IoT: rede LoRa para envio de imagens

IoT: LoRa network for sending images

Victor E. Almeida Marco A. Guerra

22 de julho de 2022

Resumo

Palavras-chave: LoRa, Rede LPWAN, envio de imagens.

Abstract

Escrever resumo

Keywords: LoRa, LPWAN Network, sending images.

1 Introdução

O barateamento dos dispositivos de Internet of Thing tem aberto novas possíbilidaes, ainda com a adquisição de novas capacidades com um menor consumo de energia e maior eficiência tem se torndo possível a implementação de sistemas embarcados cada vez mais específicos a um baixo custo.

Nesse sentido, uma das carateristicas desejáveis nos dispositivos e o envio de dados a longa distãncia, para

O seguinte trabalho apresenta as princiapis escolhas e os resultados obtidos na implementação do uma rede baseada em LoRa para o envio de imagens entre dispositivos

2 Definições

Antes de abordar as principais tecnologias, bem como os principais algoritmos utilizados durante o projeto, faz-se necessário elucidar alguns pontos.

2.1 LoRa

Os dispositivos LoRa são uma implementação da camada física do modelo OSI para o envio e recevimento de dados. Criado e mantido de forma proprietária pela empresa Semtech o mesmo utiliza comunicação atraves de ondas na frequência de radio, para codificar o envio de dados focando em abarcar longas distãncia a um baixo custo energetico. 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].

2.2 IoT

O termo *Internet of Things (IOT)*, em português internet das coisas foi elaborado pelo britânico, Kevin Ashton, em 1999[1] e se refere de forma geral a uma rede que conecta diversas "coisas" a internet, através de software, com o objetivo de trocar informações[6], tais "coisas" podem ser sensores, microcontroladores ou até mesmo objetos que nunca imaginamos tais como geladeiras, televisores, entre outros.

A estrutura do *internet of things* é baseada em três camadas principais[1]:

- **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[7], do ar[8], índice de área foliar[9], quantidade de gás SO_2[10], PH do solo[10] 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[11], ou em quais lugares da plantação tem doenças[12].

Ainda pode ser citada uma camada intermediária, sendo ela chamada de ***Middleware*** a qual facilita a interação da camada de aplicação com a de percepção[1], fornecendo ferramentas que permitem abstrações em relação a camada de percepção e possibilitando a integração de softwares recentes com o legado[1]. Um exemplo é na pesquisa de [2] essa integração foi feita a partir de um *Web processing Service* que padroniza as entradas e saídas e também por diversos serviços para refinar os dados obtidos dos sensores, fazendo assim o pré-processamento das medições, explicado com muitos detalhes na seção 2.2 de seu trabalho.

3 Materiais e métodos

3.1 Protocolos

Dados os requisitos do projeto a escolha dos algoritmos proprios 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.

3.1.1 Detecção de erros: CRC

Visando a detecção de erros um dos algoritmos

Falar dos algoritmos de Detecção de erro (CRC) e correção de erros (Hamming)

Algoritmo CRC 16 bits

```
1  static const uint16_t POLY = 0xA001;
2  static const uint16_t INIT = 0xC181;
3
4  uint16_t computeCRC(uint8_t* data_in, uint16_t length) {
5      uint16_t i;
6      uint8_t bitbang, j;
7      uint16_t crc_calc = INIT;
8
9      for(i = 0; i < length; i++) {
10         crc_calc ^= (((uint16_t)data_in[i]) & 0x00FF);
11         for(j = 0; j < 8; j++) {
12             bitbang = crc_calc;
13             crc_calc >>= 1;
14
15             if(bitbang & 1) {
16                 crc_calc ^= POLY;
17             }
18         }
19     }
20     return (crc_calc & 0xFFFF);
21 }
```

3.1.2 Stop and Wait

Do ponto de vista da camada de enlace de dados o algoritmo utilizado foi o **Stop and Wait**, o qual se carateriza pela sua simplicidade tanto durante a sua implementação como no seu funcionamento. A continuação é descrito o funcionamento dos dispositivos envolvidos no envio e recebimento de imagens.

