CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA “PAULA SOUZA”

FACULDADE DE TECNOLOGIA DE BEBEDOURO

TECNOLOGIA EM BIG DATA NO AGRONEGÓCIO

MONITORAMENTO DE LAGOAS DE ÁGUA RESIDUARIA

AUTOR: JULIANO TIMÓTEO DA SILVA

ORIENTADOR: MARACELO JOSÉ ARISTEU

BEBEDOURO

2025

JULIANO TIMÓTEO DA SILVA

MONITORAMENTO DE LAGOAS DE ÁGUA RESIDUARIA

Monografia apresentada à Faculdade de Tecnologia de Bebedouro, como parte dos requisitos para a obtenção do título de Tecnólogo em Big Data no Agronegócio

Orientador: **prof. Marcelo José Aristeu**

BEBEDOURO

2025

“A tecnologia move o mundo”

STEVE JOBS

Agradecimentos

Agradeço primeiramente com todo o meu carinho e gratidão a:

Minha a minha esposa, que sempre acreditou em mim e me apoiou incondicionalmente. Sua paciência, amor e incentivo foram fundamentais para que eu chegasse até aqui.

Meu orientador, Marcelo, por sua orientação, apoio e valiosas contribuições ao longo desta jornada. Seu conhecimento e dedicação foram essenciais para a realização deste trabalho.

Meus amigos, por estarem ao meu lado em cada etapa desta caminhada, oferecendo apoio, compreensão e muitas vezes uma palavra de encorajamento quando eu mais precisava e a todos que contribuíram, direta ou indiretamente, para a conclusão deste projeto, meu sincero agradecimento.

Agradeço em especial à professora **Mariana Moretto**, por insistir e abrir os caminhos que tornaram possível a conclusão desta jornada. Sua dedicação e apoio foram essenciais para este trabalho.

SILVA, J.T. **Monitoramento de lagoas de águas residuárias**. Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica “Paula Souza”. Faculdade de Tecnologia de Bebedouro. nº p. 2025.

RESUMO

Este trabalho tem como objetivo o desenvolvimento de uma plataforma de monitoramento de lagoas de águas residuárias utilizando sensores IoT, com coleta de dados em tempo real. A automação permite detectar transbordamentos, otimizar o uso de mão de obra e promover segurança para os colaboradores. A vinhaça, subproduto da produção de etanol da cana-de-açúcar, é aplicada por fertirrigação, contribuindo para a produtividade agrícola e sustentabilidade ambiental. O uso de sensores HC-SR04 integrados ao ESP32, com visualização de dados via Power BI, mostra-se uma solução eficaz, sustentável e de baixo custo.

Palavras-chave: IoT. Sensores Vinhaça. Fertirrigação. Monitoramento.

SILVA, J.T. **Monitoramento de lagoas de águas residuárias**. Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica “Paula Souza”. Faculdade de Tecnologia de Bebedouro. nº p. 2025.

.

ABSTRACT

This work aims to develop a monitoring platform for wastewater lagoons using IoT sensors with real-time data collection. The automation enables the detection of overflows, optimizes labor usage, and promotes worker safety. Vinasse, a by-product of ethanol production from sugarcane, is applied through fertigation, contributing to agricultural productivity and environmental sustainability. The use of HC-SR04 sensors integrated with an ESP32 microcontroller and data visualization via Power BI proves to be an effective, sustainable, and low-cost solution.

***Keywords**: IoT. Sensors. Vinasse. Fertigation. Monitoring.*

SUMÁRIO

LISTA DE FIGURAS

Lista de Gráficos

Lista de Algoritmos

Lista de Siglas

Esp32 - Microcontrolador de 32 bits com Wifi e Bluetooth
GPIOs - General Purpose Input/Output
IoT - Internet of Things
V - Volts
SQL - (Structured Query Language)
DAX - Data Analysis Expressions
DB - Date Base
Step down - Conversor de tensão
Esp - Espressif Systems Processor
Wi-Fi - Wireless Fidelity
ADC - Analog to Digital Converter
DAC - Digital to Analog Converter
SPI - Serial Peripheral Interface
I2C - Inter Integrated Circuit
UART - Universal Asynchronous Receiver
IDE - Integrated Development Environment
MQTT - Message Queuing Telemetry Transport
HC-SR04 - High Accuracy Sensor Range 04(nome do modelo)
VCC - Voltage Common Colletor
GND - Ground
TRIG - Trigger
Cm - Centímetros
Us - Microssegundos
DC-DC - Direct Current to Direct Current
VIN - Voltage Input
VOUT - Voltage Output
PWN - Pulse Width Modulation
IC - Integrated Circuit
Amp ou A – Ampere
Volt - Volt
USB - Universal Serial Bus
I/O - Input/Output
EPROM - Electrally Erasable Programmable Read Only Memory
RDBMS - Relational Database Mangement System
OLTP - Oline Transatction Processing
MVCC - Multiversion Concurrency Control
JSON - Javascript Object Notation
QPS - Queries Per Second
TPS – Transactions Per Second

1 INTRODUÇÃO

O presente trabalho propõe o desenvolvimento de uma plataforma automatizada para o monitoramento de lagoas de águas residuárias por meio de sensores IoT, com o objetivo de prevenir transbordamentos, otimizar o uso de recursos humanos e garantir a segurança dos colaboradores. Essa abordagem contribui diretamente para a preservação ambiental e o uso eficiente dos recursos hídricos, especialmente na aplicação da vinhaça como fertilizante natural na agricultura tanto na preservação ambiental quanto na gestão dos colaboradores e dos recursos hídricos pois o uso descontrolado de material pode vir a contaminar lençóis freáticos, nascentes e rios matando a fauna e a flora. Por ser um resíduo industrial contaminante, o seu descarte tem que ser correto e de forma adequada, como as empresas sucroalcooleiras dominam da produção de etanol, na etapa de destilação para cada litro de álcool estima -se que é produzido de 10 a 15 litros de vinhaça.

Objetivo 1; monitorar em tempo real o nível dos lagos

Objetivo 2; saber em tempo real a estimativa em m³ em cada reservatório

Objetivo 3; melhorar a gestão de funcionários e equipamentos dispostos ao setor

O presente trabalho está segmentado em: 1 Introdução; 2 Fundamentação Teórica; Materiais e Métodos; 4 Resultados Esperados; 5 Conclusões; ao final as Referências.

2 FUNDAMENTAÇÃO TEÓRICA

Montar um sistema eletrônico IoT que visa monitorar os níveis de água residuária que são armazenados em lagos, resíduos esses produzido pela indústria sucroalcooleira, os níveis de profundidade desses lagos de armazenamento de vinhaça não podem chegar a sua capacidade máxima, então é imprescindível o monitoramento constante e utilizando sensores conectados ao microcontrolador ESP32 visamos controlar esses níveis.

ESP32

O ESP32, fabricado pela Espressif Systems, é um microcontrolador do tipo SoC (System on Chip) de 32 bits com conectividade Wi-Fi e Bluetooth, amplamente utilizado em aplicações de Internet das Coisas (IoT) devido ao seu desempenho, versatilidade e baixo consumo energético. Ele sucedeu ao ESP8266, oferecendo mais recursos de hardware e maior robustez, ideal para sistemas embarcados e conectados.

O livro Developing IoT Projects with ESP32 (Oner, 2023) destaca que o ESP32 “é um microcontrolador SoC de baixo custo e energia eficiente, que se tornou a base de inúmeros dispositivos Wi-Fi, impulsionando a inovação em IoT”

O autor apresenta recursos como processador Xtensa dual-core (até 240 MHz), interfaces ADC, DAC, SPI, I²C, UART, além de aceleradores criptográficos e diferentes modos de economia de energia.

A programação do ESP32 pode ser feita tanto usando o ESP-IDF (framework oficial da Espressif) quanto por meio da Arduino IDE usando a camada Arduino Core, com suporte a código em C e C++. Um tutorial recente aponta que a Arduino Core habilita C++11 “sem configuração extra”, facilitando o uso de recursos modernos da linguagem

HC-SR04

Implementar sensores HC-SR04 para a o acompanhamento dos níveis dos lagos unindo a comunicação wifi do microcontrolador ESP32, terá dados históricos reais diário da produção dos resíduos, dados esses que serão gravados em banco e tratados em uma plataforma para garantir a acessibilidade das informações.

