde, cuja posição no mapa depende da variá-

Tabela 3.5 – Parâmetros da segunda variável - *Grafana*

Parâmetro	Valor
Name	*var_topic_dev_trip*
Data source	*InfluxDB*
Refresh	*On Time Range Change*
Query	*SELECT last("payload_fields_trip") FROM "banco" WHERE ("topic"= 'ttn_application/devices/[[var_topic_dev_name]]/up')*

Fonte: Autor.

vel "var_topic_dev_name". A variável "var_topic_dev_trip" define a cor do marcador utilizada, e é monitorada conforme o parâmetro *Time Range*, configurável no *dashboard*.

Para tal, é necessário adicionar um painel, em *Add Panel*, no canto superior direito, selecionar *Choose Visualization*, e por fim escolher "html" na caixa de seleção de tipo de texto, inserindo o seguinte código:

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.4.0/dist/leaflet.css"/>
<script src="https://unpkg.com/leaflet@1.4.0/dist/leaflet.js"></script>
<div id="map" style="width: 100%; height: 100%;"></div>
<script>
var nomeLocalMeter = '[[var_topic_dev_name]]';
var trip_status = '[[var_topic_dev_trip]]';
var redIcon = new L.Icon({
  iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-red.png',
  shadowUrl: 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});
var greenIcon = new L.Icon({
  iconUrl: 'https://raw.githubusercontent.com/pointhi/leaflet-color-markers/master/img/marker-icon-green.png',
  shadowUrl: 'https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png',
  iconSize: [25, 41],
```

