IoT: rede LoRa para envio de imagens

Victor Emanuel Almeida
Universidade Estadual do Oeste do Paraná (UNIOESTE)
Foz do iguaçu, Brasil
archvictor@protonmail.com

Marco Aurélio Guerra Pedroso
UNIOESTE
Foz do iguaçu, Brasil
marcoguerrapedroso@gmail.com

Antonio Marcos Massao Hachisuca
UNIOESTE
Foz do iguaçu, Brasil
shiro.foz@gmail.com

***Abstract*—Networks using LoRa communication made it possible for devices to exchange messages over long distances and using little energy. In view of the data transfer limitations of these networks, this work implements a network for sending images using algorithms and techniques already consolidated in computer networks to send images in several parts and guarantee their integrity. Devices such as esp32 and esp32-cam were used to implement this network, which execute the stop and wait algorithm. Once the network is implemented, it can be observed from tests that the average time for sending a LoRa message is 2 seconds, so it is possible to send an image of 6810 bytes in 1 minute, being possible to change the resolution of the image linearly varying the shipping time.**

***Resumo*—Redes utilizando comunicação LoRa possibilitaram que dispositivos troquem mensagens a longas distâncias e gastando pouca energia. Tendo em vista as limitações de transferência de dados dessas redes este trabalho implementa uma rede para envio de imagens utilizando algoritmos e técnicas já consolidadas de redes de computadores para enviar as imagens em diversas partes e garantir sua integridade. Para implementação dessa rede foram utilizados dispositivos como esp32 e esp32-cam que executam o algoritmo de stop and wait. Uma vez implementada a rede pode-se observar a partir de testes que o tempo médio de envio de uma mensagem LoRa é de 2 segundos, sendo assim pode-se enviar uma imagem de 6810 bytes em 1 minuto, sendo possível alterar a resolução da imagem variando de forma linear o tempo de envio.**

***Palavras-chave*—LoRa, Rede LPWAN, envio de imagens.**

I. Introdução

Redes *Long Range*, LoRa, e *Long Range Wide Area Network*, LoRaWAN, possibilitaram a comunicação de dispositivos com baixo consumo energético a longas distâncias [1], porém a largura de banda utilizada e a taxa de transmissão ainda são limitadas, sendo assim o presente trabalho propõe a implementação de uma rede de dois dispositivos IoT para recepção e transmissão de imagens utilizando comunicação LoRa.

Para atender objetivo, na seção II buscou compreender conceitos tais como o que são Redes LPWAN seção II-A, o que é LoRa seção II-B, e por fim o que é IoT seção II-C.

Uma vez visto os conceitos, na seção III expõe-se os Dispositivos utilizados seção III-A, e os principais algoritmos e protocolos na seção III-B. Em seguida apresenta-se uma Proposta de Arquitetura, e em seguida sua implementação na seção V. Por fim é apresentado os resultados obtidos e conclusões nas seções VI e VII respectivamente.

II. Definições

A. Redes LPWAN

LPWAN, *Low Power Wide Area Network*, são redes que alcançam longas distâncias gastando pouca energia, normalmente utilizadas para enviar poucos dados [2]. Dentre as tecnologias mais utilizadas estão SigFox e LoRa.

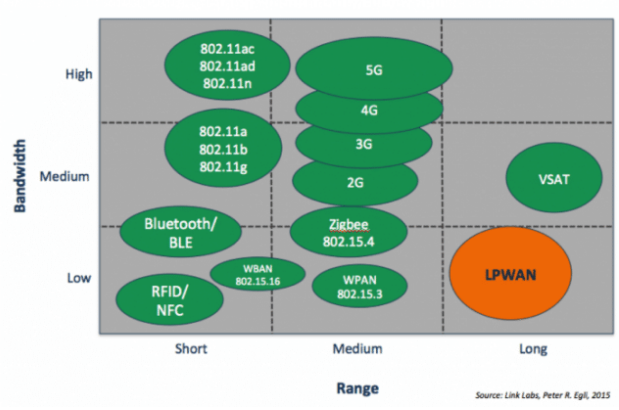


Fig. 1. Comparando Largura de Banda e distância das redes [2]

B. LoRa

LoRa, *Long Range*, é uma tecnologia que atua na camada física do modelo OSI para o envio e recebimento de dados. Criado e mantido de forma proprietária pela empresa Semtech [1] o mesmo utiliza comunicação através de ondas na frequência de radio, para codificar o envio de dados focando em abarcar longas distância a um baixo custo energético. Nesse sentido, uma rede LoRa é formada por diversas antenas [3], que se comunicam utilizando a tecnologia de *Chirp Spread Spectrum*

[4] uma tecnologia de comunicação militar adaptada para uso comercial de baixo custo [3] tais antenas possuem alcance de até 15 km em áreas rurais [5], bem como uma taxa média de duração de bateria de 10 anos [5].

