INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

COORDENAÇÃO DO CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA ELÉTRICA

GABRIEL BELIZÁRIO ALVES

MONITORAMENTO DO NÍVEL DE ÁGUA EM RESERVATÓRIOS RESIDENCIAIS UTILIZANDO SENSOR ULTRASSÔNICO

João Pessoa

2023

GABRIEL BELIZÁRIO ALVES

MONITORAMENTO DO NÍVEL DE ÁGUA EM RESERVATÓRIOS RESIDENCIAIS UTILIZANDO SENSOR ULTRASSÔNICO

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso Superior em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba.

Orientador: Prof. Dr. Thiago de Carvalho Batista

João Pessoa

2023

Dados Internacionais de Catalogação na Publicação – CIP
Biblioteca Nilo Peçanha – IFPB, *Campus* João Pessoa

A474m Alves, Gabriel Belizário
Monitoramento do nível de água em reservatórios residenciais / Gabriel Belizário Alves. – 2023.
70 f.

TCC (Graduação – Bacharelado em Engenharia Elétrica) – Instituto Federal da Paraíba – IFPB / Coordenação de Engenharia Elétrica, 2023.

Orientador: Profº Dr. Thiago de Carvalho Batista.

1. Sistema de Controle Automático – Protótipo 2. Tecnologia embarcada. 3. Monitoramento em tempo real. 4. Recursos hídricos – Monitoramento. I. Título.

CDU 681.51:556.18

Bibliotecária responsável Ivanise Andrade Melo de Almeida – CRB15/96

GABRIEL BELIZÁRIO ALVES

MONITORAMENTO DO NÍVEL DE ÁGUA EM RESERVATÓRIOS RESIDENCIAIS UTILIZANDO SENSOR ULTRASSÔNICO

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica, pelo Curso Superior em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba.

Aprovado em 19 / 12 / 2023

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
THIAGO DE CARVALHO BATISTA
Data: 20/12/2023 11:16:01-0300
Verifique em https://validar.iti.gov.br

Thiago de Carvalho Batista, Dr.

gov.br Documento assinado digitalmente
CLEUMAR DA SILVA MOREIRA
Data: 20/12/2023 16:05:28-0300
Verifique em https://validar.iti.gov.br

Cleumar da Silva Moreira, Dr.

gov.br Documento assinado digitalmente
ERIK FARIAS DA SILVA
Data: 20/12/2023 13:47:35-0300
Verifique em https://validar.iti.gov.br

Erik Farias da Silva, Dr.

Aos meus pais,

por toda dedicação em prol da

minha educação ao longo de suas vidas.

AGRADECIMENTOS

Primeiramente, agradeço à Deus por ter me dado forças em meio as turbulências para começar concretizar esse trabalho de conclusão, mesmo após o tempo que se passou desde a minha trajetória pelo curso.

Agradeço aos meus pais, Lucimara Belizário e Januário Alves, que sempre me apoiaram, aconselharam, ajudaram, exigiram na hora certa, e que continuarão sempre ao meu lado. Vocês possuem todo meu amor e gratidão por terem se sacrificado tanto à minha educação e futuro.

A Irlly Tammara Alves de Araújo, por ter escutado minhas lamentações e frustrações, mas principalmente por sempre me lembrar que eu conseguiria, bastava querer.

Aos colegas que encontrei pela caminhada do curso, com quem tive o prazer de dividir tarefas e risadas que aliviavam os dias difíceis.

Aos professores que encontrei durante toda a caminhada, o profissionalismo e a dedicação que tiveram comigo e tem com os alunos da Instituição são dificilmente encontrados em outros lugares. Em especial ao professor Thiago Batista pela orientação durante todo o tempo que necessitei.

A Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica, que sempre esteve solicita nos momentos mais necessitados.

Ao IFPB com um todo, os profissionais de todos os setores não medem esforços para proporcionar o melhor ambiente possível.

“O sucesso nada mais é que ir de fracasso em fracasso,
sem que se perca o entusiasmo.”
(Winston Churchill)

RESUMO

Inserido no contexto cada vez mais acentuado de sustentabilidade e gestão ambiental, este estudo tem como propósito realizar testes laboratoriais para iniciar a prototipagem de uma plataforma destinada ao monitoramento em tempo real do nível de água em reservatórios. Com o intuito de cumprir essa finalidade, foi desenvolvida uma interface web para disponibilizar os dados de nível de água coletados em uma plataforma online de fácil acesso e compreensão. A validação dos componentes foi realizada por meio de medições de nível de água com o protótipo desenvolvido, as quais foram analisadas estatisticamente para avaliar a precisão e consistência das leituras. Os resultados obtidos demonstraram uma performance satisfatória do protótipo, fornecendo dados precisos e confiáveis, contribuindo assim para aplicações no monitoramento hídrico residencial.

Palavras-chave: Prototipagem para monitoramento de nível de água; Tecnologia embarcada; Monitoramento em tempo real.

ABSTRACT

Inserted into the increasingly accentuated context of sustainability and environmental management, this study aims to conduct laboratory tests to initiate the prototyping of a platform for real-time monitoring of water level in reservoirs. In order to fulfill this purpose, a web interface was developed to make the collected water level data available on an online platform that is easily accessible and comprehensible. The validation of the components was performed through water level measurements with the developed prototype, which were statistically analyzed to assess the accuracy and consistency of the readings. The obtained results demonstrated satisfactory performance of the prototype, providing precise and reliable data, thus contributing to applications in environmental monitoring.

Keywords: Prototyping for water level monitoring; Embedded technology; Real-time monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A humanidade tem se ocupado com a água como uma necessidade vital e como uma ameaça potencial pelo menos desde o tempo em que as primeiras civilizações se desenvolveram as margens dos rios (DORNELLES, 2015). Enquanto a hidrologia é a ciência que estuda a água na terra, a engenharia hidrológica é a aplicação dos conhecimentos da hidrologia para resolver problemas relacionados aos usos da água (DORNELLES, 2015). No contexto desse desafio, a engenharia elétrica desempenha um papel fundamental ao proporcionar soluções inteligentes e sustentáveis. Este trabalho de conclusão de curso busca abordar uma aplicação prática e relevante dessa disciplina, explorando a implementação de um sistema de medição e monitoramento para reservatórios de água.

Planejamento, desenvolvimento e gestão dos recursos hídricos garantem o abastecimento de água de qualidade, acessível e sustentável para seres humanos e ecossistemas (LOUCKS 2017). Nesse contexto, a utilização de sensores ultrassônicos surge como uma abordagem inovadora para a determinação precisa do nível de água em reservatórios. A tecnologia ultrassônica permite a medição não intrusiva e em tempo real da altura do nível d'água, possibilitando a obtenção do volume do reservatório de forma precisa e eficaz.

O sistema proposto baseia-se na aplicação de princípios da engenharia elétrica para integrar sensores ultrassônicos, microcontroladores e técnicas de processamento de sinais. A coleta de dados em tempo real é essencial para fornecer informações precisas sobre o estado do reservatório, contribuindo para a detecção de problemas no sistema de água e a prevenção de desperdícios.

Além disso, a automação e o monitoramento do sistema proporcionam a oportunidade de implementar estratégias inteligentes de gerenciamento, como a programação de abastecimento automático, alertas de níveis críticos e a integração com sistemas de monitoramento remoto. Essas funcionalidades visam não apenas melhorar a eficiência operacional, mas também reduzir os impactos ambientais associados ao uso inadequado dos recursos hídricos.

Ao abordar esta aplicação prática na interseção entre a engenharia elétrica e a gestão de recursos hídricos, este trabalho visa contribuir para o avanço tecnológico e a promoção de práticas mais sustentáveis no uso da água. A exploração desses conceitos fundamenta-se na

compreensão dos princípios da engenharia elétrica aplicados a sistemas do mundo real, demonstrando o potencial dessa abordagem para enfrentar desafios contemporâneos e fomentar soluções inovadoras.

2. OBJETIVO GERAL

O principal objetivo deste trabalho é projetar, implementar e avaliar um sistema de medição e monitoramento de reservatórios de água, utilizando sensores ultrassônicos e princípios da engenharia elétrica. O sistema visa otimizar a gestão de recursos hídricos, fornecendo informações precisas sobre o nível e volume da água em tempo real.