O HC-SR04 é um sensor ultrassônico amplamente utilizado em sistemas embarcados e projetos de automação para medir distâncias de forma precisa e econômica. Ele funciona emitindo um pulso ultrassônico através do pino Trigger e medindo o tempo que o eco leva para retornar ao pino Echo, com base na velocidade do som. A distância até um objeto pode então ser calculada utilizando a fórmula:

DISTÂNCIA = (TEMPO DO ECO X VELOCIDADE DO SOM) /2

Esse tipo de sensor é útil em aplicações como robótica móvel, medição de nível de líquidos, sensoriamento de presença e desvios de obstáculos.

Segundo Silva (2020), o HC-SR04 é um sensor confiável e de baixo custo, operando com tensão de 5V e capaz de medir distâncias entre 2 cm e 400 cm com uma precisão de cerca de 3 mm.

STEP DOWN

Um conversor step-down, também conhecido como regulador buck, é um dispositivo eletrônico utilizado para reduzir a tensão de entrada para um valor de saída mais baixo, mantendo a corrente compatível com a carga conectada. Esse tipo de regulador é essencial em sistemas embarcados que operam com sensores, microcontroladores e módulos sensíveis a tensões específicas.

O princípio de funcionamento de um step-down baseia-se na comutação rápida de um transistor de potência em conjunto com indutores e capacitores, permitindo uma conversão eficiente de energia. Diferente dos reguladores lineares, os reguladores buck possuem alta eficiência energética, normalmente superior a 85%, pois minimizam perdas por dissipação de calor.

Segundo Horowitz e Hill (2016), os conversores buck são amplamente utilizados em aplicações modernas por permitirem operação eficiente em sistemas alimentados por baterias, fontes solares ou redes elétricas instáveis.

IDE Arduino

A IDE Arduino (Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado) é uma plataforma de software livre criada para facilitar a programação de placas eletrônicas Arduino. Seu propósito é oferecer uma interface simples, acessível e compatível com usuários iniciantes ou avançados, permitindo a escrita, compilação e envio de códigos (sketches) diretamente para a placa micro controladora.

A IDE Arduino suporta a linguagem de programação baseada em C/C++, com bibliotecas específicas que facilitam a manipulação de sensores, atuadores e módulos eletrônicos. Possui uma área de edição de código, botões para compilar e carregar, um terminal serial para comunicação em tempo real, e suporta diversas placas como Arduino Uno, Mega, Nano, ESP32, entre outras.

Segundo Banzai e Shiloh (2015), “a principal motivação por trás do Arduino era criar uma ferramenta que tornasse mais acessível o uso de microcontroladores por pessoas que não fossem engenheiras eletrônicas” (p. 22). Isso se refletiu no desenvolvimento da IDE, que abstrai grande parte da complexidade técnica e encurta a curva de aprendizado para prototipagem eletrônica.

MariaDB SQL

O MariaDB é um sistema gerenciador de banco de dados relacional (RDBMS), originado como um fork do MySQL, projetado para manter compatibilidade total, ao mesmo tempo em que implementa melhorias em desempenho, segurança e recursos.

De acordo com Razzoli, autor de Mastering MariaDB, “MariaDB combina características do MySQL, como replicação, transações e recuperação de desastres, com melhorias em desempenho, estabilidade, segurança e capacidades de monitoramento”.

Nos testes comparativos entre MariaDB e MySQL sob carga OLTP, MariaDB demonstrou desempenho competitivo, embora estudos mostrem que em cargas extremamente paralelas o MySQL geralmente leve vantagem. Por outro lado, comparado a bancos orientados a grafos como Neo4j, MariaDB apresentou desempenho até 29 vezes superior em consultas complexas após a devida otimização.

Além disso, segundo benchmarks independentes (Callaghan e ManagedServer, 2024), as versões mais recentes do MariaDB (ex. 11.4) mantêm desempenho consistente ao longo de várias versões e frequentemente superam o MySQL em throughput de inserção, leitura e escrita.

3 MATERIAIS E MÉTODOS

3.1 MATERIAIS

Hardware

- Microcontrolador ESP32
- Sensores ultrassônicos HC-SR04
- Placa Step-Down para alimentação dos sensores
- Jumpers de conexão
- Lcd I2C
- Led RGB

Software

- Arduino IDE (versão 2.3.6)
- Banco de dados MariaDB
- Power BI Desktop e Power BI App

3.2 MÉTODOS

O ESP32 é programado para realizar leituras dos sensores e enviar os dados por Wi-Fi para o banco de dados (Marias DB) esses dados são recebidos e armazenados em banco de dados local ainda na versão de testes.

Posteriormente esses dados são processados na plataforma software do Power BI, após o tratamento e gerenciamento esses dados são disponibilizados em tempo real por meio de dashboards no Power BI. A intenção de monitorar a profundidade dos lagos é calcular o volume total preenchido da lagoa, baseada em fórmulas matemáticas geométricas de volume.

Figura 1 - Esp

Figura 2 - Step Down

Figura 3 - Sensor Ultrassônico

3.3 PROGRAMAÇÃO EM C++

Programando com ide Arduino

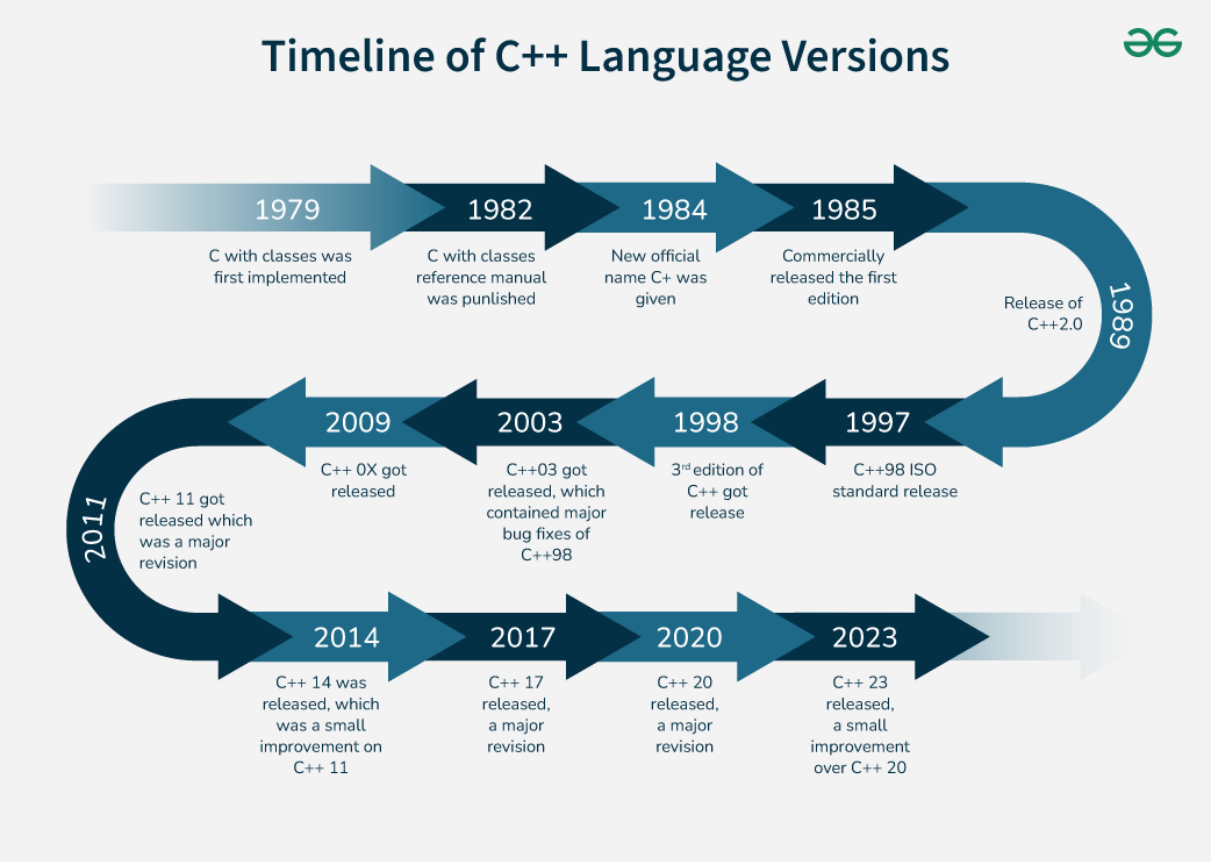
A linguagem utilizada para o desenvolvimento do código foi o C++, com o uso da IDE Arduino. O código foi elaborado para captar os dados de profundidade dos sensores HC-SR04 e realizar o envio dos dados para o banco MariaDB SQL via wifi, quer será armazenado inicialmente em uma máquina local.