```
19   iconAnchor: [12, 41],
20   popupAnchor: [1, -34],
21   shadowSize: [41, 41]
22 });
23 var locations = [
24   ["dev01", -29.711011, -53.715892],    //dev01
25   ["dev02", -29.711011, -53.715892]     //dev02
26   ];
27 setTimeout(function(){
28   var map = L.map('map', {
29     center: [-29.7203067, -53.7147709],  // Map center
30     zoom: 14.5
31   });
32   L.tileLayer('https://{s}.tile.osm.org/{z}/{x}/{y}.png', {
       attribution: '<a>href="https://www.linkedin.com/in/
       flaviogarletreck/"␣target="_blank">Flavio␣Garlet␣Reck</a
       >|&copy;<a>href="https://osm.org/copyright"target="_blank
       ">OpenStreetMap</a>'}).addTo(map);
33 for (var i = 0; i < locations.length; i++) {
34   if (trip_status==1)  L.marker([locations[i][1],locations[i
       ][2]], {icon:redIcon}).addTo(map);
35   else L.marker([locations[i][1],locations[i][2]], {icon:
       greenIcon}).addTo(map);
36   }
37 }, 1000);
38 </script>
```

A adição e modificação de pontos se dá a partir do vetor *locations*, onde "dev01" é o parâmetro *Device ID* definido para o dispositivo, e os parâmetros seguintes são a Latitude e Longitude do ponto.

4 RESULTADOS

Neste capítulo, serão abordados os principais resultados obtidos com o desenvolvimento de um protótipo. As configurações e técnicas descritas nos capítulos anteriores foram integradas em um produto. Através do *console* do TTN, será possível a análise se os dados estão sendo enviados corretamente, e por fim, resultados do sistema supervisório desenvolvido serão apresentados.

4.1 DESENVOLVIMENTO DE UM PROTÓTIPO

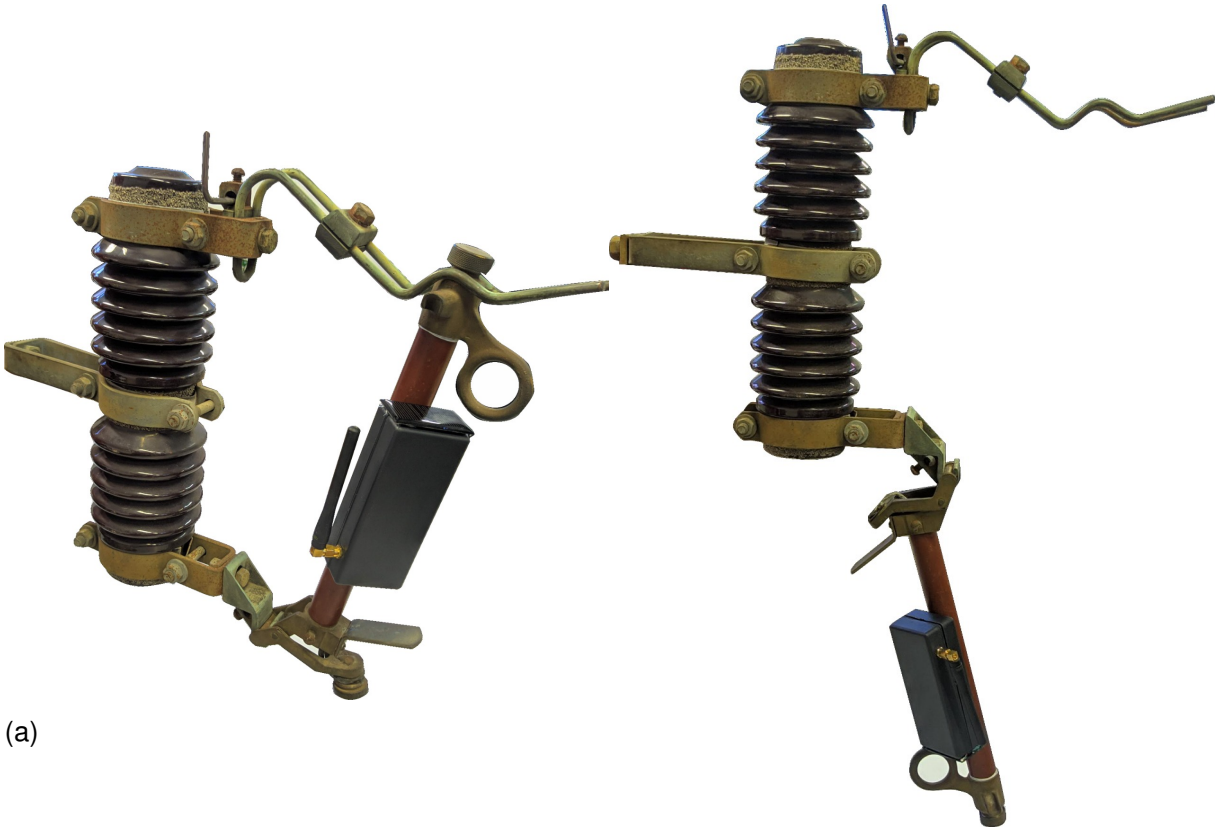
Com a escolha dos componentes conforme mencionado nos tópicos anteriores, foi desenvolvido um protótipo de dispositivo identificador de abertura de chaves fusíveis, apresentado na Figura 4.1.

Figura 4.1 – Protótipo desenvolvido.

Fonte: Autor.

O dispositivo possui vedação para impedir a entrada de água da chuva, sua instalação em uma chave fusível se dá através de um lacre, abraçando o porta-fusível da chave. O tornando um dispositivo universal, compatível com praticamente qualquer tipo de chave fusível.

Na Figura 4.2a, seu uso com uma chave fusível fechada é apresentado. Já na Figura 4.2b, é mostrado quando a chave fusível é aberta.



(a)

(b)

Figura 4.2 – Dispositivo implementado em uma chave fusível (a)fechada e (b)aberta

Fonte: Autor.

4.1.1 Estimativa de custo de produção

Para o desenvolvimento deste trabalho, foi manufaturada uma unidade de um protótipo, portanto, preços de varejo são maiores do que atacado. Uma estimativa de custo de produção do protótipo está apresentada na Tabela 4.1.

Tabela 4.1 – Estimativa de custo de produção do protótipo

Item	**Preço (R$)**
Microcontolador Arduino Pro Mini	13,75
Módulo LoRa RFM95	56,00
Módulo Bateria TP4056	10,99
Bateria	34,00
Painel Solar	16,00
Caixa	10,00
Antena 915 MHz	20,00
Diodos	0,10
Cabos	1,00
Total	161,84

4.2 CONSOLE THE THINGS NETWORK

Durante testes, o dispositivo foi posto em posição de operação normal, com a chave fusível fechada. Nesta configuração, a cada dia, um dado é enviado para garantir que o sistema está funcionando. Este dado é recebido na plataforma do TTN, conforme visto na Figura 4.3.

Após o dado cru, neste caso, *QNQQAyYAwAABHnRk6RtO*, ser decriptado e interpretado pela aplicação, temos os seguintes parâmetros recebidos:

- alive: 1 - O sistema está vivo;
- trip: 0 - A chave encontra-se fechada.

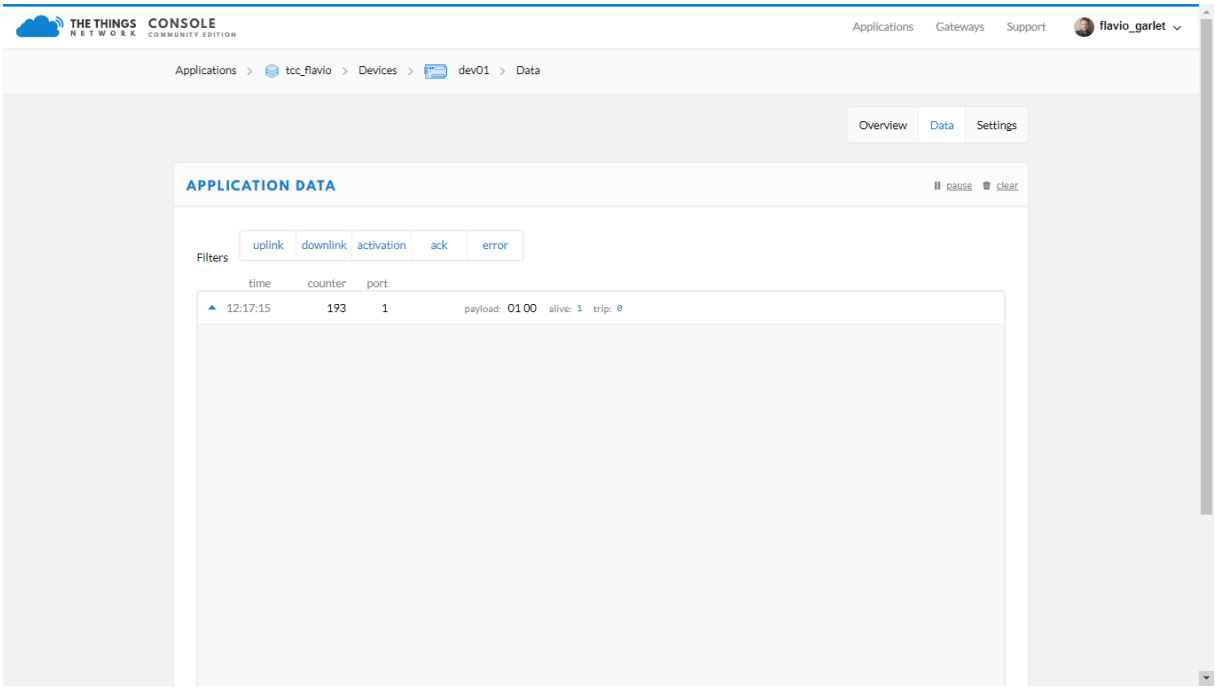


Figura 4.3 – Dado de operação normal no TTN.

Fonte: Autor.

Porém, quando a chave é aberta, um dado é enviado com esta informação. Na Figura 4.4 é apresentado um dado que é enviado pra plataforma quando uma chave é aberta.

O dado cru, neste caso, *QNQQAyYAAAABgnvbns8c*, é decriptado pela aplicação, resultando nos seguintes parâmetros:

- alive: 1 - O sistema está vivo;
- trip: 1 - A chave encontra-se aberta.

Estes mesmos parâmetros são enviados novamente a cada minuto até a chave ser fechada, para evitar possíveis falhas no envio e garantir a confiabilidade do sistema.

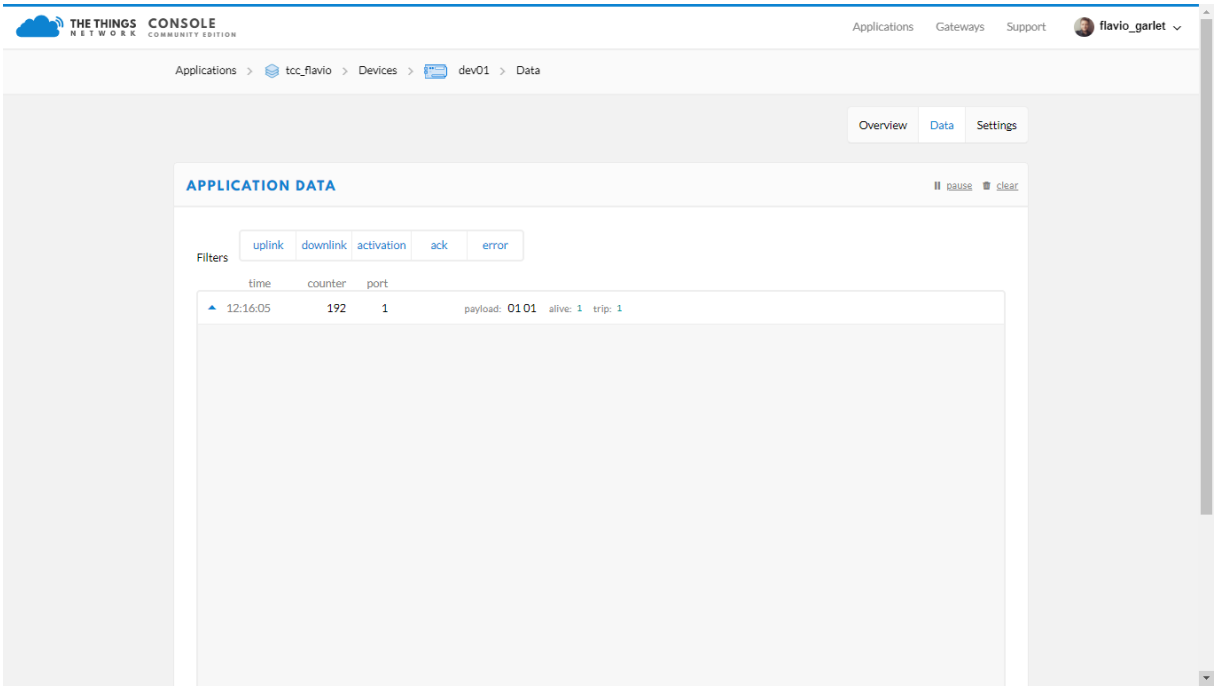


Figura 4.4 – Dado de chave aberta no TTN.

Fonte: Autor.

4.3 SISTEMA SUPERVISÓRIO

Um sistema supervisório foi criado, e nele um único ponto foi registrado, como dispositivo de testes dentro da Universidade Federal de Santa Maria. Uma operação normal é verificada na Figura 4.5. Nela, temos a posição da chave cadastrada e o ícone verde representa que a mesma se encontra fechada.

Porém, quando a chave fusível abre, um dado indicando o ocorrido é enviado, e o ícone referente a mesma tem sua cor alterada, para um ícone vermelho, indicando a abertura da chave. Essa situação é indicada na Figura 4.6. O mesmo dado é enviado a cada minuto para garantir a confiabilidade do sistema.

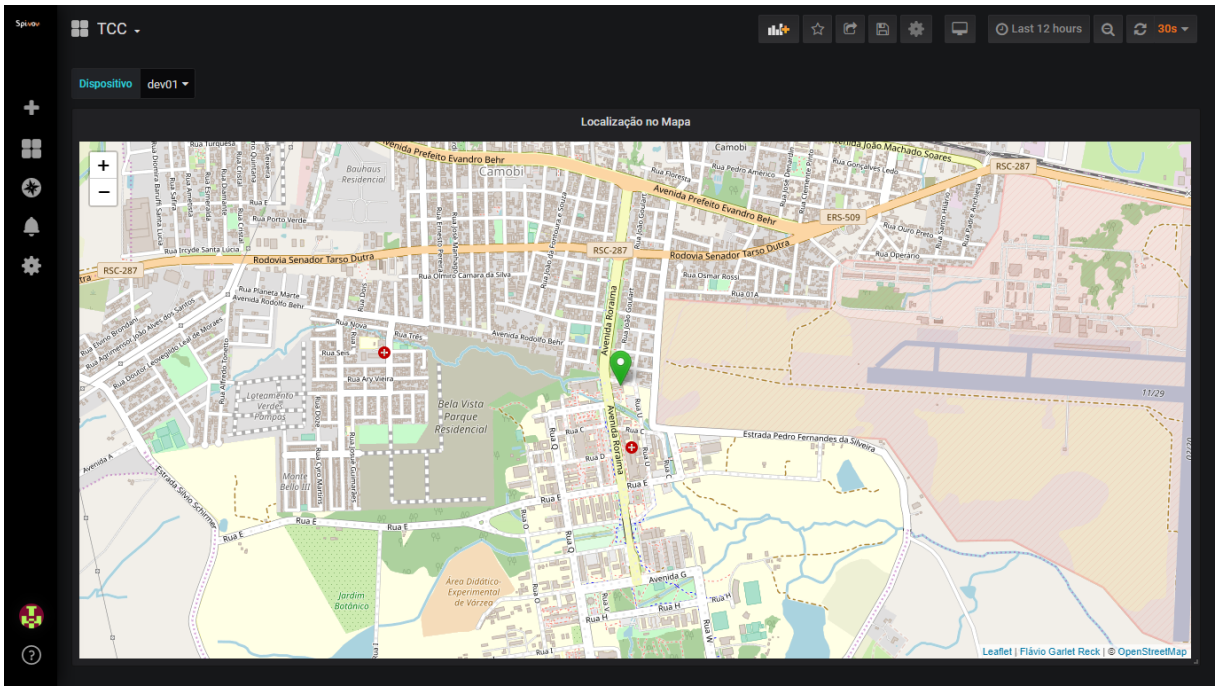


Figura 4.5 – Sistema supervisório durante operação normal.

Fonte: Autor.

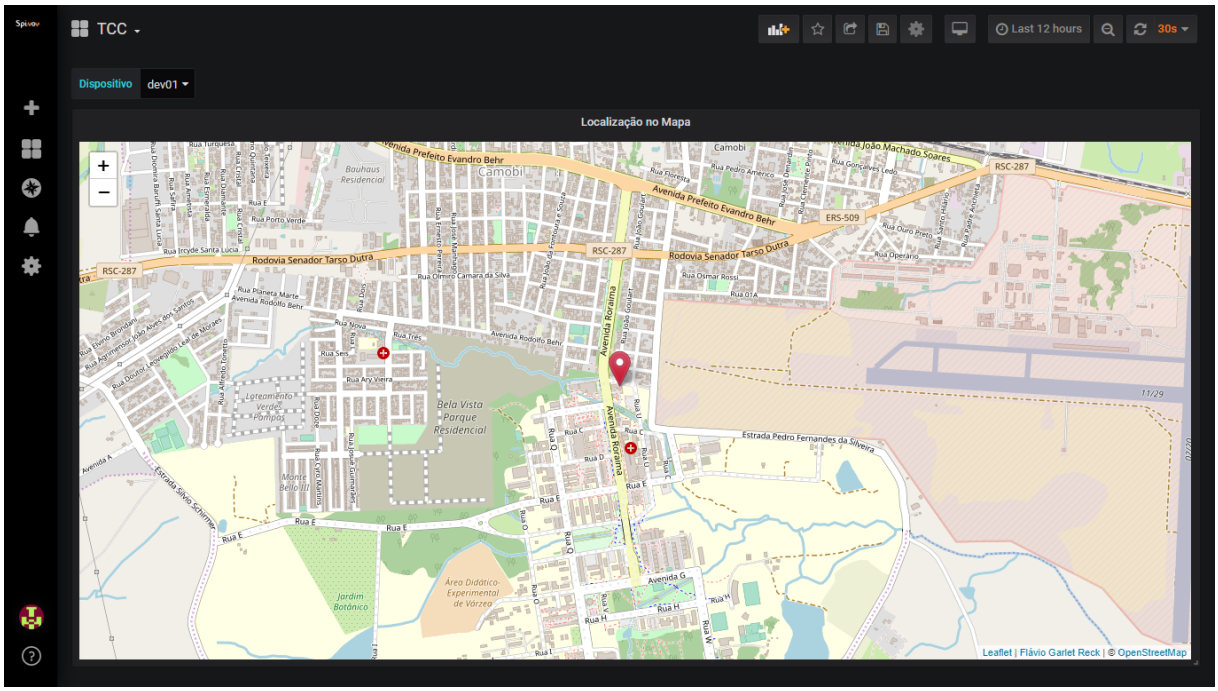


Figura 4.6 – Sistema supervisório com uma chave fusível aberta.

Fonte: Autor.

5 CONCLUSÃO

O crescimento dos sistemas de distribuição das cidades traz junto um problema relacionado com a abertura de chaves fusíveis presentes no mesmo. Quando uma falta é verificada e um consumidor tem seu fornecimento de energia interrompido, é necessário que os clientes afetados pela interrupção entrem em contato com a concessionária, informando o problema. Com várias ligações, a concessionária consegue estimar o ramal defeituoso, para enviar uma equipe de manutenção. Esse processo geralmente é demorado, podendo resultar em multas por não conformidade com índices de qualidade estipulados pela Aneel.

Visando propor uma solução para este problema, foi apresentado um sistema para identificação de abertura de chaves fusíveis. Através de um sensor magnético, um microcontrolador ATmega328p identifica quando há uma interrupção pelo sensor, causada pela abertura de uma chave fusível, e através do módulo RFM95, envia um dado através de uma rede LoRaWAN, que apresenta grande cobertura e baixo consumo de potência.

Para o desenvolvimento de uma solução completa, também foi criada uma plataforma *online*, que atua como um sistema supervisório, em que a localização de chaves fusíveis monitoradas são inseridas em um mapa. Quando um dado referente à abertura da chave é enviado pelo servidor de rede, a cor do ícone é alterada conforme o estado da chave, gerando um alerta visual em que a concessionária possui a localização de uma chave aberta, podendo enviar uma equipe de manutenção com maior agilidade.