Do lado do Sender, dispositivo que captura e envia imagens, em um primeiro momento são inicializadas a câmara e a antena LoRa. A continuação é feita a captura de uma imaagen que passa a ser dividida em partes atendendo ao tamanho maximo que pode ter o payload da mensagem que pode ser enviada pela antena LoRa. Assim, uma vez particionada a imagem inicia o seu processo de envio o qual continua se

Algoritmo Stop and Wait - Sender

```
void sender() {
    InitCamera();
    InitLoRa();
    TakePicture(image); // tira uma foto
    SplitImage(image); // separa image em pacotes
    while(image.sendedParts < image.totalParts) {
        SendImagePart(image.part()); // envia quadro contendo parte da imagem
        ReceivePacketCommand  (buffer); // espera ate receber um quadro
        if(buffer.messageType == ACK) {
            image.sendedParts++; // incrementa contador de imagens enviadas
        }
    }
}
```

Por outro lado no reciver, 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 recevida. Apos isso, o dispositivo fica esperando o recevimento de pacotes

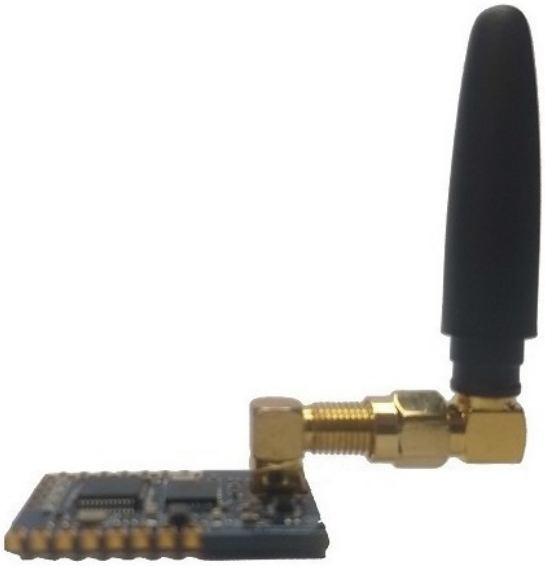
Algoritmo Stop and Wait - Reciver

```
void reciver() {
    InitImage(image); // inicializa estrutura de dados das imagens
    InitLoRa();
    while(true) {
        ReceivePacketCommand  (buffer); // espera ate receber um quadro
        SaveImageBytes(buffer); // salvar bytes na estrutura da imagem
        PrepareFrameCommand(); // prepara ACK
        SendPacket(); // envia ACK
        if(image.isComplete()) {
            SaveImage(image); // salvar imagem em arquivo
        }
    }
}
```

3.2 Dispositivos

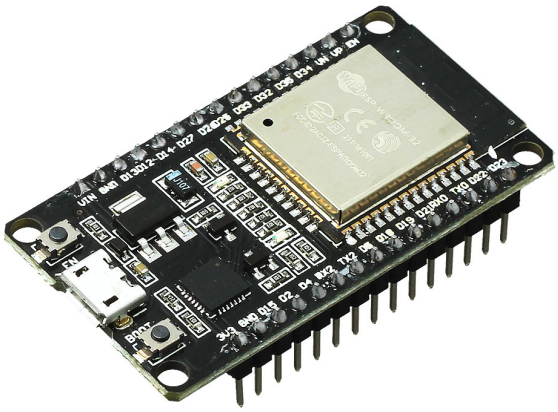
3.2.1 LoRaMESH EndDivice

Figura 1 – LoRaMESH EndDivice.



3.2.2 ESP32

Figura 2 – ESP32.



3.2.3 ESP32-CAM

O dispositivo ESP32-CAM é um controlador embarcado de baixocusto e de baixo consumo energetico

Figura 3 – ESP32-CAM.