História do C++

A linguagem C++ é uma linguagem de programação orientada a objetos e é uma combinação de linguagens de baixo e alto nível - uma linguagem de nível médio. A linguagem de programação foi criada, projetada e desenvolvida por um cientista da computação dinamarquês Bjarne Stroustrup, da Bell Telephone Laboratories (agora conhecida como Nokia Bell Labs) em Murray Hill, Nova Jersey. Como ele queria uma linguagem flexível e dinâmica que fosse semelhante a C com todos os seus recursos, mas com a adição de verificação de tipo ativa, herança básica, argumento de funcionamento padrão, classes, inlining etc., foi lançado o C com Classes(C++).

C++ era inicialmente conhecido como "C com classes" e foi renomeado C++ em 1983. ++ é uma abreviação para adicionar um à variedade na programação; portanto, C++ significa aproximadamente "um superior a C".

Figura 4 - Linha do Tempo C++



A origem da linguagem de programação C++ remonta a 1979, quando Bjarne Stroustrup estava desenvolvendo sua tese de doutorado. Uma das palavras com as quais Stroustrup teve a oportunidade de trabalhar foi uma linguagem chamada Simula, que, como o nome indica, era uma linguagem projetada principalmente para simulações. A linguagem simula 67 — essa foi a variante com a qual Stroustrup trabalhou — é considerada a principal linguagem a suportar o paradigma da programação orientada a objetos. Stroustrup descobriu que esse paradigma era útil para o desenvolvimento de pacotes; porém, a linguagem simula era muito lenta para uso prático.

Pouco depois, ele começou a trabalhar em " C com Classes ", pois o que o nome indica era para ser um superconjunto da linguagem C. Seu principal sonho era traduzir sua programação avançada orientada a objetos, criada por ele, para a linguagem C, que, em sua época, ainda era a linguagem de programação amplamente respeitada por sua mobilidade, portabilidade e compacidade, sem sacrificar a velocidade ou a praticidade de baixo nível. Sua linguagem de programação incluía inlining, herança básica, argumentos de função padrão, categorias e verificação de classificação confiável, além de todas as opções da linguagem C.

O primeiro compilador C com categorias foi chamado de Cfront, derivado de um compilador C chamado CPre . Era um programa projetado para traduzir código C com categorias para o C universal. Um ponto interessante a ser observado é que o Cfront foi escrito principalmente em C com classes, tornando-o um compilador auto-hospedado (um compilador que se

autocompila). O Cfront seria abandonado em 1993, quando se tornou difícil integrar novas opções, principalmente exceções C++. Mesmo assim, o Cfront teve um impacto significativo nas implementações de compiladores futuros e no sistema operacional UNIX.

Em 1983, o nome da linguagem foi modificado de C com categorias para C++. O operador ++ na linguagem C é o operador associado para incrementar uma variável, o que fornece algumas informações sobre como Stroustrup considerava a linguagem de programação. Várias novas opções foram adicionadas nessa época, sendo as mais notáveis funções virtuais, sobrecarga de execução, referências com a palavra-chave const e comentários de uma única linha usando as duas barras (que podem ser um recurso retirado da linguagem BCPL).

Em 1985, a relação de Stroustrup com a linguagem, intitulada " A Linguagem de Programação C++", foi impressa e lançada. Naquele mesmo ano, C++ foi imposto como um produto de propaganda e, portanto, teve seu início como um componente comercial. A linguagem de programação não foi formalmente padronizada, mas tornou o livro uma referência essencial. A linguagem de programação foi atualizada novamente em 1989 para incorporar membros protegidos e estáticos, ainda como uma herança associada de muitas categorias e classes.

Em 1990, o manual de C++ Anotado, que era uma referência, foi lançado em todo o mundo. Somente em 1990, no mesmo ano, o compilador Turbo C++ da Borland também seria lançado comercialmente como um produto de propaganda. O Turbo C++ adicionou uma série de outras bibliotecas que poderiam ter um impacto substancial no desenvolvimento do C++. Embora o último lançamento estável do Turbo C++ tenha sido em 2006, o compilador continua a ser amplamente utilizado.

Em 1998, o comitê consultivo e de padrões de C++ publicou o primeiro padrão internacional para C++, ISO/IEC 14882:1998, informalmente chamado de C++98. O Manual Anotado de C++ foi mencionado como tendo uma influência significativa no desenvolvimento da norma. A biblioteca de modelos de qualidade (também conhecida como Biblioteca de Modelos Padrão), que iniciou sua construção intelectual em 1979, também foi incluída nele. Em 2003, o comitê analisou vários problemas que estavam de acordo com o padrão de 1998 e o revisou posteriormente. A linguagem modificada foi denominada C++03.

Em 2005, o mesmo comitê de C++ publicou um relatório técnico (chamado TR1) detalhando diversas opções que visavam aprimorar o mais novo padrão C++. A nova regra foi informalmente chamada de C++0x, pois era esperado que fosse lançada antes do final da primeira década. Ironicamente, porém, o novo padrão só foi implementado em meados de 2011. Muitos relatórios técnicos foram publicados até então, e alguns compiladores começaram a adicionar suporte experimental para os novos recursos e opções.

Em meados de 2011, a nova versão C++ (chamada C++11) foi concluída. O projeto da biblioteca Boost teve um impacto substancial na nova versão, e alguns dos novos módulos foram derivados diretamente das bibliotecas Boost correspondentes. Algumas das novas opções incluídas foram:

Nova sintaxe de loop for que oferece praticidade, assim como os loops foreach em diferentes linguagens específicas

- Biblioteca de threading personalizada (que até 2011 não existia em C e C++)
- Modelos Variadic
- Palavra-chave de inferência automática de tipo (auto)
- Novas categorias e classes de instrumentação
- Nova biblioteca de tempo C++, suporte a atomics
- Biblioteca de organização abrangente
- Suporte a expressões regulares
- Maior suporte para uniões e listas de inicialização de array

Em 2020, o novo padrão C++, C++20 , foi finalizado, trazendo uma ampla gama de recursos e melhorias significativas para a linguagem. Muitas das novas funcionalidades foram influenciadas por propostas anteriores e contribuições da comunidade C++. Aqui estão alguns dos principais recursos introduzidos no C++20.

Figura 5 - Ide Arduino

4 BIBLIOTECAS USADAS NO CÓDIGO

MySQL_Connection.h

- Permite a conexão com um banco de dados MySQL através de um cliente WiFi (como o ESP32), possibilitando operações como inserção, leitura e atualização de dados.

MySQL_Encrypt_Sha1.h

- Usado para criptografar senhas com o algoritmo SHA1, como exigido por algumas versões do MySQL durante a autenticação.

MySQL_Packet.h

- Gerencia a criação e interpretação de pacotes de dados para comunicação entre o microcontrolador e o servidor MySQL.

Wire.h

- Biblioteca padrão para comunicação I2C. Usada para se comunicar com dispositivos I2C como sensores e displays LCD.

LiquidCrystal_I2C.h

- Facilita o controle de displays LCD com interface I2C, como escrever texto na tela ou limpar o display.

WiFi.h

- Permite a conexão do ESP32 (ou outro dispositivo com WiFi) a uma rede sem fio, essencial para uso da internet.

WiFiClient.h

- Permite criar clientes de rede TCP para se conectar a servidores via IP e porta (como servidores MySQL, web etc.).

TimeLib.h

- Proporciona funcionalidades para manipulação de tempo e data, como obter hora atual, fazer contagem de tempo etc.

NTPClient.h

- Utilizada para sincronizar o tempo via internet usando servidores NTP (Network Time Protocol).