2.1. OBJETIVOS ESPECÍFICOS

Analisar Fundamentos da Engenharia Elétrica

Realizar uma revisão bibliográfica dos conceitos fundamentais da engenharia elétrica relevantes para o desenvolvimento do sistema, incluindo princípios de eletrônica, instrumentação e controle.

Desenvolvimento do Sistema de Monitoramento e Sensor Adequado

Escolher e configurar um microcontrolador adequado para o sistema, e o sensor mais apropriado para a aplicação

Implementar algoritmos de processamento de sinais para converter dados dos sensores em informações de nível e volume.

Desenvolver a lógica de monitoramento para detecção e monitoramento do nível do reservatório.

Integração e Testes do Sistema

Integrar os componentes eletrônicos em um protótipo funcional do sistema.

Realizar testes em condições controladas para avaliar a precisão e confiabilidade da medição de níveis e volumes d'água.

Avaliação do Desempenho e Eficiência

Avaliar o desempenho do sistema em diferentes condições operacionais.

Analisar a eficiência do sistema em termos de economia de água, prevenção de desperdícios e respostas a situações de emergência.

Proposta de Melhorias e Otimizações

Identificar possíveis melhorias no sistema e propor otimizações para aumentar sua eficácia e aplicabilidade.

Considerar a integração de tecnologias adicionais para aprimorar a funcionalidade do sistema.

Ao atingir esses objetivos específicos, espera-se que o trabalho contribua significativamente para a aplicação prática de conceitos da engenharia elétrica no desenvolvimento de soluções inovadoras e sustentáveis no gerenciamento de recursos hídricos.

3. JUSTIFICATIVA

A gestão de recursos hídricos, no sentido lato, é a forma pela qual se se pretende equacionar e resolver as questões de escassez relativa dos recursos hídricos, bem como fazer o seu uso adequado, visando a otimização dos recursos em benefício da sociedade (MOREIRA, 2006). Nesse contexto, a engenharia elétrica emerge como uma disciplina fundamental, capaz de integrar tecnologias avançadas para enfrentar esse desafio de forma eficaz. A presente pesquisa encontra sua justificativa na urgência de desenvolver um sistema acessível e econômico para medição e monitoramento de reservatórios de água. A utilização de sensores ultrassônicos, aliada aos princípios da engenharia elétrica, representa uma abordagem estratégica para aprimorar a eficiência na gestão dos recursos hídricos. Além disso, a integração de conceitos de Internet of Things (IoT) e automação residencial fortalece a proposta, promovendo não apenas a acessibilidade econômica, mas também a implementação de um sistema inteligente e sustentável.

Otimização do Consumo de Água

A implementação de um sistema de medição preciso e em tempo real permite a otimização do consumo de água, evitando excessos e desperdícios. A informação acurada sobre o nível e volume d'água possibilita a tomada de decisões informadas sobre o abastecimento, contribuindo para a eficiência no uso desse recurso vital.

Redução de Desperdícios e Impactos Ambientais

Ao incorporar um sistema automatizado de controle do abastecimento, o sistema proposto contribui para a redução de desperdícios, minimizando impactos ambientais associados à exploração inadequada dos recursos hídricos. Isso é particularmente relevante em um contexto global de preocupação com a preservação do meio ambiente.

Integração de Tecnologias Inteligentes

A aplicação de princípios da engenharia elétrica permite a integração de tecnologias inteligentes, como algoritmos de processamento de sinais e microcontroladores, proporcionando não apenas medições precisas, mas também automação e controle eficiente do sistema. Essa abordagem alinha-se à busca por soluções tecnológicas avançadas para desafios contemporâneos.

Aplicabilidade em Diferentes Contextos:

O sistema proposto possui potencial aplicabilidade em diversos contextos, desde sistemas de abastecimento de água municipais até aplicações em setores agrícolas e industriais. Sua versatilidade o torna uma ferramenta valiosa para diferentes cenários, ampliando seu impacto e relevância.

Contribuição para a Pesquisa e Desenvolvimento Tecnológico

A pesquisa e implementação de sistemas inovadores no campo da engenharia elétrica contribuem para o avanço do conhecimento científico e tecnológico. Este trabalho busca preencher uma lacuna ao explorar uma aplicação específica dessa disciplina na gestão de recursos hídricos, oferecendo *insights* valiosos para futuras pesquisas e desenvolvimentos na área.

Em resumo, a justificativa para este trabalho reside na necessidade urgente de soluções inteligentes e sustentáveis para o gerenciamento de recursos hídricos, onde a engenharia elétrica desempenha um papel crucial na criação de tecnologias inovadoras que podem impactar positivamente a sociedade e o meio ambiente.

4. FUNDAMENTAÇÃO TEÓRICA

4.1. ARDUINO – ESP8266

O Arduino, uma plataforma de prototipagem eletrônica open-source, é essencial na engenharia elétrica, oferecendo uma abordagem prática para a implementação de sistemas embarcados (BANZI, 2011). Existem diversas placas Arduino, cada uma com características específicas, sendo escolhidas com base nos requisitos do projeto em questão. Uma placa amplamente utilizada, especialmente em projetos de automação residencial e IoT, é a D1 mini. Desenvolvido inicialmente para artistas, designers e entusiastas, o Arduino evoluiu para uma plataforma amplamente utilizada em projetos que vão desde aplicações educacionais até soluções industriais (BANZI, 2011).

A plataforma Arduino permite a criação de dispositivos de baixo custo e tecnologia de uso fácil (MONK 2016), sendo uma escolha popular devido à sua simplicidade de uso, comunidade ativa e ampla variedade de placas e módulos disponíveis (BANZI, 2011). O modelo D1 mini pro v 1.1.0, adotado neste projeto, incorpora características específicas, como conectividade Wi-Fi, que se alinham aos requisitos de monitoramento remoto e monitoramento via internet.

Para compreender o Arduino, é essencial explorar conceitos fundamentais relacionados à sua arquitetura e programação. A plataforma utiliza um microcontrolador AVR ou ARM, fornecendo entradas/saídas digitais e analógicas para interação com sensores, atuadores e outros dispositivos. A linguagem de programação simplificada baseada em C/C++, conhecida como Wiring, facilita a criação de códigos mesmo para iniciantes (MONK, 2016).

A D1 Mini Pro v 1.1.0, baseada no microcontrolador ESP8266, é reconhecida por sua conectividade Wi-Fi integrada (ESP8266EX DATASHEET). Esta característica é crucial para projetos que demandam monitoramento remoto, como o proposto, permitindo a transmissão de dados pela internet. A tensão de alimentação para a D1 mini pro v 1.1.0 varia entre 3.3V e 5V, o que oferece flexibilidade na escolha de fontes de energia (ESP8266EX DATASHEET).

