Projeto Final de TR2 - Monitoramento Remoto de Tanques de Combustível

Projeto de criação e implementação de um sistema de monitoramento remoto do nível de tanques de combustível utilizando comunicação LoRa e dispositivos Arduino para a disciplina de Teleinformática e Redes 2 na Universidade de Brasília (UnB) em 2024 (2024.1).

- Felipe Fontenele dos Santos - 190027622 - Oseias Romeiro Magalhães - 211036123 - Paulo Victor França de Souza - 200042548

> Repositório GitHub: oseias-romeiro/tr2_projeto.git

Demonstração:

Vídeo demonstrativo em ambiente de teste: youtu.be/Vv9kiqzvt7s

Setup

Para rodar o projeto, deve-se instalar os seguintes requisitos: - Arduino CLI `sh arduino-cli lib install "LoRa" arduino-cli lib install "Ultrasonic" arduino-cli lib install "Sleep_n0m1"` - Python `sh pip install pyserial pip install requests`

Executando

Utilizar os seguintes comandos no terminal:

```
  arduino-cli compile --fqbn arduino:avr:uno "%~dp0\Node\Node.ino"
  arduino-cli upload -p <porta_usb_arduino_node> --fqbn arduino:avr:uno
"%~dp0\Node\Node.ino"
  arduino-cli compile --fqbn arduino:avr:uno "%~dp0\Gateway\Gateway.ino"
  arduino-cli upload -p <porta_usb_arduino_gateway> --fqbn arduino:avr:uno
"%~dp0\Gateway\Gateway.ino"
  start cmd /k arduino-cli monitor -p "%~dp0\Node\Node.ino"
  python SerialListener.py
```

OU

- Windows: `Setup.bat`
- Linux: `chmod +x ./Setup.sh && ./Setup.sh`

Servidor Web

Executando servidor Web localmente com banco de dados SQLite:

```
cd ./web
pip install -r requirements.txt
```

```
flask db init
flask db migrate -m "init"
flask db upgrade
flask run
```

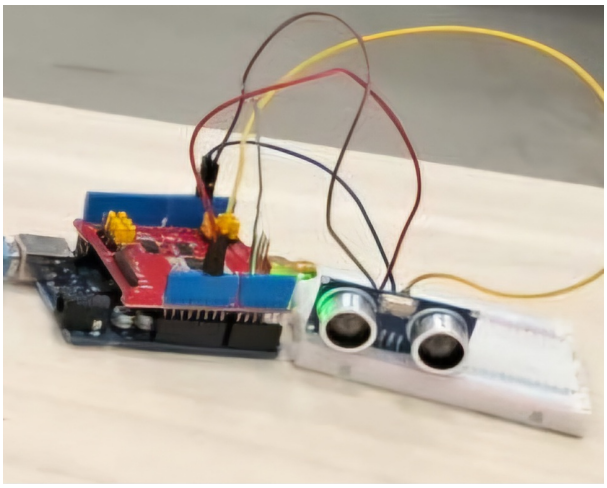
> Veja o serviço hospedado em tr2.always.net

Como o projeto funciona?

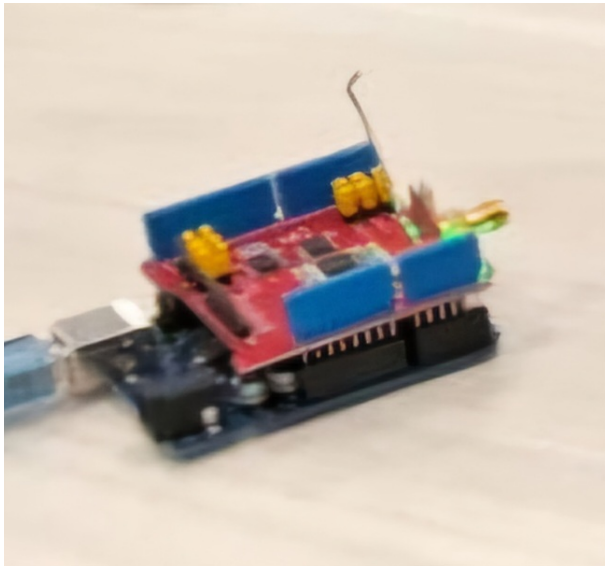
Cada tanque teria um Arduino com um sensor de ultrassom acoplado que mede o nível de combustível no tanque e envia o dado via **LoRa** (tecnologia de rádio para longo alcança com baixa energia) para um **receptor central**, que deve receber o dado e encaminhar para o um serviço web que cria um dashboard com base nos dados coletados.