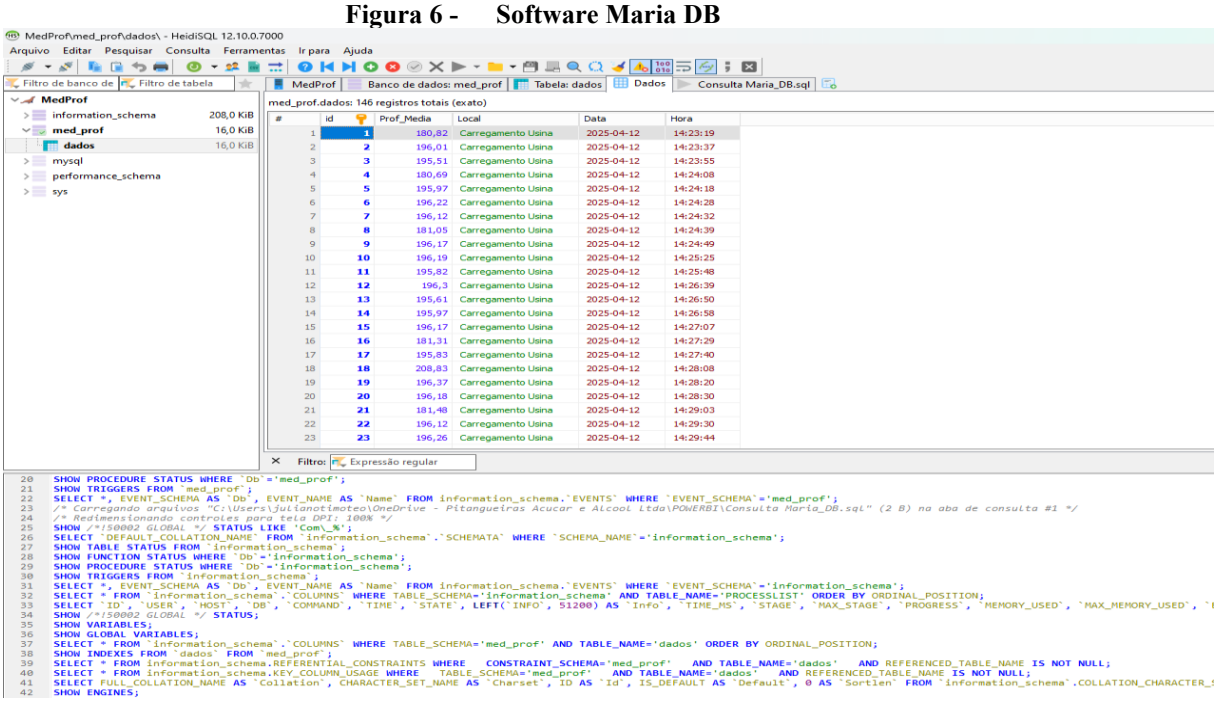
WiFiUdp.h

- Habilita a comunicação UDP sobre WiFi, essencial para o funcionamento da NTPClient.

4.1 MARIA DB SQL

Está é a interface de trabalho do software Maria DB, um sistema de banco de dados relacional compatível com MySQL usado para armazenar dados. Utiliza SQL (Structured Query Language) usada para realizar os inserções e consultas configurado para receber as informações geradas pelo projeto e armazenar em tabelas para posteriormente serem consultadas e usadas para gerar relatórios.

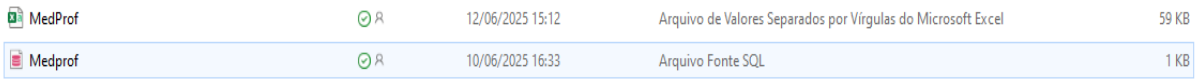
Figura 6 - Software Maria DB



Alguns comandos do banco no Maria DB

```
CREATE TABLE med_prof (
   id INT AUTO_INCREMENT PRIMARY KEY,
   nome VARCHAR (100) NOT NULL,
   especialidade VARCHAR (100),
   crm VARCHAR (20),
   telefone VARCHAR (20),
   e-mail VARCHAR (100),
   data_cadastro DATETIME DEFAULT CURRENT_TIMESTAMP
);
USE med_prof;
SHOW TABLES;
```

Figura 7 - Arquivos locais



5 RESULTADOS ESPERADOS

Com o uso do Power BI é uma ferramenta de análise de dados e criação de dashboards desenvolvidos pela Microsoft. Ela permite conectar, transformar, visualizar e compartilhar dados de forma interativa, facilitando a tomada de decisões com base em informações atualizadas.

- **Interface intuitiva**, com recursos arrastar-e-soltar.
- **Conexão com diversas fontes de dados** (Excel, SQL, web etc.).
- **Atualização automática de relatórios** e dashboards.
- **Compartilhamento fácil** de insights na nuvem ou em dispositivos móveis.

- **Uso da linguagem DAX**, que permite análises complexas e cálculos personalizados.

5.1 ANÁLISE DE DADOS

Neste aspecto a análise de dados ocorre quase que automaticamente em tempo real dependendo somente de uma conexão de rede para os dados serem visualizados em qualquer lugar, a praticidade de ter essas informações onde estiver e poder tomar a melhor decisão possível vai ajudar a gestão economizar tempo e recursos e evitar possíveis catástrofes ambientais, pois o vazamento desenfreado desses resíduos é extremamente perigoso para a fauna e flora locais.

Como já alertava PARACELSO, no século XVI: *"A diferença entre o remédio e o veneno está na dose"*

5.1.1 CALCULOS

Com base em alguns dados, é possível calcular o volume dos lagos e o volume preenchido com base nas leituras de profundidade enviadas pelos sensores permitindo realizar cálculos quase precisos.

Usando o cálculo de volume para o formato de tronco de pirâmide (ou prisma quadrado) onde a parte inferior ou a parte superior e são diferentes em tamanho, no caso especifico a base contém 45xm 45m e topo 50m x 50m e profundidade 3 m podemos usar uma fórmula de volume.

Fórmula do volume de um tronco de pirâmide quadrada:

Figura 8 - Fórmula do volume

$$V = \frac{h}{3} \times \left(A_1 + A_2 + \sqrt{A_1 \times A_2} \right)$$

Onde:

A1 = área da base superior = 50×50 = 2500 m2

A1 = área da base superior = 45×45 = 2500 m2

h = 3 m

Substituindo na fórmula:

Figura 9 - Resolução da equação

$$V = \frac{3}{3} \times \left(2500 + 2025 + \sqrt{2500 \times 2025} \right)$$

$$V = 1 \times \left(4525 + \sqrt{5\,062\,500} \right)$$

$$V = 1 \times (4525 + 2250)$$

$$\boxed{V = 6775\,\mathrm{m}^3}$$

Assim o volume do lago é de 6.775 metros cúbicos em sua totalidade.

Se quisermos saber o quanto de líquido temos se o lago tiver com 80 cm de profundidade A diferença agora é que a base inferior continua a mesma (45 × 45 m), mas a parte superior da superfície da água não é 50 × 50 m, pois a água ainda não chegou até o topo. Então, precisamos calcular a área da superfície da água proporcionalmente à altura preenchida usamos um fator de proporcionalidade da escalde altura preenchida.