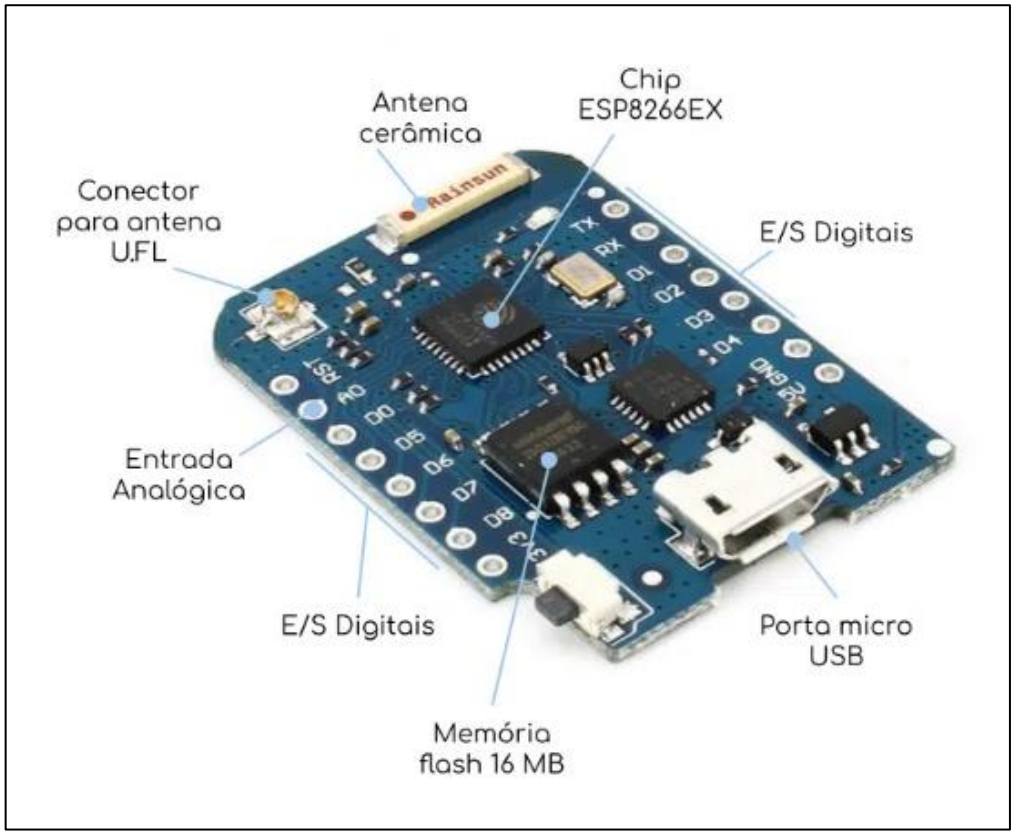
No contexto da D1 mini pro v 1.1.0, suas características específicas, como a presença de pinos de entrada/saída digital e analógica, são essenciais para a integração eficiente de sensores, como o HC-SR04. A versatilidade da placa, aliada à conectividade Wi-Fi, torna-a ideal para aplicações que demandam comunicação remota e automação residencial inteligente. Algumas das especificações técnicas da placa podem ser conferidas na Tabela 1. A Figura 1 ilustra a placa D1 Mini Pro v 1.1.0.

Tabela 1 – Especificações ESP8266 D1 Mini Pro

ESPECIFICAÇÕES TÉCNICAS	
Microcontrolador	ESP8266X
Conversor Serial para USB	CP2104
Tensão de Operação	3.3V
Tensão de Entrada	3.3 ou 5V
Pinos de Entrada/Saída Digitais	11
Pinos de PWM (Compartilhados com Entrada/Saída Digitais)	10
Pinos de Entrada Analógica	1 (10 bits), Máximo de entrada 3.2V
Corrente Contínua por Pino de Entrada/Saída	12mA (Máximo)
Portas Seriais de Hardware	1
Memória Flash	16 MBytes
RAM de Instruções	64 KBytes
RAM de Dados	96 KBytes
Velocidade do Clock	80MHz
Rede	IEEE 802.11 b/g/n WiFi
LED Embutido	Conectado ao pino 4
Conector USB	Micro-B Fêmea
Dimensões da Placa (PCB)	34.2 x 25.6mm
Peso	2.5g
Linguagens de programação	LUA, MicroPython e IDE Arduino (C/C++)

Fonte: https://www.makerhero.com/blog/conheca-a-wemos-d1-mini-pro-mais-funcionalidades-para-o-esp8266/

Figura 1 - ESP8266 D1 Mini Pro V 1.1.0



Fonte: https://www.makerhero.com/blog/conheca-a-wemos-d1-mini-pro-mais-funcionalidades-para-o-esp8266/

4.2. SENSORES ULTRASSÔNICOS

Sensores ultrassônicos, em particular o Módulo HC-SR04 de Medição Ultrassônica da Figura 2, desempenham um papel crucial em sistemas de medição, oferecendo uma solução precisa para a determinação de distâncias sem contato. Este módulo, referenciado pelos princípios delineados por BENTLEY, J. P., em "Principles of Measurement Systems" (2005), destaca-se na engenharia elétrica por suas características técnicas notáveis.

Figura 2 – Sensor Ultrassônico HC-SR04

Fonte: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

Características do Produto: O Ultrasonic Ranging Module HC-SR04 oferece uma função de medição não contactante com alcance de 2 cm a 400 cm, com uma margem de erro de aproximadamente 3 mm. Este módulo é composto por transmissores ultrassônicos, receptor e circuito de controle, operando sob um princípio de trabalho que envolve um sinal de disparo de pelo menos 10 μs (HC-SR04 DATASHEET).

O módulo emite automaticamente oito pulsos de 40 kHz e detecta os pulsos de retorno. Se um sinal de retorno é detectado, o tempo de duração do sinal de saída IO alto é proporcional à distância medida (HC-SR04 DATASHEET). Sua conexão é direta, exigindo fios para alimentação, entrada de pulso de disparo, saída de pulso de eco e aterramento. Os parâmetros elétricos, incluindo a faixa de operação, ângulo de medição e frequência de operação, tornam o HC-SR04 versátil para aplicações em projetos eletrônicos, como mostram a Tabela 2 e a Tabela 3.

Tabela 2 – Conectores do Sensor HC-SR04

PINOS DE CONEXÃO	
Alimentação	5V
Entrada de Pulso de Disparo	(Trigger)
Saída de Pulso de Eco	(Echo)
Terra	0V

Fonte: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

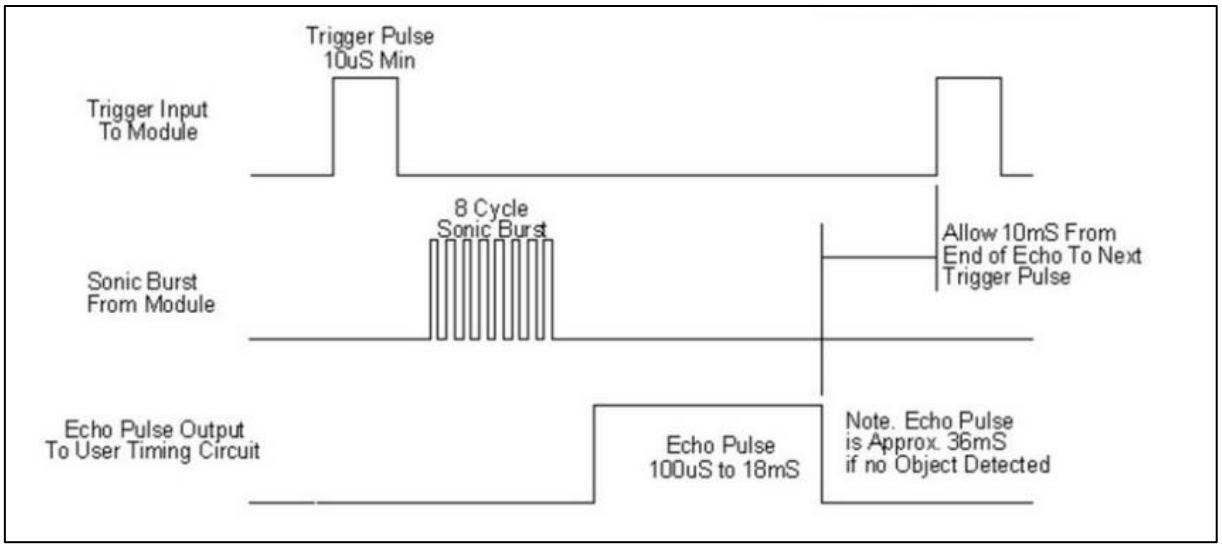
Tabela 3 - Especificações Sensor HC-SR04

PARÂMETROS ELÉTRICOS	
Tensão de Operação	DC 5 V
Corrente de Operação	15 mA
Frequência de Operação	40 Hz
Alcance Máximo	4 m
Alcance Mínimo	2 cm
Ângulo de Medição	15 graus
Sinal de Entrada de Disparo	Pulso TTL de 10 µs
Sinal de Saída de Eco	Sinal de nível TTL, proporcional à distância
Dimensões	45mm x 20mm x 15mm

Fonte: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

Diagrama de Temporização: O funcionamento do módulo é representado por um diagrama de temporização, onde um breve pulso de 10 µs na entrada de disparo inicia a medição. O módulo emite uma salva de oito ciclos de ultrassom a 40 kHz e eleva seu sinal de eco. A largura do pulso de eco é proporcional à distância medida. A fórmula para calcular a distância é: uS / 58 = centímetros ou uS / 148 = polegadas (HC-SR04 DATASHEET). Conforme mostra a Figura 3.