C. IoT

O termo *Internet of Things (IOT)*, em português internet das coisas foi elaborado pelo britânico, Kevin Ashton, em 1999 [6] e se refere de forma geral a uma rede que conecta diversas "coisas" a internet, através de software, com o objetivo de trocar informações [7], tais "coisas" podem ser sensores, microcontroladores ou até mesmo objetos que nunca imaginamos tais como geladeiras, televisores, entre outros.

A *internet of things* é baseada em três camadas [6]:

- **Camada de percepção**: É uma camada que envolve sensores, hardware e obtém-se dados relevantes a respeito dos fenômenos meteorológicos, biológicos ou físicos tais como temperatura e umidade do solo [8], do ar [9], índice de área foliar [10], quantidade de gás SO_2 [11], PH do solo [11] entre muitos outros.
- **Camada de comunicação**: Responsável por enviar os dados coletados pela camada de percepção supracitada para outras camadas, quer sejam aplicações que vão analisar tais dados ou para grandes bancos de dados ou até mesmo para serviços na nuvem.
- **Camada de aplicação**: camada a qual trás sentido aos dados coletados pelos sensores, pois é nesse momento que ocorre o processamento dos dados e a apresentação dos mesmos. Nos trabalhos lidos durante a produção desta revisão bibliográfica, essa camada será responsável principalmente por mostrar ao agricultor informações relevantes de forma simples e compreensível, bem como informá-lo qual o melhor momento para plantar [12], ou em quais lugares da plantação tem doenças [13].

III. Materiais e métodos

A. Dispositivos

(a) ESP32 (b) LoRaMESH (c) ESP32-CAM

1) LoRaMESH EndDivice: Para acessar a tecnologia de camada física LoRa utilizou o Módulo LoRaMESH homologado pela ANATEL [14], o qual possui diversos parâmetros variáveis tais como: *Spreading Factor*, *Coding rate*, Largura de banda de até 500 kHz.

A configuração e utilização do dispositivo é feita através de duas interfaces de comunicação serial (UART), sendo a primeira utilizada para enviar e receber comandos de configuração e a segunda, chamada de interface transparente, envia e recebe diretamente da antena.

2) ESP32: Para criação do protótipo do dispositivo que recebe os dados foi utilizado um doit esp32 devkit v1[1] porém o código e a implementação devem funcionar de maneira idêntica em qualquer ESP32.

3) ESP32-CAM: De maneira similar para o dispositivo que envia dados foi utilizado um esp32-cam Ai-Thinker[2] porém no caso do esp32-cam para utilizar a câmera deve-se especificar quais portas estão conectadas, sendo assim para utilizar outro modelo deve definir os pinos no arquivo hardware.h para utilização do código implementado.

B. Protocolos

Dados os requisitos do projeto a escolha dos algoritmos próprios para a tarefa a ser desempenhada se torna de extrema relevância. Nesse sentido, são abordados os diferentes algoritmos utilizados na implementação do sistema, dando especial atenção para as caraterísticas que influenciaram na sua escolha.

1) Detecção de erros CRC: Visando a detecção de erros um dos algoritmos mais aconselháveis no caso de comunicação constante é o CRC, devido ao baixo custo computacional de seu cálculo para mensagens pequenas.

O funcionamento do algoritmo, em alto nível, é que o dispositivo transmissor gere um código para ser concatenado com a mensagem, uma vez que o receptor receba a mensagem o código é recalculado e comparado com o código recebido, caso sejam iguais a mensagem é considerada válida, caso contrário é considerada inválida e houve inversões de bits durante a transmissão.

O Algoritmo CRC pode variar em relação: como iniciar seu valor, qual o polinômio gerador e quantos bytes gera na saída.