Ao todo, foram utilizadas diversas plataformas, que durante o desenvolvimento deste trabalho, foram integradas. Dentre elas, a utilização da biblioteca LMIC para Arduino, que gerencia os envios por LoRa. Através do *console* do TTN, *gateway* e aplicação são criados, para receber os dados por LoRaWAN e enviá-los pela internet. O dado recebido e interpretado no servidor de rede do TTN é encaminhado para um banco de dados do *InfluxDB* através do *Telegraf*. Por fim, esse dado é acessado pelo *Grafana* e exibido para o usuário.

Desta maneira, conclui-se que tanto os objetivos gerais como os objetivos específicos foram alcançados. O sistema desenvolvido é funcional e robusto, sendo praticamente universal para qualquer modelo de chave fusível.

REFERÊNCIAS BIBLIOGRÁFICAS

CARLOS Henrique Barriquello.

ADELANTADO, F. et al. Understanding the limits of lorawan. **IEEE Communications magazine**, IEEE, v. 55, n. 9, p. 34–40, 2017.

ALLIANCE, L. White paper: A technical overview of lora and lorawan. **The LoRa Alliance: San Ramon, CA, USA**, p. 7–11, 2015.

ANATEL. **Resolução nº 680, de 27 de junho de 2017**. 2017. <https://www.anatel.gov.br/legislacao/resolucoes/2017/936-resolucao-680>. Acesso em 10/11/2019.

ANEEL. Procedimentos de distribuição de energia elétrica no sistema elétrico nacional–prodist. **Agência Nacional de Energia Elétrica**, 2018.

____. **Compensação pela Transgressão dos Limites de Continuidade**. 2019. <https://www.aneel.gov.br/indicadores-de-compensacao-de-continuidade>. Acesso em 07/11/2019.

ANGERER, F. M. New developments in faulted circuit indicators help utilities reduce cost and improve service. In: IEEE. **2008 IEEE Rural Electric Power Conference**. [S.l.], 2008. p. B4–B4.

ANUPRIYA, K. et al. Integrating zigbee and sub ghz devices for long range networks. In: IEEE. **2016 Online International Conference on Green Engineering and Technologies (IC-GET)**. [S.l.], 2016. p. 1–5.

ARDUINO. **Arduino**. 2019. <https://arduino.cc>. Acesso em 09/11/2019.

BANZI, M.; SHILOH, M. **Getting started with Arduino: the open source electronics prototyping platform**. [S.l.]: Maker Media, Inc., 2014.

BARRIQUELLO, C. H. **Pesquisa e desenvolvimento de uma rede LoRaWAN na UFSM**. 2017. <https://portal.ufsm.br/projetos/publico/projetos/view.html?idProjeto=57712>. Acesso em 24/11/2019.

BORDIER, T. P. N. **Monitorer votre infra avec Telegraf, InfluxDB et Grafana**. 2017. <https://blog.octo.com/monitorer-votre-infra-avec-telegraf-influxdb-et-grafana/>. Acesso em 20/11/2019.

CAPELINI, R. M. et al. Methodology for fast fault location in overhead distribution networks by the application of temporary georeferenced fault indicators. In: IEEE. **2016 IEEE International Conference on High Voltage Engineering and Application (ICHVE)**. [S.l.], 2016. p. 1–4.

CHEN, M. et al. Narrow band internet of things. **IEEE access**, IEEE, v. 5, p. 20557–20577, 2017.

CHIRPSTACK. **ChirpStack, open-source LoRaWAN® Network Server stack**. 2019. <https://www.chirpstack.io/>. Acesso em 24/11/2019.

HOPERF. **Low Power Long Range Transceiver Module**. 2016. <https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf>. Acesso em 14/11/2019.

IEEE. Ieee guide for the application of faulted circuit indicators on distribution circuits. In: IEEE. **IEEE Std 1610-2016**. [S.l.], 2016. p. 01–26.

KARWATH, A. **This image shows an electrical reed switch**. 2005. Acesso em 13/11/2019. Disponível em: <\url{https://en.wikipedia.org/wiki/File:Reed_switch_(aka).jp}. >