Figura 3 - Diagrama de Temporização HC-SR04



Fonte: https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf

Algumas precauções dever ser tomadas, como a conexão direta à energia elétrica, que não é recomendada. Caso seja necessária, o terminal GND deve ser conectado primeiro para evitar interferências no funcionamento normal.

Ao testar objetos, a área de alcance não deve ser inferior a 0,5 metros quadrados, e a superfície deve ser o mais lisa possível para evitar impactos nos resultados da medição.

Esta abordagem abrangente não apenas incorpora as características técnicas distintivas do Módulo HC-SR04, mas também se baseia nos princípios estabelecidos por Bentley, oferecendo uma perspectiva robusta para sua aplicação em projetos de engenharia elétrica.

4.3. REGULADORES DE TENSÃO

Ao contrário das fontes de alimentação lineares, nas fontes de alimentação comutadas, a transformação da tensão contínua de um nível para outro é realizada utilizando circuitos conversores DC-DC (MOHAN, 2002). Os reguladores chaveados convertem uma tensão CC, em geral não regulada, em uma tensão CC regulada de saída (RASHID,2014). Essa regulação normalmente é conseguida por modulação em largura de pulsos a uma frequência fixa, sendo o dispositivo de chaveamento na maioria das vezes um BJT, MOSFET ou IGBT de potência, e são fornecidos comercialmente como circuitos integrados (RASHID, 2014).

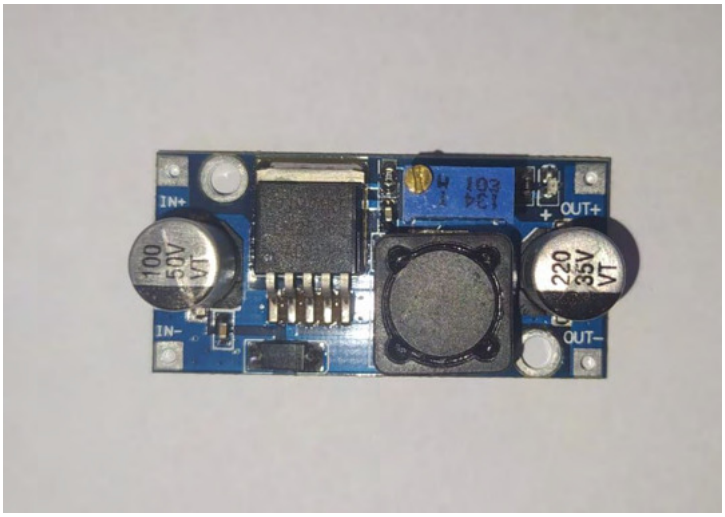
4.3.1. Regulador Buck

Em um regulador buck, a tensão média de saída Va é menor que a tensão de entrada Vs – daí, o nome buck, um regulador muito popular (RASHID, 2014). Os reguladores buck requerem apenas um transistor, são simples e têm eficiência elevada, maior que 90%. O *di/dt* da corrente de carga é limitado pelo indutor L. Entretanto, a corrente de entrada é descontínua e um filtro de alisamento de entrada normalmente é requerido. Ele fornece uma polaridade da tensão de saída e a corrente de saída é unidirecional. Ele requer um circuito de proteção em caso de possível curto-circuito através do caminho do díodo (RASHID, 2014).

O regulador de tensão LM2596 pertence à série de reguladores monolíticos integrados. Este dispositivo oferece todas as funções ativas necessárias para um regulador de comutação step-down (buck), capaz de fornecer uma carga de até 3 A com excelente regulação de linha e

carga. Disponível em diversas voltagens de saída fixas, como 3.3 V, 5 V, 12 V, e também em uma versão ajustável (LM2596 DATASHEET). A Figura 4 ilustra o regulador LM2596.

Figura 4 – Regulador LM2596



Fonte: Autoria Própria

Com a necessidade mínima de componentes externos, esses reguladores são de fácil utilização, contando com compensação de frequência interna e um oscilador de frequência fixa. Operando a uma frequência de comutação de 150 kHz, permitem o uso de componentes de filtro menores em comparação com reguladores de comutação de frequência mais baixa (LM2596 DATASHEET). Sua versatilidade é evidente em aplicações que exigem uma tensão de saída consistente, tornando-se uma escolha valiosa em diversos cenários de engenharia elétrica.

Ao integrar esses conhecimentos em projetos de engenharia elétrica, é possível desenvolver sistemas robustos e eficientes, essenciais para a operação confiável de dispositivos eletrônicos em uma variedade de contextos.

4.4. BATERIAS E OTIMIZAÇÃO DE ENERGIA

A integração de baterias em dispositivos eletrônicos é uma prática comum, conferindo-lhes mobilidade e independência de fontes de energia fixas. A compreensão dos princípios fundamentais das baterias é crucial para garantir uma fonte de energia confiável e duradoura.

O conceito básico define que uma bateria é um dispositivo que converte a energia química contida em seus materiais ativos diretamente em energia elétrica por meio de uma reação eletroquímica de oxidação-redução (redox) (LINDEN, 2013). No caso de um sistema recarregável, a bateria é recarregada por meio de uma reversão desse processo (LINDEN, 2013). Baterias portáteis compartilham os mesmos princípios gerais que as baterias primárias e outras fontes de energia eletroquímicas, como células a combustível. Elas consistem em dois eletrodos conectados eletricamente a materiais ativos e imersos em um eletrólito com um separador poroso colocado entre eles para evitar contato elétrico, mas permitindo o fluxo iônico (BARKUSOV 2013).

A seleção adequada de baterias é essencial para atender às demandas específicas de um projeto eletrônico. Fatores como capacidade química, impedância da bateria, capacidade de uso em determinada temperatura e taxa de descarga, capacidade de potência, degradação em armazenamento e ciclo, e taxa de auto descarregamento são determinantes na escolha da tecnologia de bateria mais apropriada (BARKUSOV 2013). Conhecer esses aspectos não apenas assegura uma operação estável, mas também impacta diretamente na eficiência e na autonomia do dispositivo.

Para aprofundar ainda mais o entendimento sobre o gerenciamento eficaz de baterias, fatores como carga e descarga eficientes, além de estratégias de economia de energia são de extrema importância. A principal questão relacionada à funcionalidade da bateria em uma aplicação portátil é "Quanto tempo ela vai durar?" Isso é determinado pela quantidade de materiais ativos, sua capacidade específica e suas características de voltagem (BARKUSOV 2013). Dois métodos de carga demonstrados por Barkusov exemplificam como as estratégias pode influenciar na vida útil de uma bateria. Em um sistema de carregamento linear de uma bateria de lítio, a principal desvantagem é sua menor eficiência de carga, que tem cerca de 72% de eficiência ao carregar uma única bateria de célula Li-ion a partir de uma fonte de energia de 5V. O próprio carregador dissipa 28% da potência de entrada (BARKUSOV 2013). Enquanto o conversor síncrono de comutação tem uma eficiência de conversão de energia superior a um regulador linear porque o conversor opera no modo completamente ligado ou desligado, e não em uma operação linear, podendo fornecer uma corrente de carga média 40% maior do que um carregador linear. Portanto, o carregador de comutação pode carregar a bateria mais rápido do que um carregador linear. Isso ocorre simplesmente porque o carregador de comutação tem menor consumo de energia e mais potência disponível para carregar a bateria se a potência de entrada for fixa (BARKUSOV 2013).

Em paralelo, a otimização do consumo de energia é uma consideração crítica, especialmente em dispositivos alimentados por bateria. Algoritmos eficientes, modos de baixo consumo e a implementação de práticas que favorecem a eficiência energética são fundamentais na hora de se elaborar um projeto que demande tal eficiência. O modo Deep Sleep do Arduino é um ótimo exemplo, e foi implementado nesse projeto, onde o Deep Sleep permitirá que o dispositivo desligue uma variedade de módulos internos para economizar a maior parte do consumo de energia (ARDUINO DOCUMENTATION). O modo de "Deep Sleep" pode ser configurado com um temporizador para despertar após um período definido de tempo ter transcorrido (ARDUINO DOCUMENTATION).