Fator = 0,80 / 3 = 0,2667

Se o lado superior do lago é 50 m (no topo), então:

lado da superfície da água = 45 +(50-45) x 0,2667 = 45 + 5x 0,2667 = 45 +1,3335= 46,3335m

Cálculo das áreas:

Área da base inferior:

A2 = 45×45 = 2025 m²

A1 = 46,3335 46,335 ≈ 2147,83 m²

Figura 10 - Resolução da equação simulando preenchimento

Substituindo:

$$V = \frac{0{,}80}{3} \times \left(2147{,}83 + 2025 + \sqrt{2147{,}83 \times 2025}\right)$$

$$V = 0{,}2667 \times \left(4172{,}83 + \sqrt{4348915{,}75}\right)$$

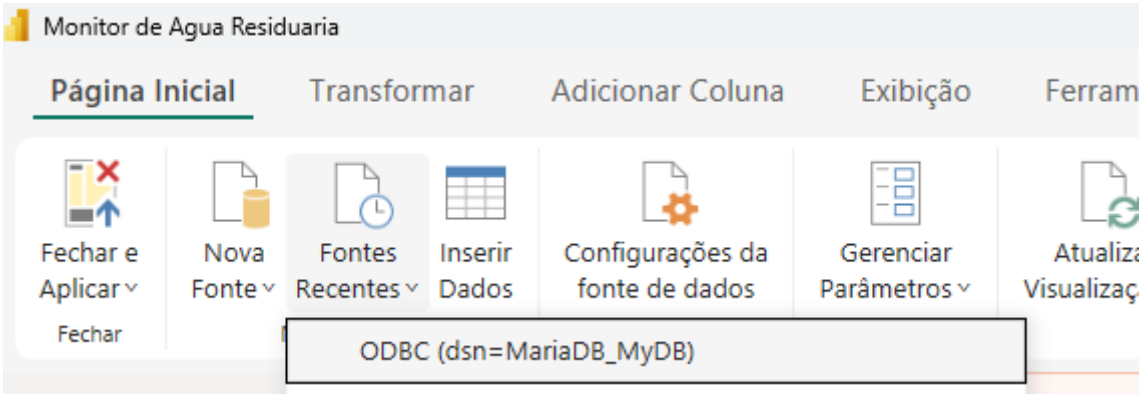
$$V = 0{,}2667 \times (4172{,}83 + 2085{,}4)$$

$$V = 0{,}2667 \times 6258{,}23 \approx \boxed{1668{,}86\,\mathrm{m}^3}$$

6 RESULTADOS E DISCUSSÕES

Já na fase final do projeto onde insiro os dados coletados pelos sensores e gravados no banco diretamente na plataforma do Power BI Desktop, acessando o usuário do banco de dados do Maria DB, Como na figura abaixo.

Figura 11 - Acessando o Banco Maria DB via Power BI



Ele recebe as informações já tratadas no banco tornando fácil o seu manuseio.

Figura 12 - Tela do Power Query do Power BI

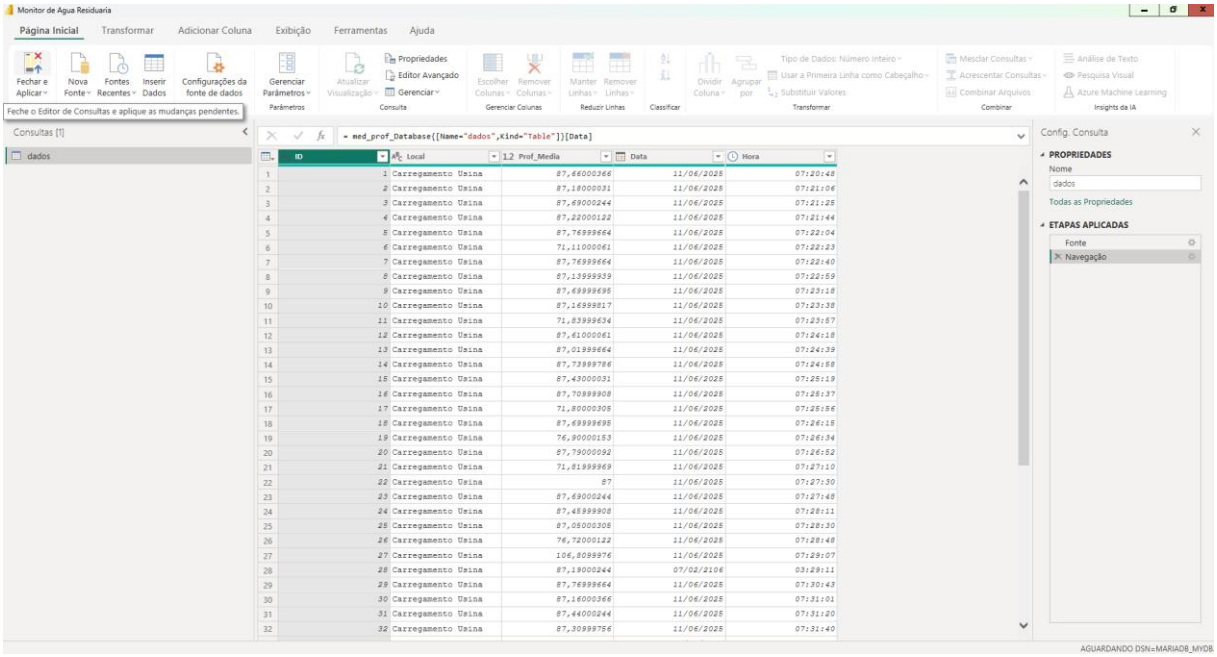


Figura 12 - Aqui a tabela já carregada me mostra como os dados gerados pelo dispositivo estão chegando para tratativas se caso necessário.