KYCHKIN, A. et al. Architecture of compressor equipment monitoring and control cyber-physical system based on influxdata platform. In: IEEE. **2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)**. [S.l.], 2019. p. 1–5.

LEE, S.-J. et al. An intelligent and efficient fault location and diagnosis scheme for radial distribution systems. **IEEE transactions on power delivery**, IEEE, v. 19, n. 2, p. 524–532, 2004.

MAURIZIO, G. **Type "MZP"Distribution Standard Cutout**. 2016. <http://www.maurizio.comercial.ws/arqsist/loja/BT-028-02en.pdf>. Acesso em 13/11/2019.

MICROCONTROLLERSLAB. **How to use Arduino interrupts explained with examples**. 2018. <https://microcontrollerslab.com/use-arduino-interrupts-examples/>. Acesso em 10/11/2019.

MOLLET, R. Overcurrent protection of dc power plant equipment using modern high performance current limiting fuses. In: IEEE. **Proceedings of INTELEC 95. 17th International Telecommunications Energy Conference**. [S.l.], 1995. p. 379–383.

MUTHANNA, M. S. A. et al. Development of intelligent street lighting services model based on lora technology. In: IEEE. **2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)**. [S.l.], 2018. p. 90–93.

NANJING. **TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8**. 2017. <https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf>. Acesso em 20/11/2019.

NEUY, J. D. **ARDUINO LOW POWER - HOW TO RUN ATMEGA328P FOR A YEAR ON COIN CELL BATTERY**. 2017. <http://www.home-automation-community.com/arduino-low-power-how-to-run-atmega328p-for-a-year-on-coin-cell-battery/>. Acesso em 20/11/2019.

PLET, C. A. **Fault response of inverter-based distributed generation**. 2012. Tese (Doutorado) — Imperial College London, 2012.

RAZA, U.; KULKARNI, P.; SOORIYABANDARA, M. Low power wide area networks: An overview. **IEEE Communications Surveys & Tutorials**, IEEE, v. 19, n. 2, p. 855–873, 2017.

ROCKETSCREAM. **Low-Power**. 2019. <https://github.com/rocketscream/Low-Power>. Acesso em 09/11/2019.

SEL. **Indicador de Falta AutoRANGER Aareo**. 2019. <https://selinc.com/pt/products/AR360/>. Acesso em 09/11/2019.

SHORT, T. A. **Electric power distribution handbook**. [S.l.]: CRC press, 2018.

SIGFOX. **Sigfox QA platform**. 2015. <https://ask.sigfox.com/index.html>. Acesso em 10/11/2019.

SORNIN, N. et al. Lora specification 1.0. **Lora Alliance Standard Specification. Available online: https://lora-alliance. org/sites/default/files/2018-04/lorawantm_specification_-v1**, v. 1, 2015.

TELEGRAF. **Why use Telegraf?** 2019. <https://www.influxdata.com/time-series-platform/telegraf/>. Acesso em 11/11/2019.

TTN, T. T. N. **LoRaWAN Overview**. 2019. <https://www.thethingsnetwork.org/docs/lorawan/>. Acesso em 10/11/2019.

VANGELISTA, L. Frequency shift chirp modulation: The lora modulation. **IEEE Signal Processing Letters**, IEEE, v. 24, n. 12, p. 1818–1821, 2017.

VELASCO, G. D. N. **Arborização viária X sistemas de distribuição de energia elétrica: avaliação dos custos, estudo das podas e levantamento de problemas fitotécnicos.** 2003. Tese (Doutorado) — Universidade de São Paulo, 2003.

WIXTED, A. J. et al. Evaluation of lora and lorawan for wireless sensor networks. In: IEEE. **2016 IEEE SENSORS**. [S.l.], 2016. p. 1–3.

APÊNDICE A – CÓDIGO DESENVOLVIDO PARA O MICROCONTROLADOR