A implementação prática desses conceitos em projetos de engenharia elétrica requer uma abordagem integrada. O designer deve considerar não apenas a escolha da bateria mais apropriada para as necessidades do sistema, mas também desenvolver algoritmos e estratégias de gerenciamento de energia que otimizem o uso dos recursos disponíveis. Essa abordagem integrada não apenas maximiza a vida útil da bateria, mas também contribui para a eficiência global do dispositivo, resultando em soluções eletrônicas mais sustentáveis e eficazes.

4.5. COMUNICAÇÃO SEM FIO (WI-FI)

Se os usuários precisarem estar conectados a uma rede por cabos físicos, sua movimentação é drasticamente reduzida. A conectividade sem fio, no entanto, não impõe tal restrição e permite uma maior liberdade de movimento por parte do usuário de rede. Como resultado, as tecnologias sem fio estão invadindo o domínio tradicional das redes "fixas" ou "com fio" (GAST, 2005). Isso tornou-se uma realidade na sociedade desde 1901, quando ocorreu a primeira transmissão remota de dados digitais sem fio (TANENBAUM; WOODHULL, 2000). Essa transmissão pioneira foi realizada por Guglielmo Marconi, um físico italiano, por meio de um telégrafo sem fio que enviava informações em Código Morse de um navio para a costa (TANENBAUM; WOODHULL, 2000). Desde então, a tecnologia de transmissão sem fio tem evoluído continuamente, chegando ao estágio atual, onde uma enorme quantidade de bits trafega pelas "vias aéreas", conectando todo o mundo.

Atualmente, uma das "famílias" de redes mais presentes no cotidiano das pessoas é a Wi-Fi (acrônimo em inglês para Wireless Fidelity), e ela tem ganhado cada vez mais destaque entre os dispositivos wireless (MA; ZHOU; WANG, 2019). Frequentemente, é preferida em comparação com outras tecnologias, como o Bluetooth. As redes Wi-Fi, assim como todas as outras redes de protocolo público, seguem uma padronização definida pelo IEEE (Institute of

Electrical and Electronics Engineers), garantindo a compatibilidade entre todos os dispositivos capazes de se conectar a esse tipo de rede. No caso do Wi-Fi, a padronização ocorre por meio da família de protocolos 802.11 (ANTONIO, 2014).

4.5.1. Integração de Wi-Fi em Sistemas Embarcados

A incorporação de Wi-Fi em sistemas embarcados, como o mencionado sensor de nível de água, exige uma compreensão aprofundada de como integrar essa tecnologia de maneira eficiente. Fred Eady (2005) define que, o principal critério para se escolher um microcontrolador para um projeto, é basicamente escolher um que seja de fácil acesso, popular e que atenda a todas as especificações necessárias, o mesmo critério vale ao escolher o dispositivo de comunicação 802.11.

4.6. SISTEMAS EMBARCADOS

Sistemas embarcados desempenham um papel central na engenharia elétrica, sendo componentes-chave em uma ampla gama de dispositivos, desde eletrodomésticos até sistemas de controle industrial.

Um sistema embarcado refere-se a sistemas eletrônicos que, mesmo utilizando um microprocessador para realizar processamento, exercem apenas uma função específica após serem programados e incorporados ao seu sistema final (ALMEIDA; MORAES; SERAPHIM, 2016). Com o avanço da microeletrônica, esses sistemas tornaram-se cada vez mais comuns em nosso cotidiano. Equipamentos complexos, como televisores e assistentes eletrônicos, assim como sistemas que originalmente eram totalmente eletromecânicos, agora incorporam microeletrônica embarcada, como é o caso de carros com assistências automatizadas, incluindo controle de tração, estabilidade, frenagem automática de emergência e centrais multimídia.

É crucial ressaltar que, embora dispositivos com sistemas embarcados possam ser considerados "aparelhos" de propósito específico, isso não os classifica necessariamente como dispositivos simples em termos de propósito ou construção. Sistemas de controle para drones autônomos, por exemplo, demandam tecnologia embarcada sofisticada e uma construção complexa para realizar tarefas extremamente delicadas.

Além dessa característica distintiva, há outras diferenças entre computadores de propósito geral e sistemas embarcados, conforme destacado por Almeida, Moraes e Seraphim (2016). Uma delas é a menor quantidade de recursos computacionais, como a capacidade de

processamento de microprocessadores ou microcontroladores, e a menor quantidade de memória disponível. A particularidade de cada microcontrolador e sistema embarcado também resulta no fato de que um mesmo código fonte aplicado a diferentes sistemas ou microcontroladores provavelmente não será capaz de executar corretamente seu propósito.

4.7. PROCESSAMENTO DE SINAIS DIGITAIS

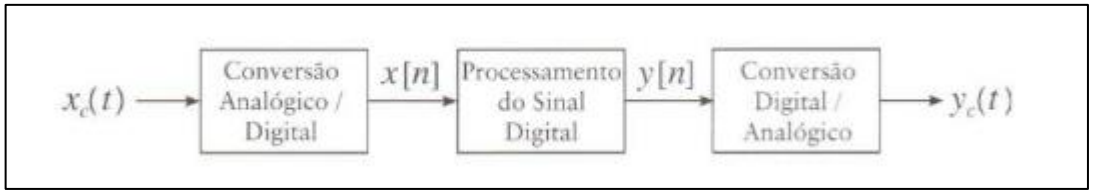
O processamento digital de sinais (PDS) é uma disciplina central na engenharia elétrica, desempenhando um papel crucial em diversas aplicações, Nalon (2009) define que a disciplina estuda como os sinais se relacionam e, principalmente, como manipular os sinais de forma a se obter um resultado desejado. Nesse contexto, o Arduino, uma plataforma de prototipagem eletrônica amplamente utilizada, também desempenha um papel significativo no processamento de sinais digitais para aplicações diversas.

4.7.1. Princípios Básicos de Processamento de Sinais Digitais

Por sua própria natureza os sinais são analógicos, e são representados por uma função contínua, o que impede que sejam tratados adequadamente por um processador digital (NALON, 2009).

Um sinal precisa ser discretizado para que isso seja possível, obtendo uma sequência de amostras que representarão o sinal, esse processo é chamado de amostragem, e um sistema de processamento digital de sinais consiste em amostrar o sinal realizar o processamento, e reconstruir o sinal processado para obter o resultado analógico (NALON, 2009), como mostra a Figura 5.

Figura 5 – Sistema de Processamento Digital



Fonte: NALON 2009

4.7.2. Filtragem Digital e Transformada de Fourier

Fundamental para o tratamento de sinais, tanto discretos quanto contínuos, é a análise em frequência, feita por meio da transformada de Fourier. Um sinal qualquer pode ser decomposto em ondas de várias frequências diferentes, representadas por funções senoidais. Pela análise das frequências e das amplitudes dessas funções é possível obter uma gama enorme de informações a respeito dos sinais analisados, e também projetar de maneira adequada o processamento a ser realizado (NALON, 2009).

4.7.3. Processamento de Sinais com Microcontroladores

Microcontroladores, reconhecidos por sua versatilidade e facilidade de uso, são frequentemente utilizados no processamento de sinais digitais. Projetos que abrangem a captura, processamento e exibição de dados analógicos podem ser eficazmente realizados por meio de microcontroladores. A comunidade de microcontroladores oferece amplo suporte, bibliotecas e exemplos práticos para simplificar a implementação de técnicas de PDS em projetos do mundo real.

4.8. PROTOCOLO HTTP

O Protocolo de Transferência de Hipertexto (HTTP) é um protocolo de comunicação de nível de aplicação projetado para sistemas de informação hipermídia distribuídos e colaborativos (FIELDING et al., 1999). Trata-se de um protocolo genérico com ampla aplicabilidade, sendo utilizado em servidores e sistemas de gerenciamento de objetos distribuídos, estendendo-se por meio de métodos de solicitação, códigos de erro e cabeçalhos. Uma característica distintiva do HTTP é a capacidade de tipificar e negociar representações de dados, permitindo que sistemas sejam desenvolvidos independentemente dos dados que estão sendo transferidos. Desde 1990, o HTTP tem desempenhado um papel crucial na iniciativa global de informações da World Wide Web (WWW) (FIELDING et al., 1999).

Cada requisição HTTP é caracterizada por um método definido, que pode ser um dos seguintes:

- OPTIONS;
- GET;
- HEAD;

- POST;
- PUT;
- DELETE;
- TRACE;
- CONNECT.