Para este trabalho o CRC foi implementado em C++ utilizando 16 bits, tendo como valor inicial $C181_{16}$ e polinômio gerador $G(X) = A001_{16}$ seguindo as recomendações dos criadores do chip [14].

[1]Documentação da placa: https://olddocs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html

[2]Documentação da placa: https://docs.ai-thinker.com/en/esp32-cam

```c
static const uint16_t POLY = 0xA001;
static const uint16_t INIT = 0xC181;
uint16_t computeCRC(uint8_t* data_in, uint16_t length) {
    uint16_t i, crc_calc = INIT;
    uint8_t bitbang, j;
    for(i = 0; i < length; i++) {
        crc_calc ^= (((uint16_t)data_in[i]) & 0x00FF);
        for(j = 0; j < 8; j++) {
            bitbang = crc_calc;
            crc_calc >>= 1;
            if(bitbang & 1) {
                crc_calc ^= POLY;
            }
        }
    }
    return (crc_calc & 0xFFFF);
}
```

Fig. 2. Implementação do CRC

2) Controle de fluxo Stop and Wait: Pelo fato dos dispositivos LoRa possuírem canais *half duplex* (permitem envio nos 2 sentidos porém apenas um sentido por vez) inviabiliza algoritmos de janela deslizante.

Sendo assim do ponto de vista da camada de enlace de dados o algoritmo de controle de fluxo utilizado foi o *Stop and Wait*, o qual se carateriza pela sua simplicidade tanto durante a sua implementação como no seu funcionamento.

Durante a execução do protocolo o transmissor envia seu primeiro quadro e espera que o receptor envie a resposta "ACK" do mesmo. Caso o receptor não envie sua resposta em um tempo predeterminado o transmissor reenvia seu quadro. Uma vez recebido o "ACK" o transmissor pode deletar o quadro antigo e enviar o próximo, seguindo assim o seguinte fluxo:

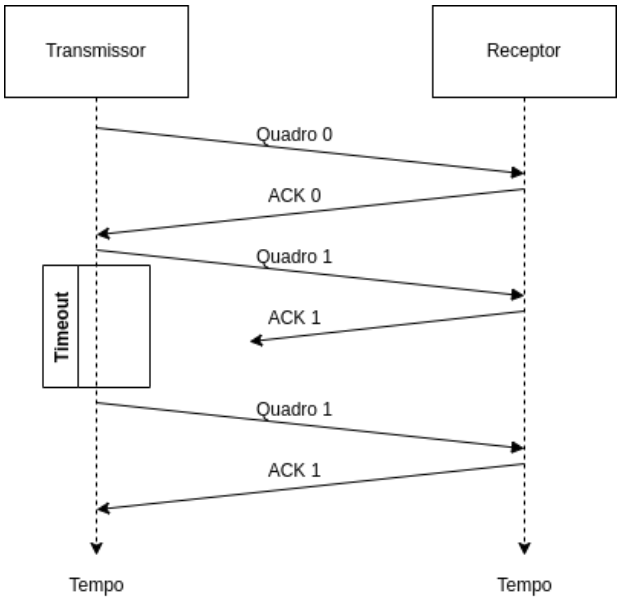


Fig. 3. Fluxo de execução no tempo do algoritmo stop and wait

IV. Proposta de Arquitetura

Após decidir os algoritmos e dispositivos mais adequados para implementação foi idealizada a arquitetura do projeto.

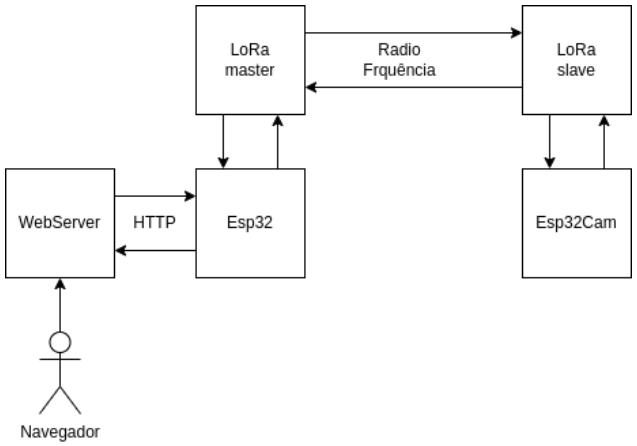


Fig. 4. Proposta de arquitetura para implementação

O foco principal da arquitetura é implementar uma rede LoRa estrela a qual possui um "mestre", chip com id 0, e diversos *slaves*, chips com id maior que 0. O chip *master* está conectado a um esp32 executando o algoritmo de receptor que será explicado na seção Implementação