```
/*************************************************************
  CONFIGURATION MODULES (ENABLE/DISABLE)
*************************************************************/
#define USER_LOWPOWER

/*************************************************************
  AUXILIARY LIBRARIES
*************************************************************/
#include <lmic.h>
#include <hal/hal.h>
#include <avr/wdt.h>
#include <LoraPayload.h>
#ifdef USER_LOWPOWER
#include <LowPower.h>
#endif //end if USER_LOWPOWER

/*************************************************************
  FIRMWARE VERSION x.x
*************************************************************/
#define FIRMWARE_VERSION "1.0" //Firmware version

/*************************************************************
  VARIABLES AND DEFINITIONS
*************************************************************/
// Set UART to debugging
#define SERIAL_DEBUG            Serial  // UART for de
    debugging
#define SERIAL_BAUD             9600    // Baud default

// Set pins RFM95
#define LORAWAN_RFM95_PIN_NSS   9       //Antes era 10
#define LORAWAN_RFM95_PIN_RST   5
#define LORAWAN_RFM95_PIN_D0    4
#define LORAWAN_RFM95_PIN_D1    3

```

```
35 // Set parameters LoRaWAN
36 #define LORAWAN_INTERVAL        15      // Limit mininal of 10s
       and max 1hour
37 #define KEEP_ALIVE              35      // Limit mininal of 10s
       and max 1hour
38 #define SLEEP_COMPENSATION      1
39
40 #define LORAWAN_SESSION_PORT    1       // Port session
41 #define LORAWAN_ACK             false   // ACK true or false
42 #define LORAWAN_ATTEMPTS        3       // Transmit attempts
      for confirmed frames
43
44 #define LORAWAN_DR              3       // Data rate (DR0 - DR5
      )
45 #define LORAWAN_DN2DR           8       // AU915 Data rate (DR8
      )
46 #define LORAWAN_DR_500          6       // AU915 Data rate (DR6
      )
47
48 #define LORAWAN_START_CH        8       // Start channel 125KHz
       (0-63)
49 #define LORAWAN_END_CH          15      // End channel 125KHz
      (0-63)
50 #define LORAWAN_CH_500          0       // Channel 500KHz
      (64-71) if 0 disable
51
52 // Set pins Reed Switch
53 #define REED                    2
54
55 // Set keys LoRaWAN
56 uint8_t LORAWAN_DEVADDR[4] = { 0x26, 0x03, 0x10, 0xD4 };
57 uint8_t LORAWAN_NWSKEY[16] = { 0xD4, 0xBA, 0x79, 0x33,
58 0xDB, 0x78, 0xAB, 0x6D, 0x7C, 0x1F, 0xEA, 0xF0, 0xB2, 0xBE, 0
      x77, 0x0F };
59 uint8_t LORAWAN_APPKEY[16] = { 0x6B, 0xF2, 0x95, 0xBF, 0xE7, 0
      x46, 0x1C, 0x68, 0xA7, 0xAB, 0x84, 0x32, 0xA4, 0x95, 0xF8, 0
      x15 };
60
61 // Set utility LoRaWAN
```

