Projeto Final
Monitoramento Remoto de Tanques de Combustível

Andre de Oliveira Melo - 222008252
Gustavo Santana Lima - 211038235

1 Introdução

1.1 Objetivo

O objetivo do projeto é desenvolver um sistema de monitoramento remoto de tanques de combustível utilizando microcontroladores associados à redes LoRa. O sistema deve ser responsável por monitorar os níveis de combustível dos tanques de cada gerador, utilizando do LoRa - e seus mecanismos de baixa energia - para transmitir os dados entre Arduinos e depois mandar essas informações para a nuvem.

1.2 Motivação

Atualmente os geradores de energia da Universidade de Brasília não possuem monitoramento automatizado do nível de combustível dos seus tanques. Dessa forma, é necessário que periódicamente uma pessoa tenha que verificar presencial e manualmente cada tanque. Automatizar essa tarefa otimizaria a verificação dos níveis além de possibilitar alertas em caso de baixo nível de combustível, evitando assim um possível mau funcionamento de algum gerador em momentos de necessidade.

2 Descrição do Sistema

2.1 Visão Geral da Arquitetura

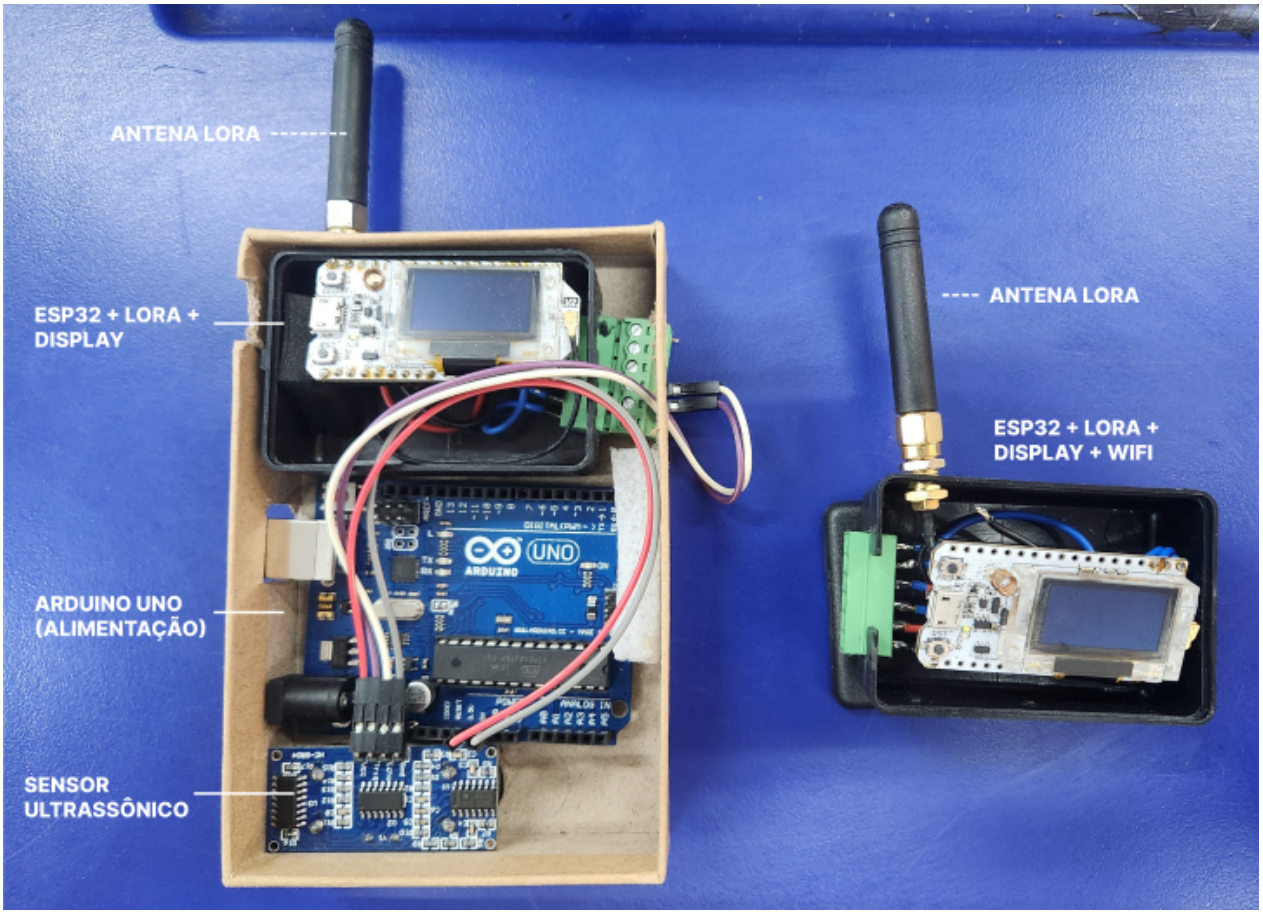
Há um número fixo de tanques de combustível espalhados no Campus Darcy Ribeiro, portanto em cada tanque teríamos um módulo LoRa e um ultrassom acoplados à um microcontrolador que comunicaria com um microcontrolador central que deve ser capaz de se conectar a internet e enviar as informações para um servidor web.