Neste projeto, utilizaremos os termos *Node* para os Arduínos que coleta e envia os dados do sensor e *Gateway* para o Arduíno que faz a função de receptor central.

- ***Node***: Consiste em um Arduino Uno conectado a uma protoboard, que possui um sensor ultrassônico para medir o nível de combustível no tanque com base no tempo de reflexão das ondas, calculando o quão vazio está. Além disso, ele é conectado a um LoRa Shield para transmitir os dados coletados pelo sensor.



- ***Gateway***: Consiste em um Arduino Uno conectado a um LoRa Shield para recebimento dos dados enviados por algum *Node*.



Comunicação *Gateway* e *Nodes*

A comunicação se inicia com o *gateway* enviando um sinal de broadcast para os *nodes*. Após isso, ele aguarda 10 segundos o sinal de algum *node* enviando o seu ID. Essa tentativa de conexão é realizada 3 vezes, a fim de garantir que mesmo com algumas perdas de pacotes ou dessincronização (com um certo limite) a conexão seja concluída com sucesso. Caso contrário, infere-se que não há nenhum *node* acordado, portanto ele entra em modo de baixo consumo de energia por um determinado tempo. Caso receba sinal de algum *node*, o gateway confirma para assim estabelecer a conexão entre eles.

Então o *node* captura os dados do sensor, faz a medição em **centímetros**, envia o dado para o *gateway* e entra em estado de baixo consumo de energia por um determinado tempo.

O *gateway* recebe os dados do *node*, insere em formato JSON juntamente com logs das etapas anteriores e escreve no portal serial para ser lidas pelo programa SerialListner, depois formatada e enviada para o servidor web.

Serviço Web

O serviço tr2.alwaysdata.net recebe os dados de sensores recebidos por requisição post e salva em um banco de dados (PostgreSQL em produção de SQLite em desenvolvimento). A partir dos dados armazenados é gerado um **dashboard** para visualizar nível dos tanques com gráfico do histórico e uma **predição** de vida útil em horas de consumo que o tanque aguenta.

Endpoint	Método	Descrição
/	GET	Lista de tanques monitorados
/tanque/{id}	GET	Dashboard do tanque
/tanque/{id}	POST	Cadastro de tanque e adição de dados de sensores
/tanque/{id}/edit	GET	Tela para edição de informações opcionais do tanque
/tanque/{id}/edit	POST	Edição de informações opcionais do tanque
/logs	GET	Registro de comunicação entre os dispositivos
/logs/delete	POST	Apaga os logs existentes

Os logs são usados tanto para registro de comunicações entre os *nodes* e o *gateway* como também para estatísticas de controle de troca de pacotes na rede LoRa que conecta os dispositivos. Pode-se ver um exemplo de log nas imagens da Seção:Interface web do servidor no final deste arquivo. Porém, os exemplos não satisfazem todas as situações possíveis de rede coberto pelos logs.

Correções e Aprimoramentos

Apesar de não ter sido implementada, podemos aprimorar o projeto adicionando uma placa Wi-Fi ao *gateway* para ele conectar-se diretamente ao servidor web.

A nossa implementação quanto aos logs, foi bem aplicada no quesito de registrar a comunicação, porém melhorias podem ser feitas para trazer mais informações estatísticas sobre a condição da rede.

Outro ponto importante é a questão dos IDs que foram implementadas como constantes no código do *node*, porém, a ideia é não haver necessidade de programação de cada um. Deve ser atribuído automaticamente ao

conectar-se na rede, pois a princípio não se sabe a quantidade de *nodes* e deve poder ser inserido um a qualquer momento.