```
62 #define ARRAY4_TO_HEX1(p) (((u4_t)((p)[0]) << 8 | (p)[1]) << 8
     | (p)[2]) << 8 | (p)[3] // Convert u4_t in u1_t(array)
63 #define IS_CHANNEL_125khz(c) (c<64)
64 #define IS_CHANNEL_500khz(c) (c>=64 && c<72)
65 #define ENABLED_CHANNEL(chnl) ((LMIC.channelMap[(chnl >> 4)] &
     (1<<(chnl & 0x0F))) != 0)
66
67 // Set pin mapping RFM95
68 const lmic_pinmap lmic_pins = {
69   .nss = LORAWAN_RFM95_PIN_NSS,
70   .rxtx = LMIC_UNUSED_PIN,
71   .rst = LORAWAN_RFM95_PIN_RST,
72   .dio = {LORAWAN_RFM95_PIN_D0, LORAWAN_RFM95_PIN_D1,
       LMIC_UNUSED_PIN}, // D0, D1, D2
73 };
74
75 // Counter channels 125KHz e 500KHz and controller
76 uint8_t COUNT_DATA = 0;
77 uint8_t COUNT_CH_125khz = 0;
78 uint8_t COUNT_CH_500khz = 0;
79
80 volatile bool flag = false;
81
82 /*****************************************************
83   INSTANCE OF OBJECTS
84 *****************************************************/
85 // osjob_t object
86 static osjob_t sendjob;
87
88 // LoraPayload object
89 LoraPayload lpp(51);    // Create a buffer of 51 bytes to store
     the payload
90
91 /*****************************************************
92   FUNCTION SETUP
93 *****************************************************/
94 void setup()
95 {
96   // Set pin direction I/O
```

```
97    pinMode(REED, INPUT_PULLUP);
98
99    // Set it for a speed of 9600
100   SERIAL_DEBUG.begin(SERIAL_BAUD);
101
102   // Init and config stack LoRaWAN
103   init_LoRaWAN(LORAWAN_SESSION_PORT, LORAWAN_DEVADDR,
         LORAWAN_NWSKEY, LORAWAN_APPKEY,  // Set port and keys (
         Device address, Network session, Application session)
104                 LORAWAN_START_CH, LORAWAN_END_CH, LORAWAN_CH_500
                       ,                          // Select channel
                       enable
105                 LORAWAN_ATTEMPTS,

                       // Transmit attempts for confirmed frames
106                 LORAWAN_DR,

                       // Set data rate TX for uplink
107                 LORAWAN_DN2DR);

                       // Set data rate RX2 for downlink
108
109   // Enable Interruption
110   attachInterrupt(digitalPinToInterrupt(REED), wakeUp, FALLING)
         ;
111
112   // Start job
113   do_send(&sendjob);
114 }
115
116 /*************************************************************
117   FUNCTION LOOP
118 *************************************************************/
119 void loop()
120 {
121   os_runloop_once(); // Send data timer (LoRa)
122   wdt_reset();       // Feed watchdog
123 }
124
```

```
125 /*************************************************************
126   INTERRUPT
127 *************************************************************/
128 void wakeUp() {
129   if (!digitalRead(REED)) {
130     detachInterrupt(digitalPinToInterrupt(REED));
131     Serial.println("Interrupt");
132     Serial.flush();
133     flag = true;
134   }
135 }
136
137 /*************************************************************
138   FUNCTION EVENT TIME SYSTEM
139 *************************************************************/
140 void do_send(osjob_t* j)
141 {
142   detachInterrupt(digitalPinToInterrupt(REED));
143   wdt_enable(WDTO_8S);
144   SERIAL_DEBUG.flush();
145   SERIAL_DEBUG.print(F("Send␣TX..␣("));
146   SERIAL_DEBUG.print(LMIC.seqnoUp);
147   SERIAL_DEBUG.println(F(")"));
148
149   lpp.reset(); // clear the buffer
150
151   uint8_t alive = 1;
152   uint8_t trip;
153
154   if (!digitalRead(REED)) trip  = 1;
155   else trip  = 0;
156
157   lpp.addUint8(alive);
158   lpp.addUint8(trip);
159
160   // Auto select DR
161   if (COUNT_CH_500khz)
162   {
```

```
163     if (COUNT_CH_125khz) LMIC.datarate = (++COUNT_DATA % (
          COUNT_CH_125khz + COUNT_CH_500khz)) ? LORAWAN_DR :
          LORAWAN_DR_500;
164     else LMIC.datarate = LORAWAN_DR_500;
165   }
166
167   // Prepare upstream data transmission at the next possible
        time.
168   if (!sendTX_LoRaWAN(LORAWAN_SESSION_PORT, lpp.getBuffer(),
        lpp.getSize(), LORAWAN_ACK)) AVRreset(); // Reset by
        watchdog
169 }
170
171 /*************************************************************
172   FUNCTION CALLBACK EVENTS
173 *************************************************************/
174 void onEvent (ev_t ev)
175 {
176   switch (ev)
177   {
178     case EV_TXCOMPLETE:
179       wdt_reset();
180       SERIAL_DEBUG.println("Ok");
181       SERIAL_DEBUG.flush();                          // Clear
            Serial
182       if (!digitalRead(REED)) {
183         sleep((uint32_t)(LORAWAN_INTERVAL * 1000));    // Sleep
              now
184         do_send(&sendjob);             // Start transmission
              LoRaWAN
185       }
186       else {
187         attachInterrupt(digitalPinToInterrupt(REED), wakeUp,
              FALLING);
188         for (int i = 0; i < 8; i++) {
189           if (!flag) {
190             LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
191           }
192           else {
```

```
193             break;
194           }
195         }
196         do_send(&sendjob); // Start transmission LoRaWAN
197       }
198       flag = false;
199       break;
200   }
201 }
202
203 /*************************************************
204   FUNCTION INIT STACK LORAWAN
205 *************************************************/
206 void init_LoRaWAN(uint8_t port, uint8_t* devaddr, uint8_t*
      nwkskey, uint8_t* appskey,
207                   uint8_t channelStart_125khz, uint8_t
                        channelEnd_125khz, uint8_t channel_500khz,
208                   uint8_t retry,
209                   uint8_t upDr,
210                   uint8_t dn2Dr)
211 {
212   // LMIC init
213   os_init();
214
215   // Reset the MAC state. Session and pending data transfers
        will be discarded.
216   LMIC_reset();
217
218   // LoRaWAN NwkSKey, network session key
219   // LoRaWAN AppSKey, application session key
220   // LoRaWAN end-device address (DevAddr)
221   // This is the default Semtech key, which is used by the
        early prototype TTN network.
222   LMIC_setSession(port, ARRAY4_TO_HEX1(devaddr), nwkskey,
        appskey);
223
224   // We'll disable all 72 channels used by TTN
225   for (uint8_t ch = 0; ch < 72; ch++) LMIC_disableChannel(ch);
226
```