4.9. HTML

Segundo Raggett et al. (1999), a publicação de informações em distribuições globais demanda o uso de uma linguagem universalmente compreendida, uma espécie de língua materna que todos os computadores têm o potencial de entender. A linguagem de publicação empregada pela World Wide Web é a HyperText Markup Language (HTML), que proporciona aos usuários meios para:

1. Publicar documentos on-line contendo títulos, textos, tabelas, listas, fotos, entre outros elementos;
2. Recuperar informações on-line por meio de links de hipertexto, com um simples clique de botão;
3. Desenvolver formulários para a realização de transações com serviços remotos, facilitando a busca de informações, a realização de reservas, a encomenda de produtos, entre outros;
4. Inserir planilhas, videoclipes, clipes de som e outros aplicativos diretamente em seus documentos.

4.10. CSS

Segundo Silva (2007), CSS, ou Cascading Style Sheet, é um documento que delineia as regras de formatação ou estilos a serem aplicados aos elementos estruturais de marcação, como o HTML. O propósito fundamental do CSS, conforme enfatizado pelo autor, é separar completamente as declarações de formatação do HTML. Isso implica remover do HTML qualquer aspecto relacionado à formatação do documento. As principais vantagens do uso de CSS incluem:

1. **Controle Integral da Apresentação do Site:** Através de um arquivo central, proporciona controle total sobre a apresentação do site.

2. **Agilidade na Manutenção e Reconstrução do Estilo do Site:** Facilita a manutenção e a reconstrução ágeis do estilo do site.
3. **Saída para Diferentes Tipos de Mídia:** Permite a saída para diversos dispositivos (celulares, tablets, computadores, etc.) por meio de uma única versão de HTML.
4. **Redução do Tempo de Carregamento dos Documentos na Web:** Contribui para a redução do tempo de carregamento de documentos na web.
5. **Aumento Considerável na Portabilidade dos Documentos Web:** Proporciona um aumento significativo na portabilidade dos documentos web.

Em síntese, conforme enfatizado pelo autor, a regra é clara: HTML é utilizado para estruturar, enquanto o CSS é empregado para apresentar.

4.11. JAVASCRIPT

JavaScript é uma linguagem de programação fundamentalmente criada para definir o comportamento de páginas na web, desempenhando um papel preponderante na construção dinâmica e interativa de conteúdos online (Flanagan, 2011). Sua adoção é praticamente universal em sites modernos, e todos os navegadores contemporâneos incorporam intérpretes específicos para essa linguagem. Dessa forma, JavaScript se destaca como a linguagem de programação mais amplamente difundida na história.

Cada linguagem de programação necessita de uma biblioteca padrão ou API (Interface de Programação de Aplicações) de funções para realizar tarefas fundamentais, como entrada e saída básicas. No caso do JavaScript, uma API mínima é estabelecida para manipulação de texto, matrizes, datas e expressões regulares. Entretanto, é importante notar que ela não inclui funcionalidades de entrada ou saída, deixando essa responsabilidade para o "ambiente host" no qual o JavaScript está incorporado, frequentemente sendo o navegador web (Flanagan, 2011).

4.11.1. Highcharts

Highcharts é uma poderosa biblioteca JavaScript para a criação de gráficos interativos em páginas web. Ela oferece uma ampla gama de opções de personalização e é amplamente utilizada para visualizar dados de maneira envolvente e informativa. Sua facilidade de integração e recursos avançados fazem dela uma escolha popular para desenvolvedores que buscam incorporar gráficos dinâmicos em suas aplicações web.

4.11.2. FusionCharts

FusionCharts, por sua vez, é outra biblioteca JavaScript focada na criação de gráficos, proporcionando uma variedade de tipos de gráficos e recursos visuais. Seu diferencial muitas vezes está em gráficos específicos, como medidores de cilindro, o que pode ser útil em determinados contextos. Assim como o Highcharts, o FusionCharts é projetado para ser fácil de usar e altamente personalizável, oferecendo uma solução robusta para representação visual de dados em aplicações web.

5. METODOLOGIA

A metodologia adotada para a elaboração e execução deste projeto originou-se da proposta de desenvolver um sistema de medição e monitoramento do nível de água, utilizando componentes de fácil acesso e garantindo precisão e funcionalidade.

Para o presente projeto de monitoramento e medição de nível de água foram utilizados os materiais presentes na Tabela 4.

Tabela 4 - Materiais Utilizados

Quantidade	Material
1	Protoboard
1	ESP8266 Wemos D1 Mini Pro
1	ESP8266 D1 Mini
1	Sensor Ultrassônico HC-SR04
1	Resistor 1k ohm
1	Resistor 500ohm
1	Led Vermelho
1	Bateria GSP753868X 2100mAh
1	Conector JST
1	Regulador de Tensão Buck LM2596
1	Micro Interruptor
1	Fios diversos
1	Placa PCB

Fonte: Autoria Própria

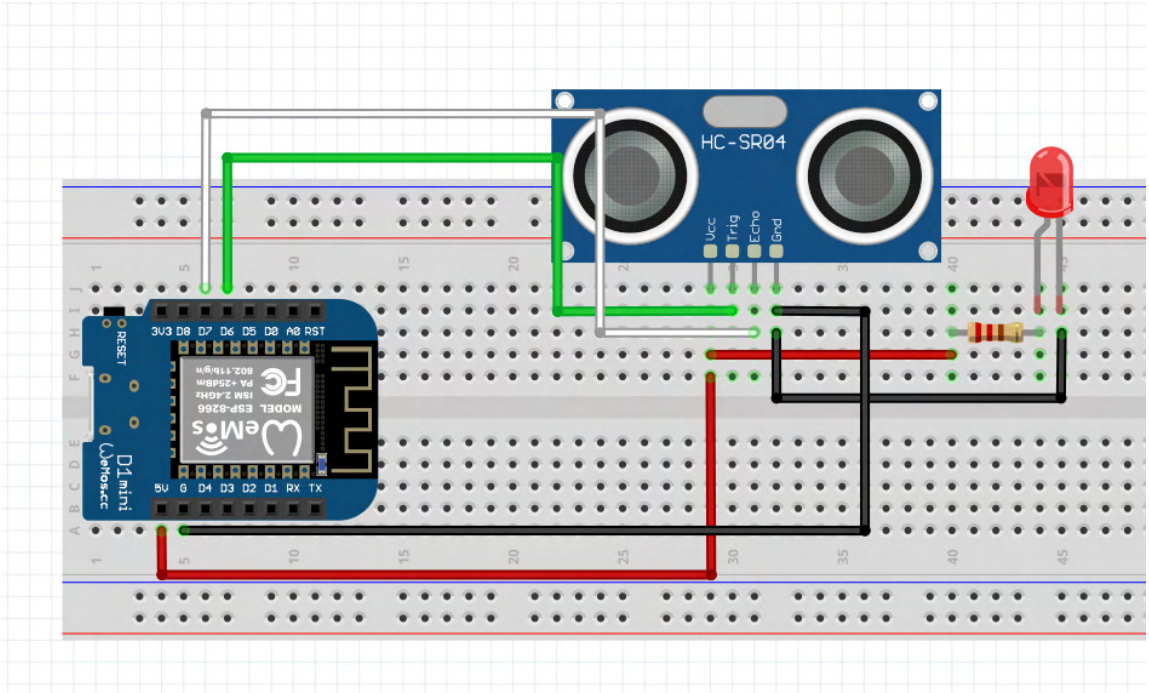
A concepção inicial envolveu a criação de um protótipo de sistema, empregando um sensor de ultrassom na plataforma Arduino, juntamente com uma placa que possibilitasse a conectividade com a internet para o envio de dados ao usuário, seja por meio de um computador ou celular. Com as dimensões da caixa d'água conhecidas, o sistema seria instalado internamente, permitindo que o sensor calculasse a altura do nível da água e suas variações. Realizando leituras em intervalos determinados, seria possível determinar o nível de água em tempo real e gerar um relatório diário de consumo. Desta forma, todos os objetivos propostos seriam alcançados, otimizando o consumo, reduzindo desperdícios, minimizando os impactos ambientais e aplicando conceitos da Internet das Coisas (IoT) na engenharia moderna, com foco em automação residencial. Além disso, contribuiria para a pesquisa e desenvolvimento nesta área, resultando no desenvolvimento de um protótipo acessível para aqueles que necessitam desse tipo de sistema.