Além disso, um problema presente na implementação atual é a dificuldade do *gateway* de se comunicar com os *nodes* em um ambiente muito concorrido com diversos dispositivos, o que será abordado logo abaixo.

Solução para Problemas de Comunicação na Implementação

Quando o *gateway* tenta responder um *node*, a rede pode estar conturbada, pois vários *nodes* estão ao mesmo tempo, tentando enviar seu ID ao *gateway*. Neste momento nenhum nó consegue receber a resposta do *gateway* e assim nenhuma conexão será estabelecida, fazendo assim, com que todos os *nodes* fiquem esperando o *gateway*, enquanto para ele, não há nenhum *node* acordado.

Uma solução é impedir que todos os *nodes* tentem estabelecer conexão ao mesmo tempo, sendo necessário então o *gateway* dizer quando um *node* pode acordar e tentar comunicar-se, formando assim um unicast entre o *gateway* e o *node*. Para isso, o *gateway* deve ser capaz de: 1. Reconhecer cada *node* que é inserido na rede e o atribuindo um ID. 2. Designar momentos específicos para a comunicação de cada *node*. 3. Remover *nodes* inativos da rede do controle.

Dessa forma, a comunicação entre o gateway e os *nodes* será mais eficiente, reduzindo interferências e garantindo que as respostas sejam recebidas corretamente.

Conclusões

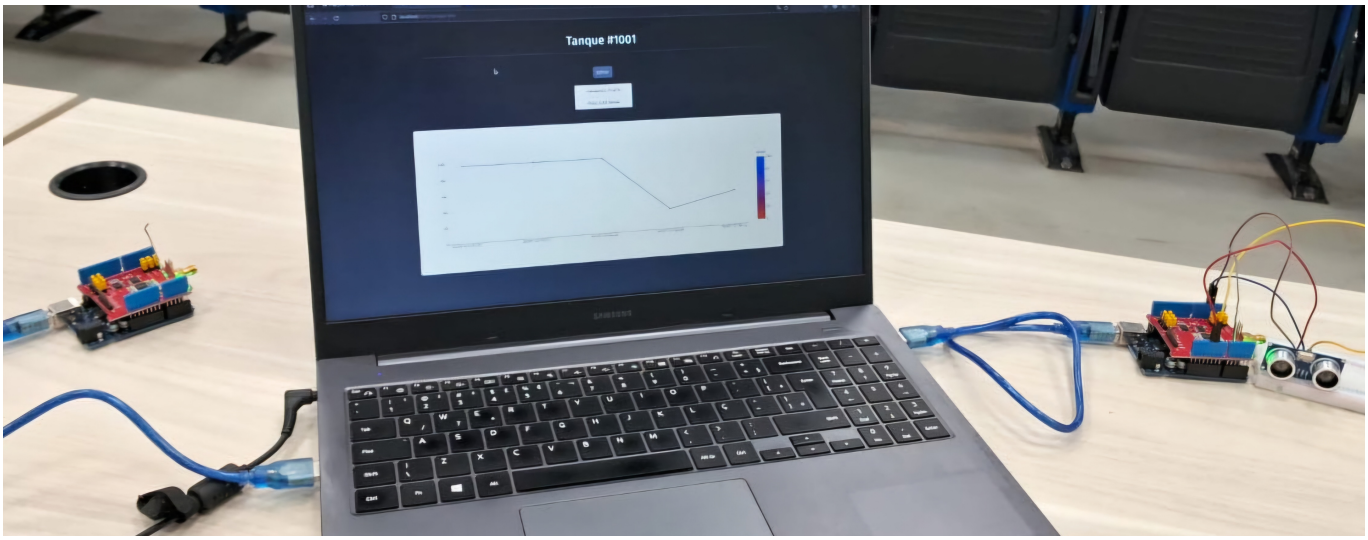
Apesar dos problemas abordados e das propostas de aprimoramentos em Correções e Aprimoramentos, o projeto é funcional como mostrado em vídeo com o ambiente de teste que tivemos acesso, aplicando assim, na prática, os conhecimentos da disciplina na implementação de uma solução para um problema real.