Figura 13 - Area de trabalho interno do Power BI

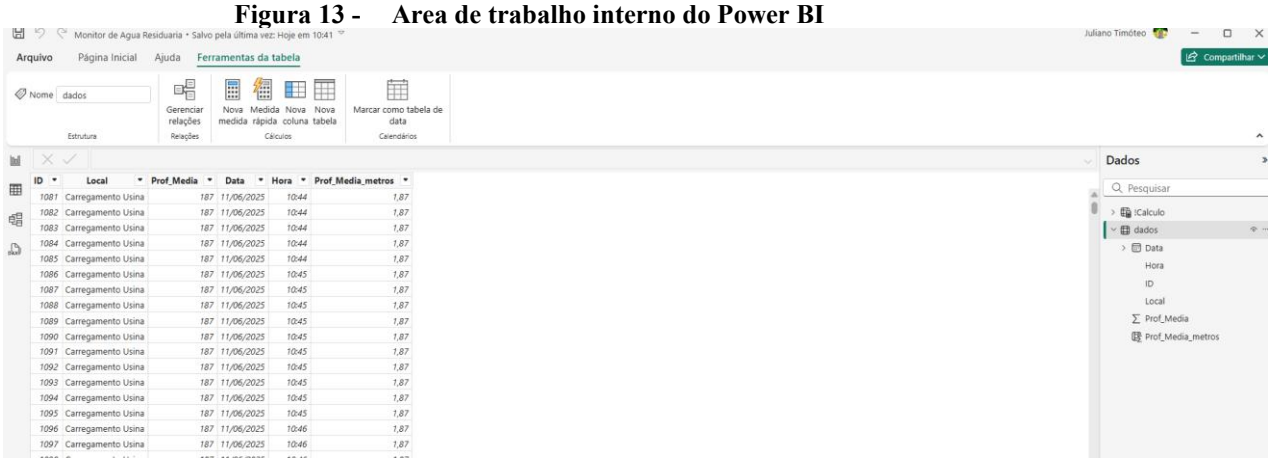


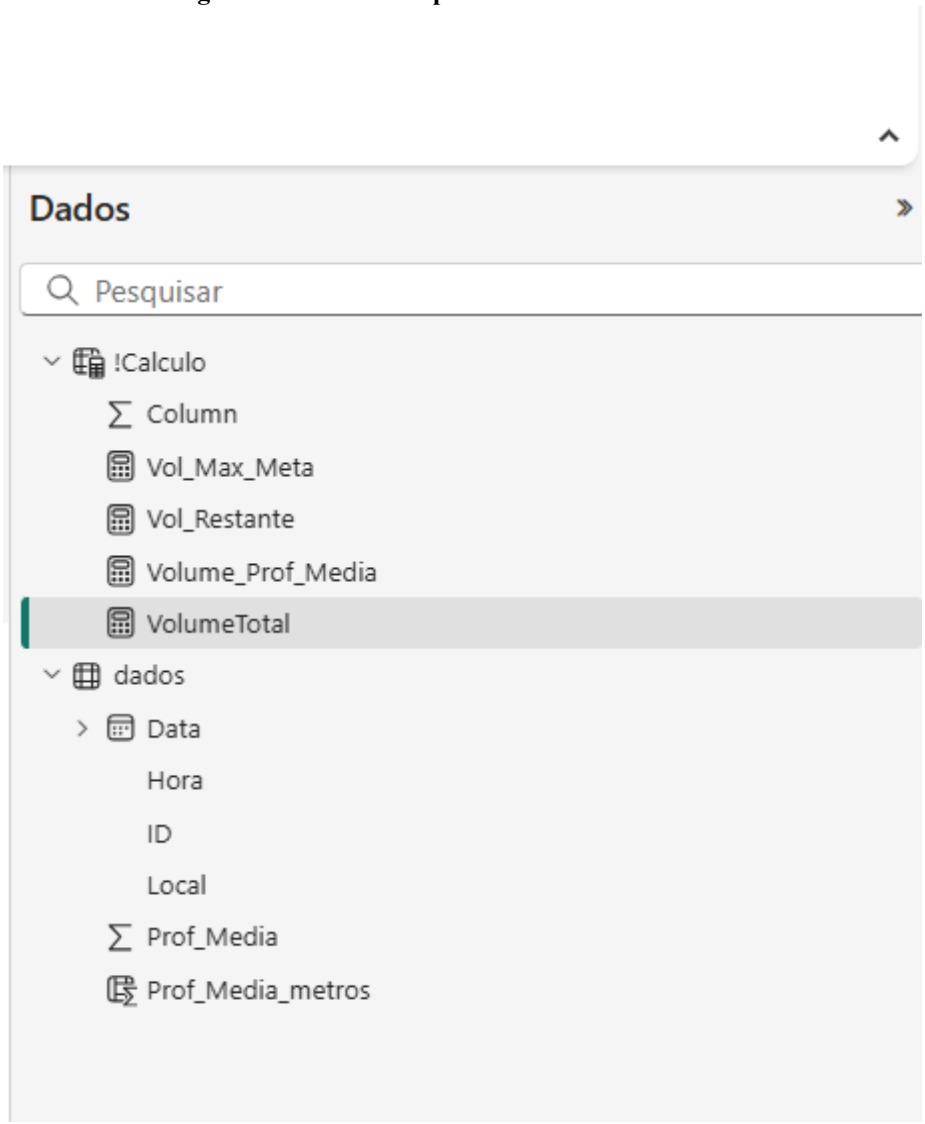
Figura 13 - Nessa parte onde criamos os cálculos em DAX e se preciso novas tabelas de apoio para execução no dashboard final.

Esse é um exemplo de fórmula DAX criada para calcular o volume total do lago que possivelmente receberia o experimento.

Figura 14 - Fórmula Dax de Volume

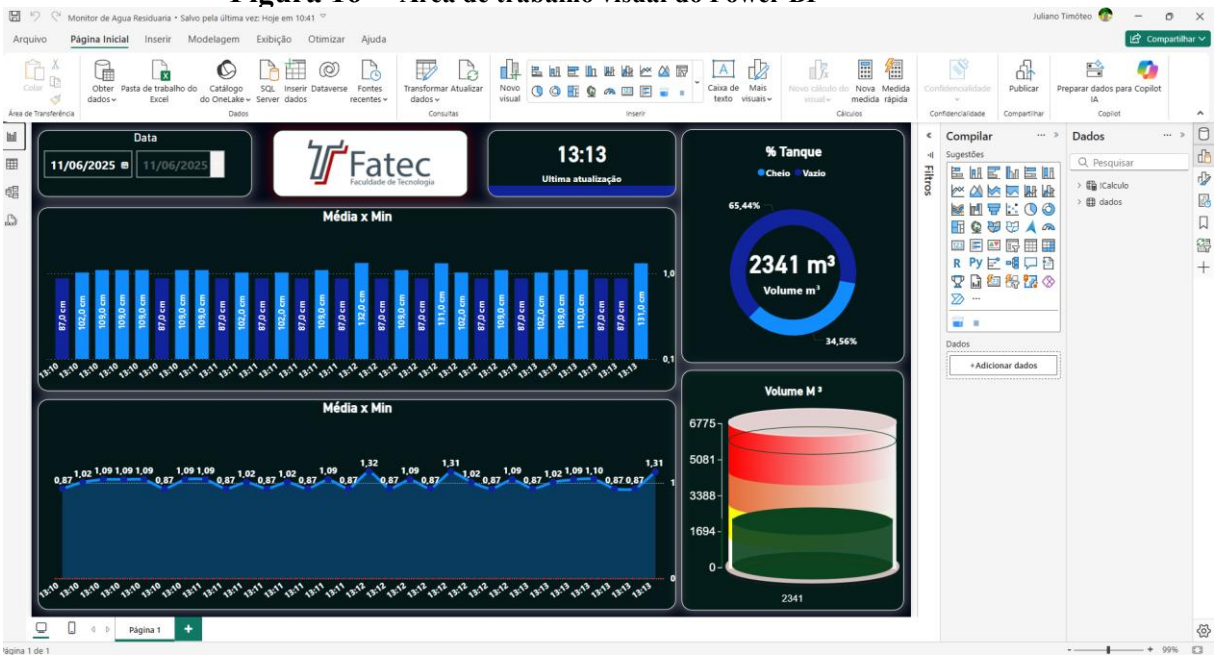

```
VolumeTotal =
VAR Altura = 3
VAR LadoTopo = 50
VAR LadoFundo = 45
VAR AreaTopo = LadoTopo * LadoTopo
VAR AreaFundo = LadoFundo * LadoFundo
VAR Volume =
    (Altura / 3) * (AreaTopo + AreaFundo + SQRT(AreaTopo * AreaFundo))
RETURN
    Volume
```

Figura 15 - Tela das pastas de trabalho



Nesta parte é extremamente importante deixar tudo organizado separando novas pastas de cálculos de dados obtidos para evitar qualquer tipo de confusão na hora de imputar os dados nos visuais do dashboard.

Figura 16 - Area de trabalho visual do Power BI



Com os dados devidamente tratados e cálculos feitos, então entramos na parte de desenvolvimento final onde são escolhidos os gráficos visuais, tabelas e imagens a serem usadas pois esta versão tem que ser pensada no entendimento do usuário final mostrando somente o que a de mais importante para ser visualizado.

7 CONCLUSÃO

O sistema de monitoramento proposto demonstrou-se eficaz na coleta de dados no envio e na visualização. O uso de tecnologias acessíveis, como sensores HC-SR04, ESP32, Maria DB e Power BI, permitiram que as amostras de dados fossem tratadas de forma fácil, pois chegaram organizadamente ao destino, uma gestão mais inteligente e eficiente dos resíduos líquidos da indústria sucroalcooleira. Além disso, a iniciativa poderá contribuir para práticas mais sustentáveis, reduzindo o impacto ambiental e melhorando a produtividade e desempenho da equipe agrícola.

8 REFERÊNCIAS

ARDUINO. Arduino IDE Download. Disponível em: https://www.arduino.cc/en/software/

Acesso em: 10 jan. 2025.

CANAL. Brincando com Ideias. Disponível em: Brincando com Ideias - YouTube. https://brincandocomideias.com.br/

Acesso em: 05 fev. 2025.

CANAL. Eletrônica Maker. Disponível em: Eletrônica Maker - YouTube.

https://www.youtube.com/@eletronicamaker

Acesso em: 05 fev. 2025.

CHEE, Sanhwa. Design Note 105: A New, High Efficiency Monolithic Buck Converter (LTC1265). Analog Devices, 2025. https://www.analog.com/en/resources/design-notes/ltc1265-monolithic-buck-converter.html

Acesso em: 14 fev. 2025.

COOPER, Chris. Fundamentals of Buck Converter Efficiency. Electronic Design, 23 maio 2013. https://www.electronicdesign.com/technologies/power/article/21798171/fundamentals-of-buck-converter-efficiency

Acesso em: 14 fev. 2025.

ESPRESSIF SYSTEMS. ESP32 Data Sheet (2017) e ESP32 Overview na Wikipédia. Disponível em: https://en.wikipedia.org/wiki/ESP32.

Acesso em: 01 Jan. 2025.

MARIADB FOUNDATION. MariaDB Download. Disponível em: https://mariadb.org. Acesso em: 03 fev. 2025.

MICROSOFT. Download do Power BI. Disponível em:

SEEED STUDIO. HC-SR04 features & tutorial. 2019. Disponível em: https://www.seeedstudio.com/blog/2019/11/04/hc-sr04-features-arduino-raspberrypi-guide/.

Acesso em: 02 jan. 2025.

TONGKAW, Sasalak; TONGKAW, Aumnat. “A comparison of database performance of MariaDB and MySQL with OLTP workload”. 2016 IEEE Conference on Open Systems (ICOS), 2016.