Na etapa seguinte, foi decidida a plataforma exata a ser utilizada, e para atender aos requisitos propostos, escolheu-se a placa de processamento WEMOS D1 MINI PRO V1.1.0. Essa placa oferece a pinagem necessária para a conexão de todos os componentes do circuito e possui conectividade integrada com a internet. A WEMOS D1 MINI PRO V1.1.0 pode ser programada para gerar uma página da web em uma rede, permitindo visualizar os dados coletados e processados pelo Arduino por meio de um IP gerado por ela.

O sensor escolhido foi o HC-SR04, um sensor ultrassônico compatível diretamente com as placas Arduino, atendendo perfeitamente aos objetivos do projeto.

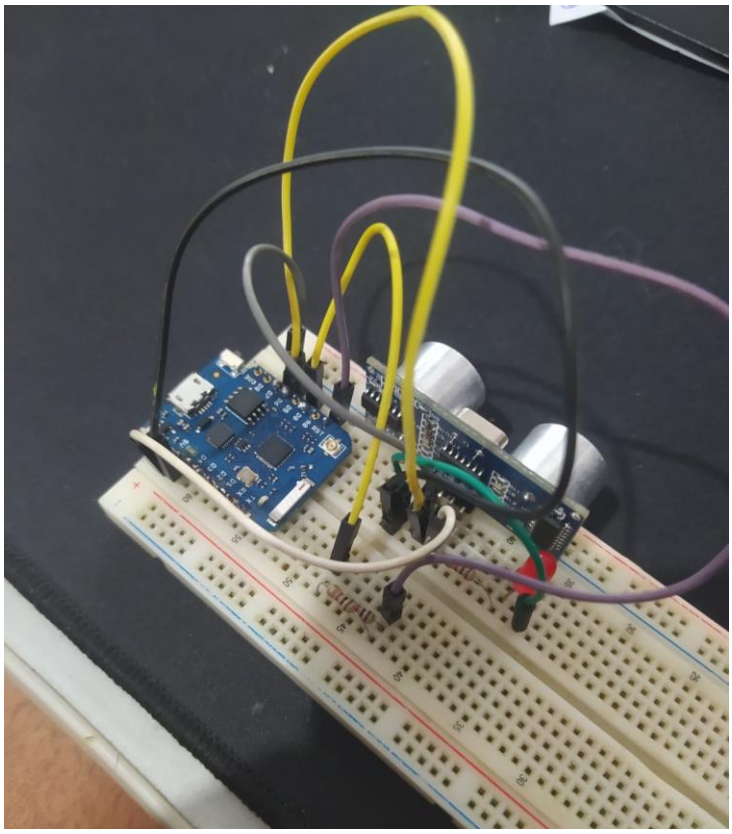
Em seguida, procedeu-se à montagem do circuito, sendo a conexão do sensor com a placa ESP8266 direta e simples, conforme o esquema abaixo na Figura 6. A foto real do circuito nessa etapa do projeto é ilustrada na Figura 7

Figura 6 – Esquema Inicial do Projeto



Fonte: Autoria Própria (Software Fritzing)

Figura 7 – Foto real do circuito

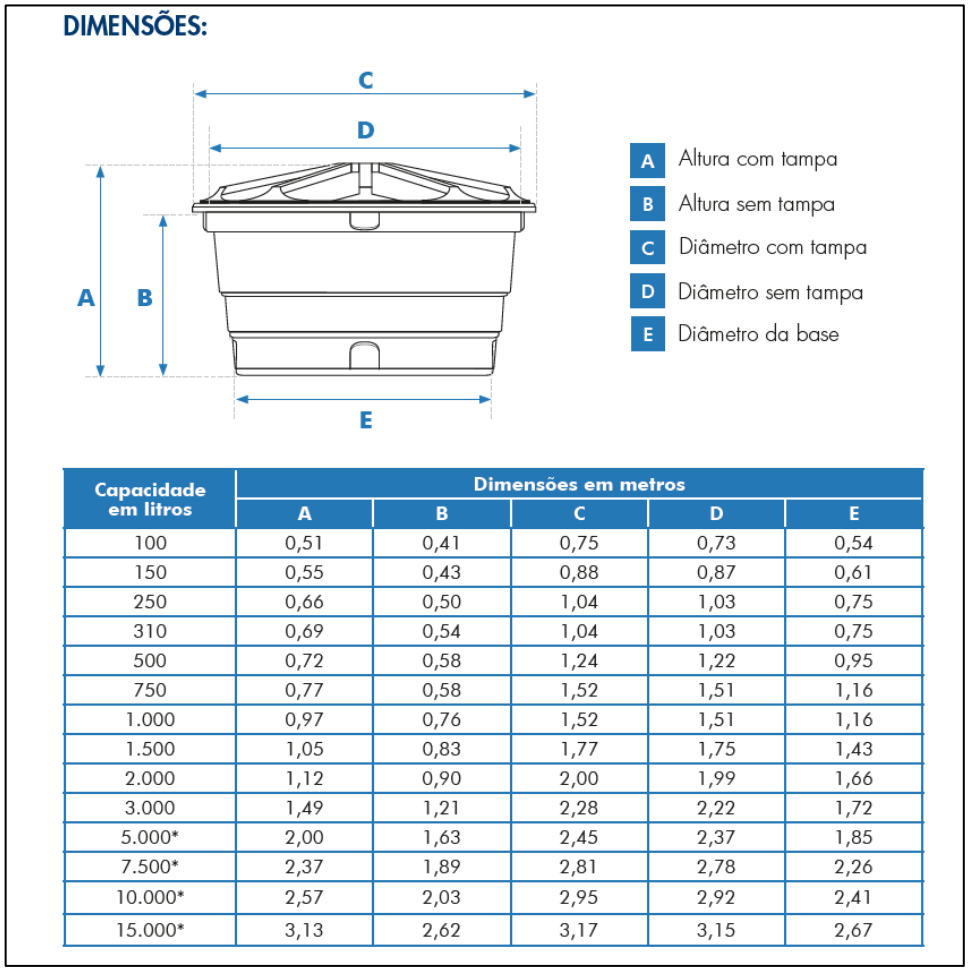


Fonte: Autoria Própria

A própria IDE do Arduino já incorpora uma biblioteca de suporte para o sensor HC-SR04, que oferece um modelo de código para testar o sensor. Os primeiros testes foram realizados e comparados com medições de réguas e trenas para verificar a calibração do sensor. O passo subsequente foi definir e aprimorar o código para simular o cálculo da distância até o espelho d'água, permitindo assim determinar o volume da caixa.

O cálculo do volume da caixa d'água é bastante simples, sendo utilizado como referência o documento de especificações das caixas d'água da FORTLEV. Esta marca é amplamente presente nos lares brasileiros, sendo escolhida como base de dados para o projeto. As dimensões das caixas d'água da FORTLEV podem ser conferidas na Figura 8.