Para teste foi criado 2 versões do código de dispositivos emissores, sendo eles:

1) Sempre envia, quando recebe o ack do último frame tira uma nova foto e envia a mesma.
2) Envia apenas quando solicitado, fica esperando até receber uma mensagem do *master* dizendo para tirar uma nova foto e enviar.

Para testar o recebimento e visualizar de maneira mais simples foi criado no dispositivo receptor um ponto de acesso WiFi na qual um usuário final pode se conectar informando o nome e senha da rede, também é disponibilizado um servidor web os seguintes *endpoints*:

- **/lora_img**
  - Objetivo: Mostrar no dispositivo do usuário uma imagem JPEG com a última foto salva no dispositivo;
  - Método: GET;
  - Retorna: image/jpeg
- **/req_img/{id}**
  - Objetivo: Fazer com que o LoRa mestre envie uma mensagem pedindo ao dispositivo que tire uma foto e envie;
  - Parâmetros: o id do chip LoRa que vai enviar a imagem;

– Método: GET;
– Retorna: text/plain, indicando se foi possível ou não fazer a requisição.

V. Implementação

Visto a proposta de arquitetura supracitada, faz-se necessário implementar dois projetos sendo eles: um transmissor, *Sender*, que envia uma imagem separada em diversos quadros utilizando o algoritmo *stop and wait*, e um receptor, *Receiver* que recebe e serve essas imagens para outras aplicações.

A. Sender

Do lado do *Sender*, dispositivo que captura e envia imagens, em um primeiro momento são inicializadas a câmera e a antena LoRa. Em seguida é feita a captura de uma imagem que passa a ser dividida em partes atendendo ao tamanho máximo do *payload* da mensagem que pode ser enviada pela antena LoRa. Assim, uma vez dividida a imagem inicia o seu processo de transmissão, no qual se envia o quadro e espera uma resposta. Caso não receba nada em determinado tempo ou receba algo que não seja o "ACK" esperado a mensagem será reenviada.

Sendo assim tem-se o seguinte fluxo de execução:

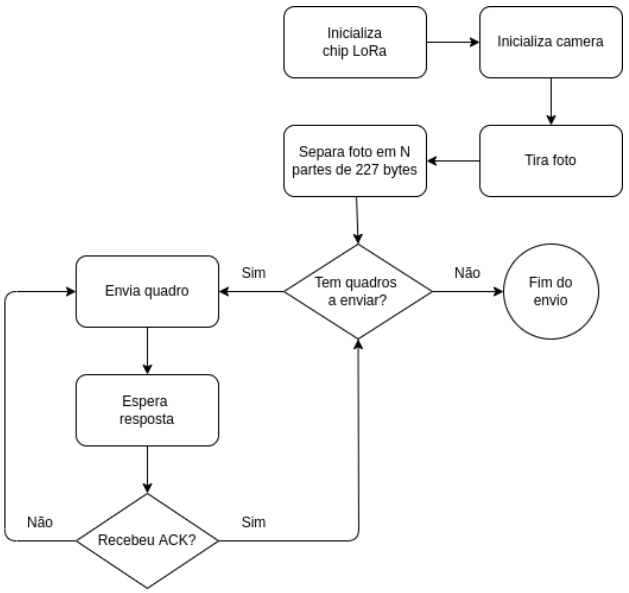


Fig. 5. Fluxograma geral do algoritmo do sender.

B. Receiver

Por outro lado no *receiver*, em um primeiro momento também são inicializadas as estruturas de controle, bem como é alocado um espaço que serve de buffer para colocar as partes da imagem que está sendo recebida. Após isso, o dispositivo fica esperando o recebimento de pacotes.