WIKIPEDIA. Buck converter. Atualizado em junho 2025. Disponível em: https://en.wikipedia.org/wiki/Buck_converter.

Acesso em: 13 jun. 2025.

APÊNDICE

Gráfico 1 - FAIXAS

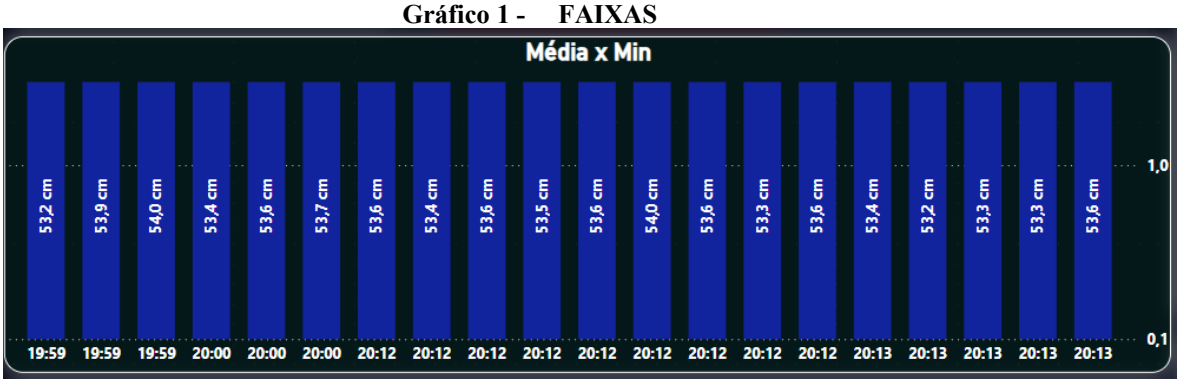


Gráfico 2 - AREA EMPILHADA

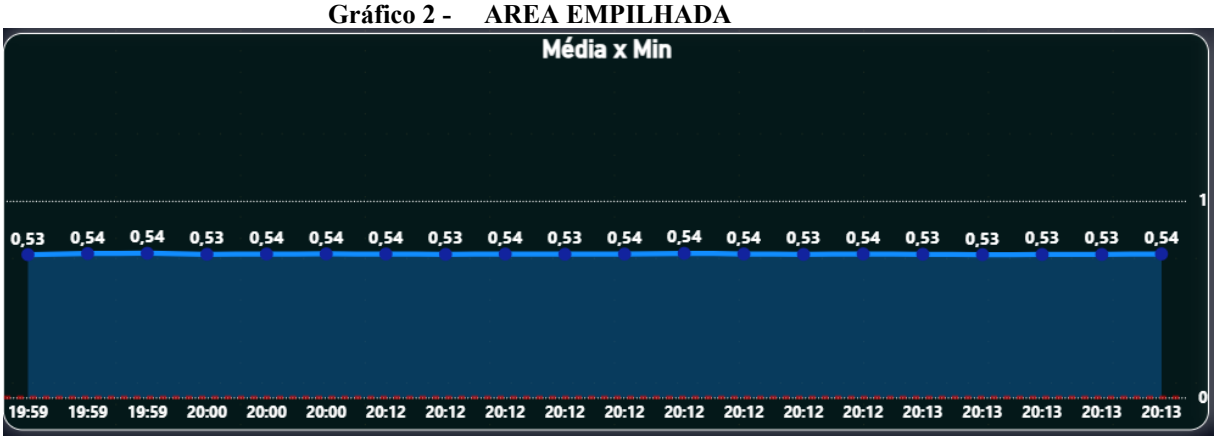
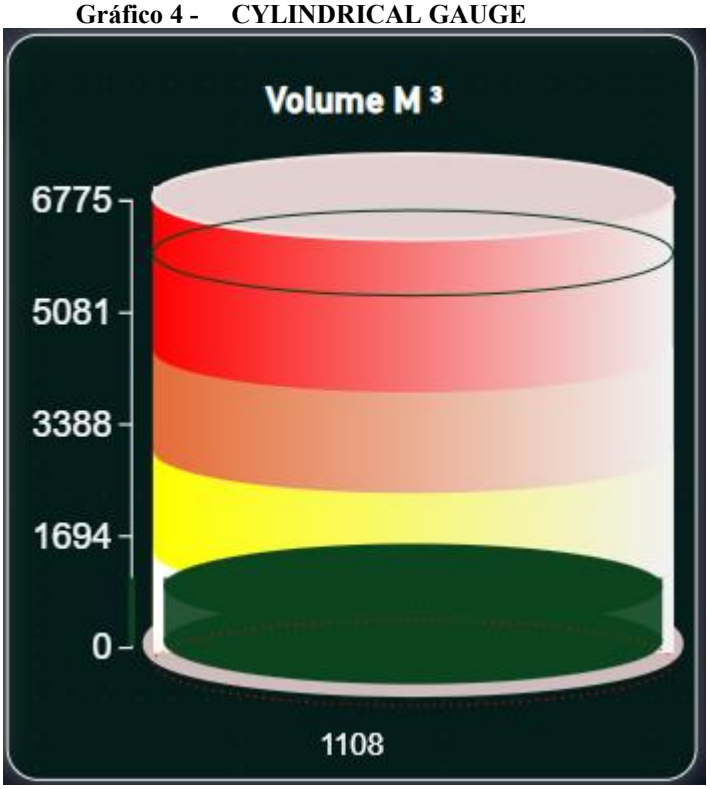