3 Especificações Técnicas

3.1 Lista de Componentes Utilizados

- 2 módulos Heltec WiFi LoRa 32(V2) (ESP32 com display e chip LoRa embutidos);
- 1 módulo HC-SR04 (Sensor de distância ultrassônico);
- 1 Arduino Uno (Utilizado somente para a alimentação do sensor);

3.2 Esquemas de conexão e diagrama



3.3 Capacidades físicas

O módulo de Ultrassom é capaz de medir em especificação de hardware de 2 à 400cm, para os fins utilizados ele foi configurado para ler de 2 à 200cm. Além disso, o módulo Heltec é - segundo às especificações - capaz de se comunicar em até 3,6 km sem obstruções. Em nossos testes conseguimos nos comunicar em até aproximadamente 850m com obstruções.

3.4 Servidor Utilizado

Foi utilizada a plataforma Adafruit IO como mecanismo de servidor (com banco de dados e dashboard).

4 Implementação

4.1 Código Sender

O código do sender se baseia em 4 partes, sendo a primeira a utilização da biblioteca NewPing para a utilização do ultrassom e suas funções para fazer a leitura, descobrir a altura e baseado nisso, calcular a porcentagem de diesel que ainda possui no tanque. A segunda se baseia na transferência dos dados do ID do sender(usado para diferenciação do tanque) e o dado da porcentagem de quanto diesel ainda há no tanque. Já a terceira parte se baseia num timer, enquanto o sender espera o ACK do gateway e, caso não aconteça, haverá outra tentativa de envio do dado. Se caso receba o ACK com seu próprio ID, o sender entrará em modoo deep sleep, onde todas as funções do microcontrolador e periféricos ficarão inativas, exceto os timers internos, que começam uma contagem até um valor pré-fixado. E, após o timer chegar a tal valor, o sender será religado e ele será reiniciado, realizando todas as funções novamente.

4.2 Código Receiver

O código do receiver se baseia também em 4 partes, onde a primeira é receber um dado de um sender, ao receber o dado, ele será dissolvido, separando em id do sender e porcentagem do tanque com id recebido. A segunda parte se baseia em enviar o ACK com o id do sender, para identificar qual é o sender que foi ouvido e realizar uma contagem para saber quantos tanques ainda devem ser ouvidos. A terceira parte é a interação desses dados com o Adafruit IO, onde, baseado no ID, o dado será enviado para o feed configurado para aquele ID e salvo no banco de dados do Adafruit. Por último, o receiver permanecerá em estado ativo até que ele receba dados de todos os IDs configurados e após isso entrará em modo deep sleep até um tempo pré determinado. Que seja aproximadamente, o mesmo tempo do sender menos a quantidade de tempo que gastou recebendo dados de cada sender. Para que seja ativado um pouco antes do primeiro sender enviar seus dados. E assim reiniciando todo o ciclo.

4.3 Adafruit

A escolha do Adafruit foi decidida para um maior gama de opções na hora de tratar os dados e suas praticidades. Nos permitindo usar seu próprio banco de dados e enviar emails em casos necessários. Sua conexão ocorre devido a criação de uma variável global localizada no gateway e, se conectado a internet, toda vez que essa variável do ESP for atualizada, o dado será automaticamente

enviado para o feed escolhido e poderá ser acessada, tanto em modo de tabela, como em modo de gráfico, ou até mesmo em formato de dashboard. Há ainda a possibilidade de avisos em determinados casos. Se caso a porcentagem de tanque no diesel for menor que 25, por exemplo, Será enviado um email para o dono do feed que o tanque está com uma baixa porcentagem e que necessita ser reabastecido. Além de que a plataforma do Adafruit, nos permite criar uma planilha no google planilhas com todos os dados de todos os tanques e a última vez que cada dado foi atualizado.