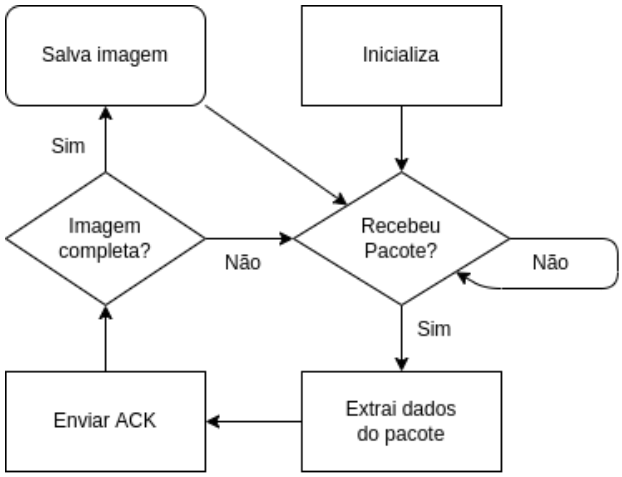
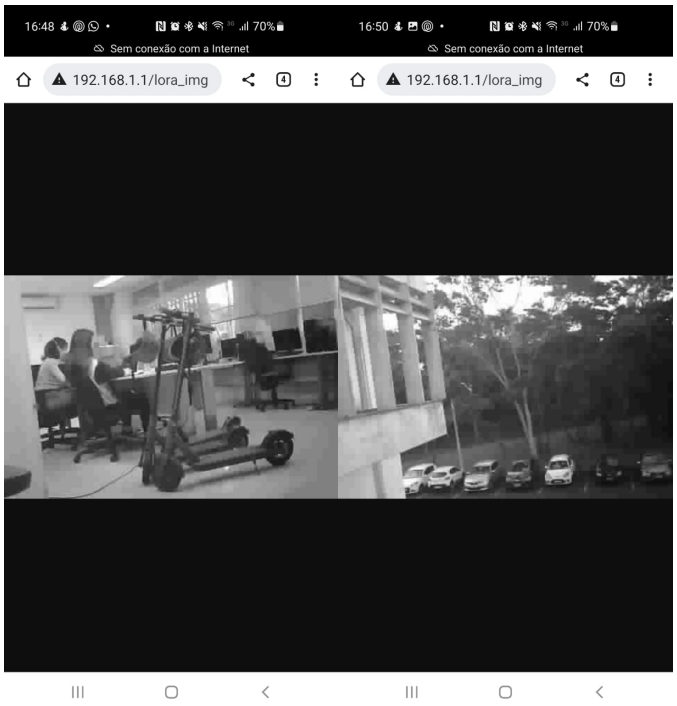


Fig. 6. Fluxograma geral do algoritmo do receiver.

Uma vez que a imagem foi salva, fica disponível ao usuário utilizando o navegador em seu computador ou celular chamar o *endpoint* /lora_img explicado na Seção IV, Proposta de arquitetura, obtendo imagens como:



(a) Imagem do laboratório (b) Imagem do estacionamento

Fig. 7. Imagens capturadas com esp32-cam mostradas no navegador

C. Formato das mensagens

Para enviar mensagens utilizando este chip LoRa, é preciso seguir o formato:

TABELA I
ESTRUTURA PADRÃO DE MENSAGENS LORA.

ID	Command	Payload	CRC
2 bytes	1 byte	1–231 bytes	2 bytes

Sendo assim sempre os primeiros 2 bytes de uma mensagem são o id do chip que está enviando, o terceiro byte é o comando, podendo ser para realizar alguma configuração no chip, ler algum dado específico, ou simplesmente para enviar para antena, seguidos de 1 a 231 bytes de mensagem e por fim os últimos 2 bytes são o CRC calculado para a mensagem específica.

Tendo uma quantidade máxima de 231 bytes para escrever os dados da imagem, optou-se pelos seguintes campos de cabeçalho:

TABELA II
ESTRUTURA DEFINIDA PARA O PAYLOAD DA MENSAGEM.

Payload				
Type	ID	Part	Total	Message
1 byte	1 byte	1 byte	1 byte	1–227 bytes

Sendo que:

- **Type**: indica como os próximos bytes devem ser interpretados, podendo ser um ACK, ou uma parte de imagem.
- **ID**: identificador único da imagem;
- **Part**: qual a parte da imagem que está nessa mensagem;
- **Total**: a quantidade total de partes da imagem;
- **Message**: os bytes da imagem

Esses campos são utilizados apenas se a mensagem a ser enviada for do tipo imagem, ou seja o 1 byte, campo *Type* for igual ao valor definido em código. Caso contrário, a mensagem será interpretada como um ACK, e terá apenas mais uma informação em seguida um outro byte dizendo qual parte de imagem está atrelado o "ACK", sendo esse uma cópia do campo *Part* da mensagem de imagem.