```
227   // Select channel enable channel 125KHz (0-63)
228   if (channelStart_125khz >= 0 && channelEnd_125khz < 64) for (
          uint8_t ch = channelStart_125khz; ch <= channelEnd_125khz;
           ch++) LMIC_enableChannel(ch);
229
230   // Select channel enable channel 500KHz (64-71) if 0 disable
231   if (channel_500khz > 63 && channel_500khz < 72)
          LMIC_enableChannel(channel_500khz);
232
233   // Init Channel random
234   LMIC.chRnd = random(channelStart_125khz, channelEnd_125khz) -
           1;
235
236   // Transmit attempts for confirmed frames
237   LMIC.txconfattempts = constrain(retry, 1, 15);
238
239   // Disable data rate adaptation
240   LMIC_setAdrMode(0);
241
242   // Disable link check validation
243   LMIC_setLinkCheckMode(0);
244
245   // RX2 DR8 AU915-928 (Working in TTN)
246   LMIC.dn2Dr = dn2Dr;
247
248   // Use a medium spread factor. This can be increased up to
          SF12 for
249   // better range, but then the interval should be (
          significantly)
250   // lowered to comply with duty cycle limits as well.
251   // Set data rate and transmit power for uplink (note: txpow
          seems to be ignored by the library)
252   LMIC_setDrTxpow(upDr /*Data rate (DR0 - DR5)*/, 20); // Power
           option: 2, 5, 8, 11, 14 and 20
253
254   //Let LMIC compensate for +/- 1% clock error
255   LMIC_setClockError(MAX_CLOCK_ERROR * 1 / 100);
256 }
257
```

```
258 /*************************************************
259   FUNCTION SEND DATA TX
260 *************************************************/
261 bool sendTX_LoRaWAN(uint8_t port_tx, uint8_t *data_tx, uint8_t
      data_tx_len, uint8_t confirmed)
262 {
263   // Check if there is not a current TX/RX job running
264   if (!(LMIC.opmode & OP_TXRXPEND))
265   {
266     // Prepare upstream data transmission at the next possible
          time.
267     LMIC_setTxData2(port_tx, data_tx, data_tx_len, confirmed);
268     return true; // Transmission success
269   }
270   return false;
271 }
272
273 /*************************************************
274   FUNCTION RECEIVER DATA RX
275 *************************************************/
276 bool getRX_LoRaWAN(uint8_t *data_rx, uint8_t &data_rx_len,
      uint8_t &downPort)
277 {
278   //https://os.mbed.com/teams/Semtech/code/LMiC
279   // Check if we have a downlink on either RX1 or RX2 windows
280   if ((LMIC.txrxFlags & (TXRX_DNW1 | TXRX_DNW2)) && LMIC.
        dataLen)
281   {
282     for (uint8_t i = 0; i < LMIC.dataLen; i++) *data_rx++ =
          LMIC.frame[LMIC.dataBeg + i];
283     data_rx_len = LMIC.dataLen; // Get length payload
284     downPort = LMIC.frame[LMIC.dataBeg - 1]; // Get Port
285     return true;
286   }
287   return false;
288 }
289
290 /*************************************************
291   FUNCTION VERIFY ACK
```

```
292 *****************************************************/
293 bool availableACK_LoRaWAN()
294 {
295   if ((LMIC.txrxFlags & TXRX_ACK) || (LMIC.txrxFlags &
        TXRX_NACK)) return true;
296   return false;
297 }
298
299 /*****************************************************
300   FUNCTION STATUS ACK
301 *****************************************************/
302 bool getACK_LoRaWAN()
303 {
304   if (LMIC.txrxFlags & TXRX_ACK) return true;
305   return false;
306 }
307
308 #ifdef USER_LOWPOWER
309 /*****************************************************
310   FUNCTION SLEEP
311 *****************************************************/
312 void sleep(uint32_t milliSeconds)
313 {
314   // ATmega328P, ATmega168, ATmega32U4
315   //https://github.com/tnugent97/FutAir/blob/754
        ab65420c5921bac08100ae0baf05414845137/lora/libraries/lmic/
        examples/kotahi-promini-low-power/kotahi-promini-low-power
        .ino#L144
316   while (milliSeconds >= 8000)
317   {
318     LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
319     milliSeconds -= 8000;
320   }
321   if (milliSeconds >= 4000)
322   {
323     LowPower.powerDown(SLEEP_4S, ADC_OFF, BOD_OFF);
324     milliSeconds -= 4000;
325   }
326   if (milliSeconds >= 2000)
```

```
327   {
328     LowPower.powerDown(SLEEP_2S, ADC_OFF, BOD_OFF);
329     milliSeconds -= 2000;
330   }
331   if (milliSeconds >= 1000)
332   {
333     LowPower.powerDown(SLEEP_1S, ADC_OFF, BOD_OFF);
334     milliSeconds -= 1000;
335   }
336 }
337 #endif //end if USER_LOWPOWER
338
339 /*****************************************************
340   FUNCTION SOFTWARE RESET BY WDT
341 *****************************************************/
342 void AVRreset()
343 {
344   // Enable watchdog
345   wdt_enable(WDTO_500MS);
346   // Reset by watchdog
347   while (true);
348 }
```