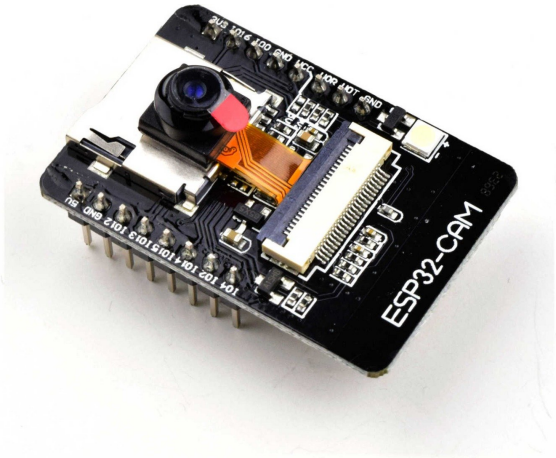


Tabela 1 – Estrutura padrão de mensagens LoRa.

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

Tabela 2 – Estrutura definida para o payload da mensagem.

Payload				
Type	ID	Part	Total	Message
1 byte	1 byte	1 byte	1 byte	1 - 227 byte

4 Implementação

Definição da estrutura do payload

```
1  struct _payload {
2      uint8_t byte_array[APPLICATION_MAX_PAYLOAD_SIZE];
3      uint8_t size;
4  };
5
6  struct _fields {
7      uint8_t type;
8      uint8_t id;
9      uint8_t part;
10     uint8_t last_part;
11 };
12
13 union ImagePart {
14     _fields fields;
15     _payload payload;
16 };
```

4.1 Sender

4.2 Reciver

Figura 4 – Fluxograma geral do algoritmo do sender.

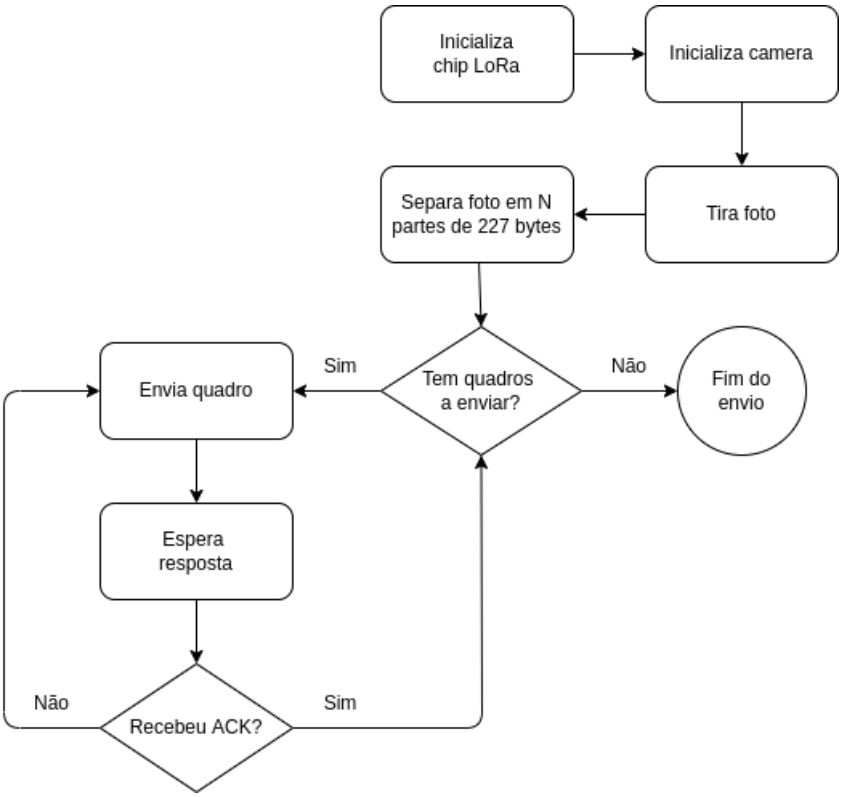
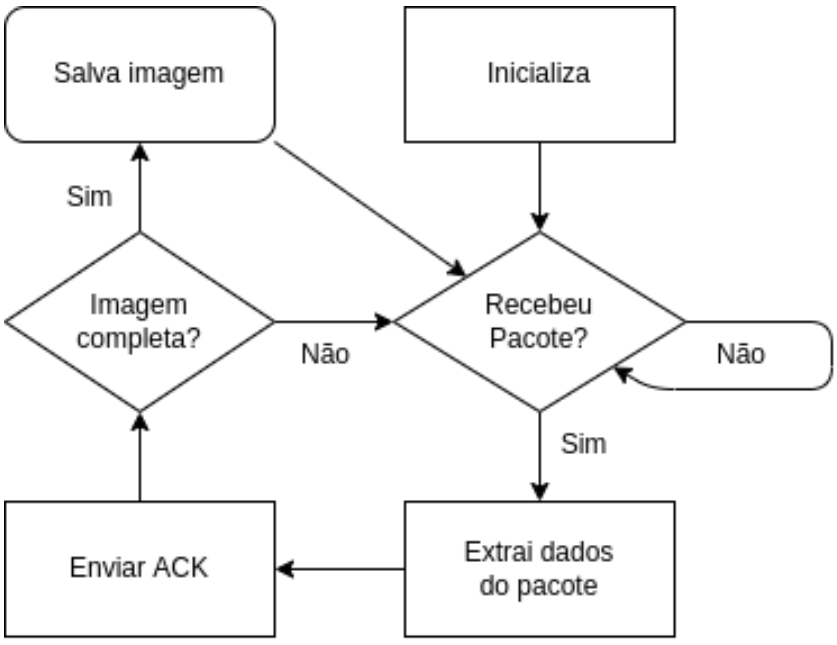