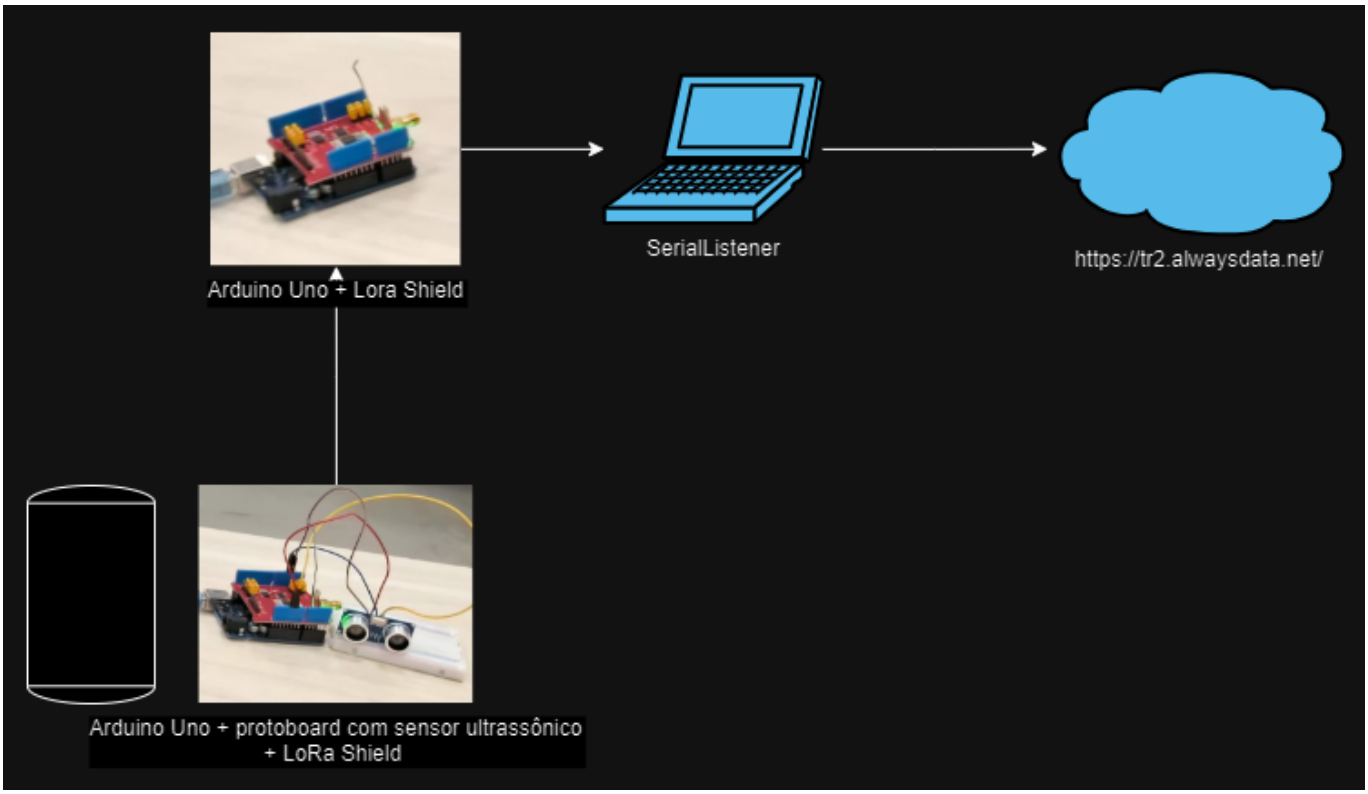
Referências

- Arduino LoRa
- Lora Shield
- Ultrasonic arduino
- Arduino CLI
- Flask

Imagens

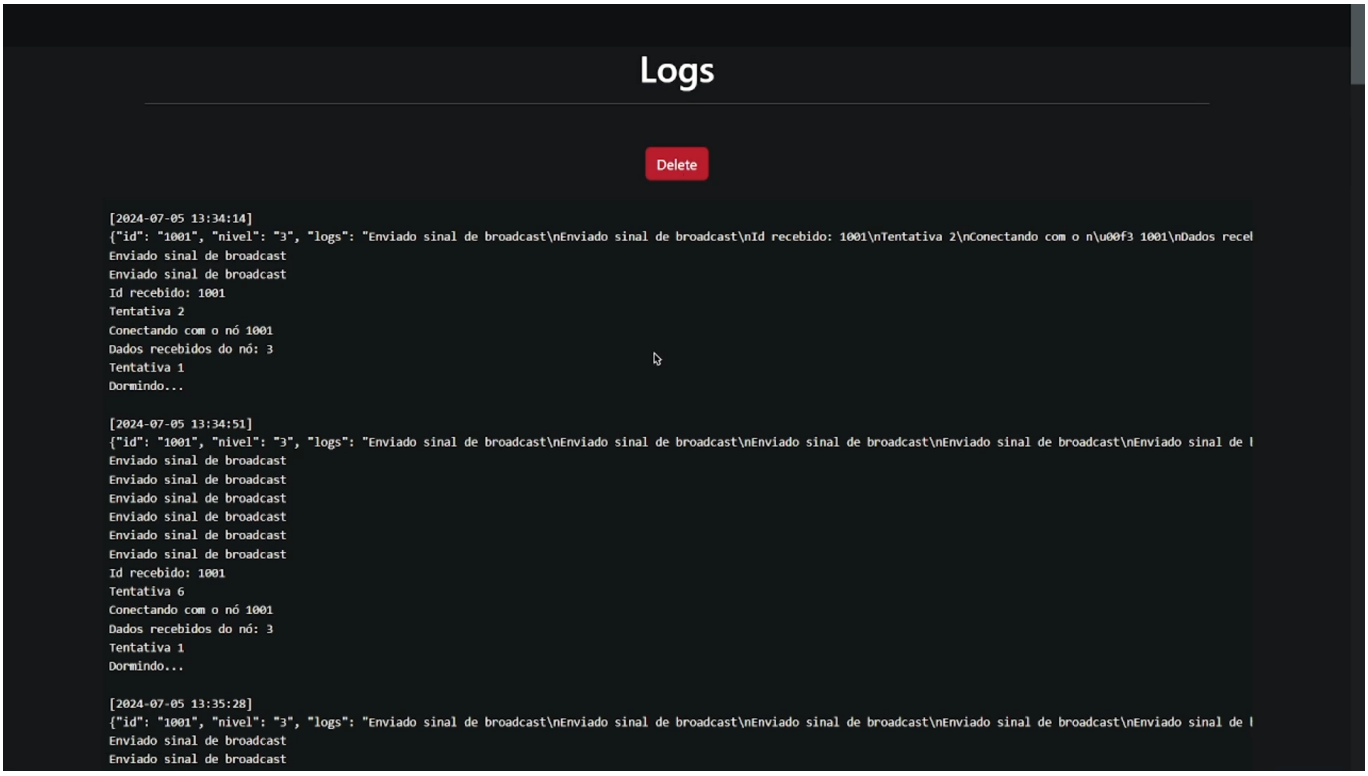
Gateway e *Node*

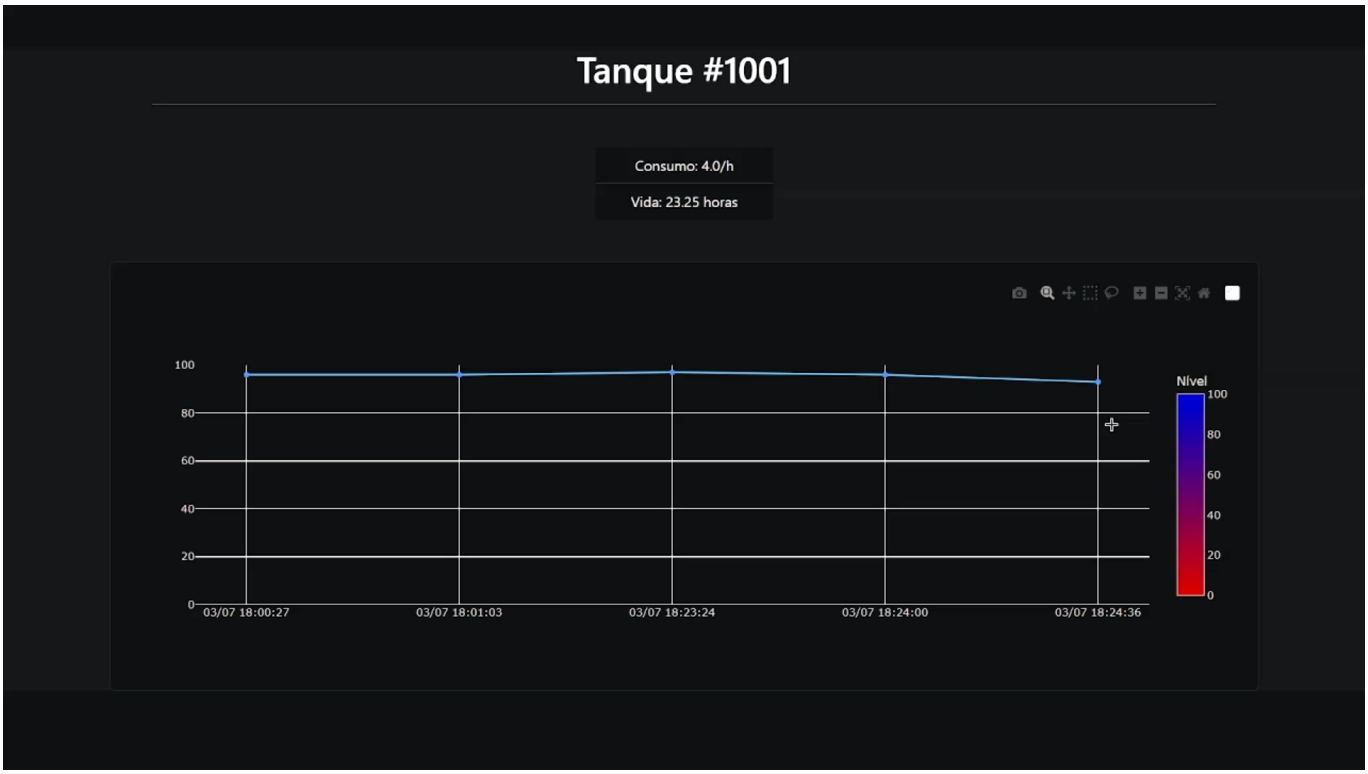




Interface web do servidor


Monitoramento de tanques de combustíveis
logs
Tanque #1001
Capacidade = 100.0
None
Tanque #1
Capacidade = 100.0
tanque de teste 1
Tanque #2
Capacidade = 100.0
Tanque de teste id=2


Logs
Delete
[2024-07-05 13:34:14]