Gráfico 3 - ROSCA

Gráfico 4 - CYLINDRICAL GAUGE



CODIGO C++

```
#include <MySQL_Connection.h>
```

```
#include <MySQL_Cursor.h>
#include <MySQL_Encrypt_Sha1.h>
#include <MySQL_Packet.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <TimeLib.h>
#include <NTPClient.h>
#include <WiFiUdp.h>

#define COL 16
#define LIN 4
#define ENDE 0x27
LiquidCrystal_I2C lcd(ENDE, COL, LIN);

// Pinos sensores ultrassônicos
#define TRIG_PIN_1 13
#define ECHO_PIN_1 12
#define TRIG_PIN_2 27
#define ECHO_PIN_2 14
#define TRIG_PIN_3 33
#define ECHO_PIN_3 32

// LEDs
#define LED_VERMELHO 4
#define LED_AZUL 2
#define LED_VERDE 15

// Credenciais WiFi e banco MariaDB
const char* ssid = "JULIANO";
const char* password = "*******";
char server[] = "192.168.167.23";
uint16_t server_port = 3306;
```

```
char user[] = "MedProf";
char db_password[] = "*******#";
char database[] = "med_prof";
char table[] = "dados";
char local[] = "Carregamento Usina";

WiFiClient client;
MySQL_Connection conn((Client *)&client);

// Controle
bool conectadoWifi = false;
unsigned long lastWifiReconnectAttempt = 0;
unsigned long wifiReconnectInterval = 60000;
float lastDistance = -1; // Distância inválida inicial

// NTP
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "pool.ntp.org", -3 * 3600, 60000);

// Função para limpar uma linha do LCD
void limparLinha(int linha) {
  lcd.setCursor(0, linha);
  lcd.print("                "); // 16 espaços para limpar a linha
}

// Função para mostrar o IP alinhado 3 casas à esquerda na linha 3 do LCD
void mostrarIPnaLinha3() {
  limparLinha(3);
  String ipStr = WiFi.localIP().toString();
  int startCol = ((16 - (int)ipStr.length()) / 2);
  if (startCol < 0) startCol = 0; // Evita posição negativa
  lcd.setCursor(startCol, 3);
  lcd.print(ipStr);
}
```

```
// Função para conectar ao WiFi
void conectarWiFi() {
  Serial.println("Iniciando conexão com o Wi-Fi...");
  limparLinha(2);
  lcd.setCursor(0, 2);
  lcd.print("Conectando...");
  limparLinha(3);

  WiFi.begin(ssid, password);
  int tentativas = 0;

  while (WiFi.status() != WL_CONNECTED && tentativas < 6) {
    delay(1000);
    Serial.print(".");
    tentativas++;
  }

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("\nConectado ao Wi-Fi!");
    Serial.print("IP: ");
    Serial.println(WiFi.localIP());

    conectadoWifi = true;
    timeClient.begin();

    limparLinha(2);
    lcd.setCursor(0, 2);
    lcd.print("WiFi OK!");
    mostrarIPnaLinha3();

    digitalWrite(LED_VERDE, HIGH);
    digitalWrite(LED_VERMELHO, LOW);
    digitalWrite(LED_AZUL, LOW);
    delay(2000);
```

```
  } else {
    Serial.println("\nFalha ao conectar ao Wi-Fi.");
    conectadoWifi = false;

    limparLinha(2);
    lcd.setCursor(0, 2);
    lcd.print("Erro WiFi!");
    limparLinha(3);
    lcd.setCursor(0, 3);
    lcd.print("Reconectando...");

    digitalWrite(LED_VERMELHO, HIGH);
    digitalWrite(LED_AZUL, LOW);
    digitalWrite(LED_VERDE, LOW);
  }
}

// Função para medir distância do sensor ultrassônico
float getDistance(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
  if (duration == 0) return -1;  // Sem leitura válida
  return (duration / 2.0) * 0.0343;
}

// Função para verificar sensores, exibir e enviar dados
void verificarSensores() {
  float distance1 = getDistance(TRIG_PIN_1, ECHO_PIN_1);
  float distance2 = getDistance(TRIG_PIN_2, ECHO_PIN_2);
  float distance3 = getDistance(TRIG_PIN_3, ECHO_PIN_3);
```

```
  Serial.print("Distancia Sensor 1 (TRIG "); Serial.print(TRIG_PIN_1); Serial.print("): "); Serial.println(distance1);
  Serial.print("Distancia Sensor 2 (TRIG "); Serial.print(TRIG_PIN_2); Serial.print("): "); Serial.println(distance2);
  Serial.print("Distancia Sensor 3 (TRIG "); Serial.print(TRIG_PIN_3); Serial.print("): "); Serial.println(distance3);


 float sumDistances = 0;
 int validReadings = 0;
 String sensorErrorMsg = "";

 if (distance1 > 2 && distance1 < 300) {
  sumDistances += distance1;
  validReadings++;
 } else {
  sensorErrorMsg += "S1 ";
 }
 if (distance2 > 2 && distance2 < 300) {
  sumDistances += distance2;
  validReadings++;
 } else {
  sensorErrorMsg += "S2 ";
 }
 if (distance3 > 2 && distance3 < 300) {
  sumDistances += distance3;
  validReadings++;
 } else {
  sensorErrorMsg += "S3 ";
 }

 if (validReadings > 0) {
  float averageDistance = sumDistances / validReadings;
  lastDistance = averageDistance;

  limparLinha(0);
```

```
    lcd.setCursor(0, 0);
    lcd.print("Profundidade:");
    limparLinha(1);
    lcd.setCursor(0, 1);
    lcd.print(averageDistance);
    lcd.print(" cm");

    Serial.print("Profundidade media: ");
    Serial.println(averageDistance);

    if (validReadings < 3) {
      Serial.print("Falha na leitura dos sensores: ");
      Serial.println(sensorErrorMsg);
    }

     // Agora sempre insere no DB a média válida (mesmo com menos de 3 sensores válidos)
    inserirDadosMariaDB(averageDistance);

  } else {
    Serial.println("Falha na leitura de TODOS os sensores.");
    limparLinha(0);
    lcd.setCursor(0, 0);
    lcd.print("Profundidade:");
    limparLinha(1);
    lcd.setCursor(0, 1);
    if (lastDistance != -1) {
      lcd.print(lastDistance);
      lcd.print(" cm (S. OFF)");
    } else {
      lcd.print("N/A (S. OFF)");
    }
  }
}
```

```
// Função para inserir dados no banco MariaDB
void inserirDadosMariaDB(float averageDistance) {
  if (conectadoWifi) {
    if (conn.connect(server, server_port, user, db_password)) {
      Serial.print("Conectado ao servidor MariaDB no IP: ");
      Serial.println(server);

      delay(500);
      timeClient.update();
      time_t rawtime = timeClient.getEpochTime();
      struct tm *ti = localtime(&rawtime);
      char data[11];
      char hora[9];
      strftime(data, sizeof(data), "%Y-%m-%d", ti);
      strftime(hora, sizeof(hora), "%H:%M:%S", ti);

      String INSERT_SQL = String("INSERT INTO ") + database + "." + table +
                          " (Local, Prof_Media, Data, Hora) VALUES ('" + local + "', " +
                          String(averageDistance) + ", '" + data + "', '" + hora + "')";

      Serial.print("Query SQL: ");
      Serial.println(INSERT_SQL);

      MySQL_Cursor *cur_mem = new MySQL_Cursor(&conn);
      if (!cur_mem->execute(INSERT_SQL.c_str())) {
        Serial.println("Erro ao inserir dados");
        limparLinha(2);
        lcd.setCursor(0, 2);
        lcd.print("Erro DB!");
        limparLinha(3);
        lcd.setCursor(0, 3);
        lcd.print("Falha ao gravar");
      } else {
        Serial.println("Dados inseridos com sucesso!");
        limparLinha(2);
```

```
      lcd.setCursor(0, 2);
      lcd.print("Dados enviados!");
      limparLinha(3);
      lcd.setCursor(0, 3);
      lcd.print("                ");
      delay(1000);
      limparLinha(2);
      lcd.setCursor(0, 2);
      lcd.print("WiFi OK!");
      mostrarIPnaLinha3();
     }
     delete cur_mem;
     conn.close();
    } else {
     Serial.println("Falha na conexao com o banco de dados.");
     limparLinha(2);
     lcd.setCursor(0, 2);
     lcd.print("Erro DB!");
     limparLinha(3);
     lcd.setCursor(0, 3);
     lcd.print("Verifique rede");
    }
  } else {
    Serial.println("Nao conectado ao Wi-Fi, nao inserindo dados no MariaDB.");
  }
}

// Função para reconectar ao WiFi periodicamente
void reconectarWiFi() {
  if (!conectadoWifi && (millis() - lastWifiReconnectAttempt > wifiReconnectInterval)) {
    Serial.println("Tentando reconectar Wi-Fi...");
    conectarWiFi();
    lastWifiReconnectAttempt = millis();
  }
}
```

```
void setup() {
  Serial.begin(115200);
  lcd.init();
  lcd.backlight();
  lcd.clear();

  pinMode(LED_VERMELHO, OUTPUT);
  pinMode(LED_AZUL, OUTPUT);
  pinMode(LED_VERDE, OUTPUT);

  pinMode(TRIG_PIN_1, OUTPUT);
  pinMode(ECHO_PIN_1, INPUT);
  pinMode(TRIG_PIN_2, OUTPUT);
  pinMode(ECHO_PIN_2, INPUT);
  pinMode(TRIG_PIN_3, OUTPUT);
  pinMode(ECHO_PIN_3, INPUT);

  conectarWiFi();
}

void loop() {
  reconectarWiFi();
  verificarSensores();
  delay(2000);
}
```

CODIGOS SQL

Criação e recriação do usuário MedProf com senha ******* e, posteriormente, **********.
Hosts permitidos:
'localhost'
'%' (qualquer host)
'IP_DO_ESP32' (específico para integração com ESP32)
Permissões:

GRANT ALL PRIVILEGES ON med_prof.* TO 'MedProf'@'...'

Seguido sempre por FLUSH PRIVILEGES

Alterações de senha usando:

SET PASSWORD FOR ...

ALTER USER ... IDENTIFIED BY ...

ALTER USER ... IDENTIFIED VIA mysql_native_password USING PASSWORD(...)

Banco de Dados e Tabelas

Criação do banco:

CREATE DATABASE med_prof;

Criação da tabela dados (repetidamente, com pequenas alterações):

sql

CopiarEditar

```
CREATE TABLE dados (
    id INT AUTO_INCREMENT PRIMARY KEY,
    Local VARCHAR(100),
    Prof_Media FLOAT,
    Data DATE,
    Hora TIME
);
```

Uso da base:

USE med_prof;

Consultas Executadas

Visualização de usuários:

SELECT User, Host FROM mysql.user WHERE User = 'MedProf';

Visualização de tabelas e estrutura:

SHOW TABLES;

DESCRIBE dados;

Visualização de registros:

SELECT * FROM med_prof.dados ORDER BY id DESC LIMIT 10;