Figura 5 – Fluxograma geral do algoritmo do reciver.



Resolução (pixels)	Tamanho (bytes)
640x480	73260
480x320	39139
400x296	35916
320x240	23510
240x176	14242
176x144	9147

Tabela 3 – Mudança de resolução afetando o tamanho da imagem

5 Resultados

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,

6 Discussão

Referências

1 TZOUNIS, A. et al. Internet of things in agriculture, recent advances and future challenges. **Biosystems engineering**, Elsevier, v. 164, p. 31–48, 2017. Citado na página 2.

2 SAWANT, S.; DURBHA, S. S.; JAGARLAPUDI, A. Interoperable agro-meteorological observation and analysis platform for precision agriculture: A case study in citrus crop water requirement estimation. **Computers and electronics in agriculture**, Elsevier, v. 138, p. 175–187, 2017. Citado na página 2.

3 LORA-ALLIANCE. **LoRaWAN What is it?** 2015. Disponível em: <https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf>. Acesso em: 21 de fevereiro de 2021. Citado na página 2.

4 DAVCEV, D. et al. Iot agriculture system based on lorawan. In: IEEE. **2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)**. [S.l.], 2018. p. 1–4. Citado na página 2.

5 ADELANTADO, F. et al. Understanding the limits of lorawan. **IEEE Communications magazine**, IEEE, v. 55, n. 9, p. 34–40, 2017. Citado na página 2.

6 Chen, S. et al. A vision of iot: Applications, challenges, and opportunities with china perspective. **IEEE Internet of Things Journal**, v. 1, n. 4, p. 349–359, 2014. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6851114>. Acesso em: 16 de janeiro de 2021. Citado na página 2.

7 KIZITO, F. et al. Frequency, electrical conductivity and temperature analysis of a low-cost capacitance soil moisture sensor. **Journal of Hydrology**, Elsevier, v. 352, n. 3-4, p. 367–378, 2008. Citado na página 2.

8 MESAS-CARRASCOSA, F. et al. Open source hardware to monitor environmental parameters in precision agriculture. **Biosystems engineering**, Elsevier, v. 137, p. 73–83, 2015. Citado na página 2.

9 BAUER, J. et al. On the potential of wireless sensor networks for the in-situ assessment of crop leaf area index. **Computers and Electronics in Agriculture**, Elsevier, v. 128, p. 149–159, 2016. Citado na página 2.

10 KARIMI, N. et al. Web-based monitoring system using wireless sensor networks for traditional vineyards and grape drying buildings. **Computers and Electronics in Agriculture**, Elsevier, v. 144, p. 269–283, 2018. Citado na página 2.

11 KATH, J.; PEMBLETON, K. G. 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**, Elsevier, v. 162, p. 783–792, 2019. Citado na página 2.

12 OLIVER, S. T. et al. **Development of an open sensorized platform in a smart agriculture context: a vineyard support system for monitoring mildew disease**. 2019. Citado na página 2.