{"id": "1001", "nivel": "3", "logs": "Enviado sinal de broadcast\nEnviado sinal de broadcast\nId recebido: 1001\nTentativa 2\nConectando com o n\u00f3 1001\nDados recel
Enviado sinal de broadcast
Enviado sinal de broadcast
Id recebido: 1001
Tentativa 2
Conectando com o nó 1001
Dados recebidos do nó: 3
Tentativa 1
Dormindo...
[2024-07-05 13:34:51]
{"id": "1001", "nivel": "3", "logs": "Enviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de I
Enviado sinal de broadcast
Enviado sinal de broadcast
Enviado sinal de broadcast
Enviado sinal de broadcast
Enviado sinal de broadcast
Enviado sinal de broadcast
Id recebido: 1001
Tentativa 6
Conectando com o nó 1001
Dados recebidos do nó: 3
Tentativa 1
Dormindo...
[2024-07-05 13:35:28]
{"id": "1001", "nivel": "3", "logs": "Enviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de broadcast\nEnviado sinal de I
Enviado sinal de broadcast
Enviado sinal de broadcast


Tanque #1001
Consumo: 4.0/h
Vida: 23.25 horas
100
80
60
40
20
0
03/07 18:00:27
03/07 18:01:03
03/07 18:23:24
03/07 18:24:00
03/07 18:24:36
Nivel
100
80
60
40
20
0