Dessa forma foi implementado a seguinte estrutura de dados em C++ para armazenar as partes da imagem e seus respectivos cabeçalhos:

```
23  struct _payload {
24      uint8_t byte_array[APPLICATION_MAX_PAYLOAD_SIZE];
25      uint8_t size;
26  };
27
28  struct _fields {
29      uint8_t type;
30      uint8_t id;
31      uint8_t part;
32      uint8_t last_part;
33  };
34
35  union ImagePart {
36      _fields fields;
37      _payload payload;
38  };
```

Fig. 8. Estruturas de dados utilizadas para armazenar as partes da imagem

VI. RESULTADOS

Tendo em vista a necessidade de testes, realizou-se o protótipo tanto do dispositivo transmissor, quanto do receptor conforme as especificações expostas nas seções IV e V.

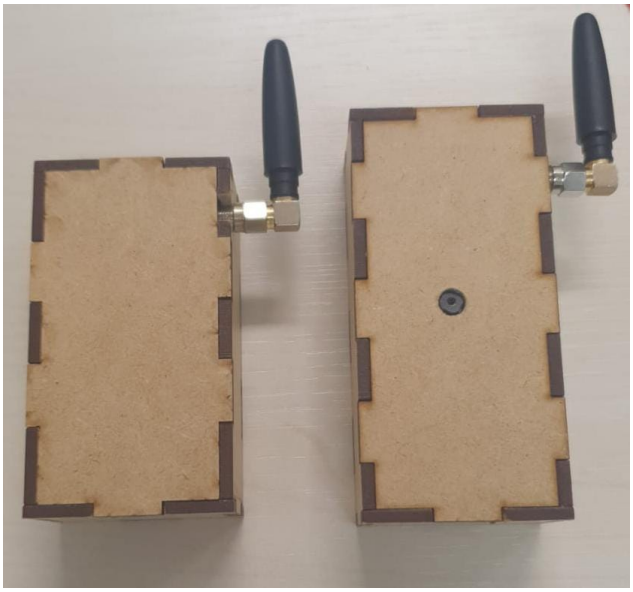


Fig. 9. Protótipos dos dispositivos LoRa

A fim de avaliar o funcionamento da implementação foram realizados diversos testes, dando especial atenção para a forma como a resolução e a taxa de compressão da imagem influenciam no tamanho da mensagem enviada no payload. Os principais resultados são mostrados a continuação.

No primeiro grupo de testes realizado o foco foi avaliar como a resolução da imagem influencia no tamanho em bytes da mensagem enviada. Para isso seguiu-se os seguintes critérios:

- 3 fotos por resolução escolhendo sempre a mediana.
- Fotos tiradas do mesmo local na mesma posição;
- Taxa de compressão do JPEG em 0;
- Imagens em escala de cinza;

TABELA III
MUDANÇA DE RESOLUÇÃO AFETANDO O TAMANHO DA IMAGEM

Resolução (pixels)	Tamanho (bytes)
640×480	73260
480×320	39139
400×296	35916
320×240	23510
240×176	14242
176×144	9147

No segundo grupo de testes o foco foi avaliar a taxa de compressão do JPEG, sendo assim seguiu-se os critérios supracitados porém mantendo a resolução fixa em 480x320 e alterando a perda de qualidade da imagem.

TABELA IV
MUDANÇA DE QUALIDADE DA IMAGEM AFETANDO O TAMANHO

Qualidade (0–63)	Tamanho (bytes)
0	39139
10	8456
20	6371
30	5613
40	5161
50	4842
60	4665
63	4616

Avaliando também a taxa de transmissão percebeu-se que utilizando o algoritmo *stop and wait* cada troca de quadros demora aproximadamente 2 segundos sendo assim pode-se calcular o tempo médio de envio de cada imagem dependendo unicamente da quantidade de quadros que a mesma ocupa:

$$tempo = 2 \cdot quadros$$

Sendo o tempo dado em segundos, e a quantidade de quadros podendo ser calculada:

$$quadros = \frac{imagem}{227}$$

Sendo imagem o tamanho da imagem gerada em bytes.