5 Funcionalidades

- Dashboard customizável;
- Possibilidade de múltiplos tanques;
- Verificação de último nível medido de combustível de cada tanque;
- Gráfico com histórico de nível de cada tanque;
- Arquivo JSON com histórico dos dados;
- Alerta enviado por email em caso de baixo nível de combustível;

6 Testes e Resultados

Testamos o alcance da comunicação do LoRa em 2 situações, uma em um espaço pequeno e sem obstáculos e outro maior e com mais obstáculos.

No primeiro, estávamos nas 2 pontas do estacionamento do PAT, com aproximadamente 250m entre os módulos. Nesta situação, não obtivemos perda de pacote e o sinal estava excelente (-60dBm)

Já no segundo caso, um dos módulos estáva neste mesmo estacionamento porém o outro estáva em movimento (carro), vindo da STI para observar onde o módulo começava a receber pacotes. Quando o módulo passou por trás do final do ICC Sul, mesmo com um sinal muito fraco (-185 dBm) começaram a chegar pacotes, porém com muitas perdas já que o sinal estava ruim graças à distância (aprox 900m) e a quantidade de obstáculos entre as antenas.

7 Desafios e Soluções

Dentre as dificuldades que tivemos, as mais significativas foram - valores estranhos no ultrassom, compatibilidade com múltiplos tanques, envio somente com mudança de nível e envio de ACKs.

Para o sensor de distância com ultrassom, o problema era a alimentação do sensor, que foi resolvida alimentando-o com um Arduino Uno.

A compatibilidade com múltiplos tanques foi problemática por conta da plataforma Adafruit que não permite a mudança do parâmetro na conexão que deve ser constante. Para resolver o código do receptor faz a conexão com todos os "feeds" ao ligar e decide para qual enviar com um switch case que decide por um id em qual das conexões o valor deve ser salvo.

O envio de ACKs, para multiplos tanques também foi problemático, uma vez que havia o problema de sincronização e a forma de se resolver isso foi fazendo com que os dados fossem

enviados até que o ACK com seu ID fosse recebido. De modo que a partir do segundo envio, os próprios senders se organizam em fila e haverá apenas um ligado por vez, sem mais problemas com vários LoRas transmitindo ao mesmo tempo.

Por último, o envio de dados somente com a mudança do nível de combustível. Quando passamos a utilizar os modos de baixa energia, percebemos que as váriaveis (mesmo se globais) não eram armazenadas e toda vez que o módulo dorme elas se perdem. Dessa forma, não conseguimos saber qual foi o úlitmo nível medido para saber se ele mudou e se deve ser enviado porque houve uma mudança. Foi visto que há outros modos de sono no ESP32, no entanto, como o ESP32 armazenaria dados e seus periféricos ficariam ativos, mesmo que sem transmir ou receber dados, foi percebido que não valeria a pena e o consumo de energia ficaria maior, não sendo tão eficiente.

8 Possível Melhoria

Após a apresentação do projeto para o professor, ele nos pediu para propor uma possível melhoria na forma como realizamos a sincronia entre os nós para que eles possam dormir. Foi apontado pelo professor que quando a rede é iniciada ou quando um novo nó é adicionado à rede, que há um período significativo em que os dispositivos vão ficar mais acordados do que deveriam tentando sincronizar. A melhor solução que encontramos, de modo que ainda fique parecido com o que já foi implementado, seria a utilização de um mecanismo similar ao TDMA (time-division multiple access), porém simplificado. A ideia seria que o receiver transmitiria periodicamente um sinal de sincronização contendo um timestamp, o número de slots e a duração de cada slot. Dessa forma o sistema seria mais adaptativo à mudanças como a adição de um novo nó e teria um processo de sincronização mais eficiente, reduzindo as possíveis colisões de pacotes e melhorando a eficiência energética dos dispositivos que passariam menos tempo ligados desnecessáriamente.

9 Conclusão

Pode-se concluir que, dentre o proposto pelo professor, o projeto foi concluído com êxito e o resultado apresentado é satisfatório. Conseguimos elaborar os "nós/módulos" que comunicam entre si utilizando LoRa e suas funções de baixa energia. Além disso, a comunicação com a internet para fácil utilização por um usuário comum por meio de um dashboard também foi realizada e cumpre os requisitos do projeto.