Dessa maneira se for gerada uma imagem de 6810 bytes podemos calcular o tempo de envio da mesma:

$$tempo = 2 \cdot \frac{imagem}{227}$$
$$tempo = 2 \cdot \frac{6810}{227}$$
$$tempo = 2 \cdot 30$$
$$tempo = 60$$

Consequentemente o tempo de envio de uma imagem de 6810 bytes é de 60 segundos, 1 minuto.

VII. CONCLUSÃO

Apesar de todas as limitações da tecnologia LoRa tais como velocidade de transmissão, direção do canal, ainda é possível transmitir pequenas e comprimidas imagens com alta latência.

Sendo assim possível implementar uma rede para transmissão de conteúdos maiores que o suportado por uma única mensagem LoRa (231 bytes), abrindo possibilidades para envio de imagens, arquivos e grandes mensagens.

Para trabalhos futuros pretende-se implementar uma biblioteca mais genérica que disponibilize uma interface para outros chips LoRa, também testar envio de arquivos binários tanto de código quanto de configuração.

AGRADECIMENTOS

Primeiramente agradeço a Universidade Estadual do Oeste do Paraná por proporcionar as bases teóricas em redes de computadores necessárias para realização deste trabalho.

Agradeço ao Laboratório de Internet of Things e ao Parque tecnológico de Itaipu por fornecer equipamentos e dispositivos tanto para elaboração de protótipos quanto para testes.

REFERÊNCIAS

[1] M. Bor, J. E. Vidler, and U. Roedig, "Lora for the internet of things," 2016.
[2] P. Melo, "Introdução ao LPWAN (Low Power Wide Area Network)," Jan. 2017. [Online]. Available: https://embarcados.com.br/introducao-ao-lpwan/
[3] Lora-Alliance, "Lorawan what is it?" 2015. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf
[4] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski, and N. Koteli, "Iot agriculture system based on lorawan," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–4.
[5] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications magazine*, vol. 55, no. 9, pp. 34–40, 2017.
[6] A. Tzounis, N. Katsoulas, T. Bartzanas, and C. Kittas, "Internet of things in agriculture, recent advances and future challenges," *Biosystems engineering*, vol. 164, pp. 31–48, 2017.
[7] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6851114

[8] F. Kizito, C. S. Campbell, G. S. Campbell, D. R. Cobos, B. L. Teare, B. Carter, and J. W. Hopmans, "Frequency, electrical conductivity and temperature analysis of a low-cost capacitance soil moisture sensor," *Journal of Hydrology*, vol. 352, no. 3-4, pp. 367–378, 2008.

[9] F. Mesas-Carrascosa, D. V. Santano, J. Meroño, M. S. De La Orden, and A. García-Ferrer, "Open source hardware to monitor environmental parameters in precision agriculture," *Biosystems engineering*, vol. 137, pp. 73–83, 2015.

[10] J. Bauer, B. Siegmann, T. Jarmer, and N. Aschenbruck, "On the potential of wireless sensor networks for the in-situ assessment of crop leaf area index," *Computers and Electronics in Agriculture*, vol. 128, pp. 149–159, 2016.

[11] N. Karimi, A. Arabhosseini, M. Karimi, and M. H. Kianmehr, "Web-based monitoring system using wireless sensor networks for traditional vineyards and grape drying buildings," *Computers and Electronics in Agriculture*, vol. 144, pp. 269–283, 2018.

[12] J. Kath and K. G. Pembleton, "A soil temperature decision support tool for agronomic research and management under climate variability: adapting to earlier and more variable planting conditions," *Computers and Electronics in Agriculture*, vol. 162, pp. 783–792, 2019.

[13] S. Trilles Oliver, J. Torres-Sospedra, O. Belmonte, F. Zarazaga-Soria, A. González Pérez, and J. Huerta, "Development of an open sensorized platform in a smart agriculture context: a vineyard support system for monitoring mildew disease," 2019.

[14] RADIOENGE, "Módulo loramesh, manual de utilização," 2022. [Online]. Available: https://www.radioenge.com.br/storage/2021/08/LoRaMESH_RD42C_Manual.pdf