Figura 8 – Tabela de dimensões de reservatórios FORTLEV



Capacidade em litros	Dimensões em metros				
	A	B	C	D	E
100	0,51	0,41	0,75	0,73	0,54
150	0,55	0,43	0,88	0,87	0,61
250	0,66	0,50	1,04	1,03	0,75
310	0,69	0,54	1,04	1,03	0,75
500	0,72	0,58	1,24	1,22	0,95
750	0,77	0,58	1,52	1,51	1,16
1.000	0,97	0,76	1,52	1,51	1,16
1.500	1,05	0,83	1,77	1,75	1,43
2.000	1,12	0,90	2,00	1,99	1,66
3.000	1,49	1,21	2,28	2,22	1,72
5.000*	2,00	1,63	2,45	2,37	1,85
7.500*	2,37	1,89	2,81	2,78	2,26
10.000*	2,57	2,03	2,95	2,92	2,41
15.000*	3,13	2,62	3,17	3,15	2,67

Fonte: Catálogo Técnico Caixa D'água Fortlev

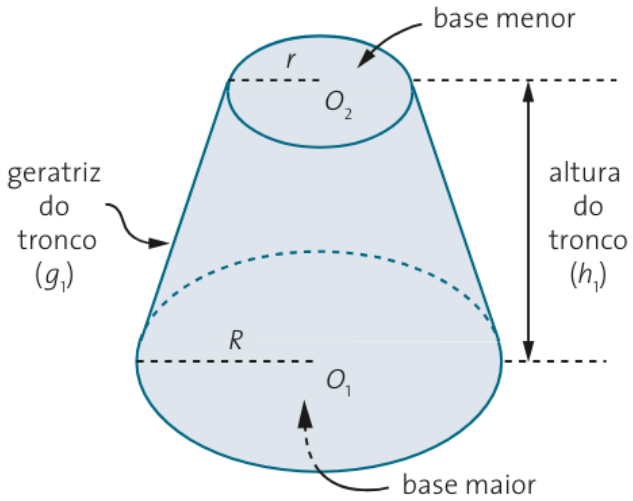
Para os propósitos dos testes, optou-se por utilizar as dimensões da caixa d'água de 500 litros, pois a caixa d'água real que viria a ser utilizadas para os testes práticos é do mesmo

modelo e capacidade. A figura geométrica que representa a caixa d'água é o tronco de cone reto, podemos observar a semelhança visual na Figura 9, Dante (2006) define que o cálculo para o volume do tronco de cone reto é dado por:

$$Volume\ V = \frac{\pi h}{3}\ (R^2 + Rr + r^2)$$

Sendo, h a altura do cone, R o raio da base maior e r o raio da base menor.

Figura 9 – Tronco de cone reto



Fonte: Dante 2006

Ao inserir as dimensões, como altura com tampa, diâmetro da base, diâmetro sem tampa e altura sem tampa, em metros, torna-se viável calcular o volume aproximado da caixa. A altura com tampa representa a altura onde o sensor será instalado, o diâmetro sem tampa é o diâmetro correspondente ao nível máximo de água, e a altura sem tampa também é considerada. Isso ocorre porque a altura do nível máximo é influenciada pela altura da instalação do cano de entrada. Nos testes realizados em uma caixa d'água, a diferença de altura em relação ao nível máximo foi de 5 cm. O cálculo do volume da caixa d'água é determinado por:

Altura com tampa = A;

Altura do nível = *h*;

Diâmetro sem tampa = *D*;

Diâmetro da base = *d*;

Sendo assim,

$$\text{Altura do nível } h = 0{,}58 - 0{,}5 = 0{,}53$$

$$D = 1{,}22$$

$$A = 0{,}72$$

$$d = 0{,}95$$

Determinando, assim

$$altura\ do\ cone\ h = 0{,}53$$

$$Raio\ da\ base\ maior\ R = \frac{D}{2} = \frac{1{,}22}{2} = 0{,}61$$

$$raio\ da\ base\ menor\ r = \frac{d}{2} = \frac{9{,}5}{2} = 0{,}475$$

Logo,

$$V = \frac{\pi h}{3}\,(R^2 + Rr + r^2) = \frac{\pi\,.\,(0{,}53)}{3}\,(\,0{,}61^2 + (0{,}61)\,.\,(0{,}475) + 0{,}475^2)$$

$$V = \frac{1{,}6642}{3}\,(0{,}3721 + 0{,}28975 + 0{,}22563) = 0{,}555\,.\,(0{,}88748) = 0{,}49255\ m^3$$

Tendo esse valor em Litros,

$$V_L = V_{cm^3}(1000) \cong 492{,}55\ L$$

Para determinar a leitura da caixa d´água em tempo real, basta considerar uma variável desconhecida *x*, que é a diferença de altura de instalação do sensor, para a altura da leitura do sensor naquele instante.

Digamos que o leitor realizou uma captura de 25cm de distância até o espelho d'água, logo temos:

$$altura\ h = altura\ do\ sensor\ A - altura\ da\ leitura\ x$$

$$h = A - x$$

$$h = 0{,}72 - 0{,}25 = 0{,}47$$

$$V = 0{,}4366\ m^3$$

$$V = 436{,}6\ L$$

Então, naquele momento a leitura da caixa d'água detectou que o volume atual da caixa é de 436,6 litros.

Convertendo esse cálculo em código de programação, foi possível determinar a altura do nível da água e o volume da caixa d'água em tempo real. Feito da seguinte maneira:

As variáveis das características da caixa e leitura foram definidas:

```
float raio1 = 0.60;         // Raio interno da caixa d'água em metros
float raio2 = 0.475;        // Raio externo da caixa d'água em metros
float altura2 = 0.72;       // Altura total da caixa d'água em metros
float altura1;              // Altura calculada com base na distância medida

float distancia;            // Armazena a distância medida pelo sensor ultrassônico em centímetros
float distancia1;           // Armazena a distância convertida de centímetros para metros
String result;              // Armazena a altura (distância) como uma string
String result2;             // Armazena o volume da caixa d'água como uma string
float volume;               // Volume calculado da caixa d'água em litros
int capacidade = 500;       // Capacidade máxima da caixa d'água em litros
```

A biblioteca do sensor, definida <*Ultrasonic.h*> possui a função *Ranging*, que retorna o valor da distância lida em centímetros em forma de *float*. Assim, foi criada a função hcsr04 que faz a leitura do sensor e armazena na variável *distancia*, e em seguida calcula o volume lido na caixa d'água.

```
void hcsr04() {

  digitalWrite(D6, LOW);                    //SETA O PINO 6 COM UM PULSO BAIXO "LOW"
  delayMicroseconds(2);                     //INTERVALO DE 2 MICROSSEGUNDOS
  digitalWrite(D6, HIGH);                   //SETA O PINO 6 COM PULSO ALTO "HIGH"
  delayMicroseconds(10);                    //INTERVALO DE 10 MICROSSEGUNDOS
  digitalWrite(D6, LOW);                    //SETA O PINO 6 COM PULSO BAIXO "LOW" NOVAMENTE

  // Função Ranging converte o tempo de resposta do echo em centímetros e armazena na variável "distancia"

  distancia = ultrasonic.Ranging(CM);
  result3 = String(distancia);
  distancia1 = distancia / 100;
  altura1 = altura2 - distancia1;
  volume = ((3.14 * altura1) / 3) * ((raio1 * raio1) + (raio1 * raio2) + (raio2 * raio2));
  volume = volume * 1000;
  result = String(altura2);
  result2 = String(volume);
```

O próximo passo consistiu em configurar a ESP8266 para gerar uma página em um servidor web, permitindo visualizar as informações coletadas.

```
// CRIA O WEBSERVER

AsyncWebServer server(80);


 // CONEXÃO DO WIFI

 WiFi.begin(rede, senha);
 client.setTrustAnchors(&cert);
 while (WiFi.status() != WL_CONNECTED)            //Aguarda a conexão
 {
   Serial.print("Estabelecendo conexão com ");
   Serial.println(WiFi.SSID());                   //Imprime o nome da Rede
   delay(500);
 }
 Serial.print("Conectado a rede! Endereco IP ESP -> ");
 Serial.println(WiFi.localIP());                  //Imprime o IP local da ESP, que exibirá a página WEB
```

O Arduino também disponibiliza uma biblioteca de comunicação web, e após a definição das funções apropriadas, tornou-se possível enviar os dados lidos, formatados como strings, para a página web e observar a leitura do volume da caixa.

Após estabelecer a leitura dos dados e o envio para a página web, deu-se início ao processo de programação em HTML. A primeira etapa envolveu a pesquisa de uma abordagem para associar a página web a um arquivo separado, visando a separação do código de processamento do Arduino do código HTML. A solução encontrada foi a utilização do sistema de arquivos SPIFFS, que possibilita o acesso à memória FLASH da ESP8266, permitindo o armazenamento de informações sem a necessidade de um cartão de memória SD.

O código da página web, composto por elementos de HTML, CSS e Javascript, foi compilado para a memória do Arduino. Para realizar esse processo, foi necessário criar uma pasta denominada "data" na mesma localização do arquivo com o código do Arduino. Dentro dessa pasta, o arquivo desejado para gravação na memória FLASH foi inserido. No código do Arduino, uma chamada para carregar os dados do arquivo, denominado "index", foi executada.

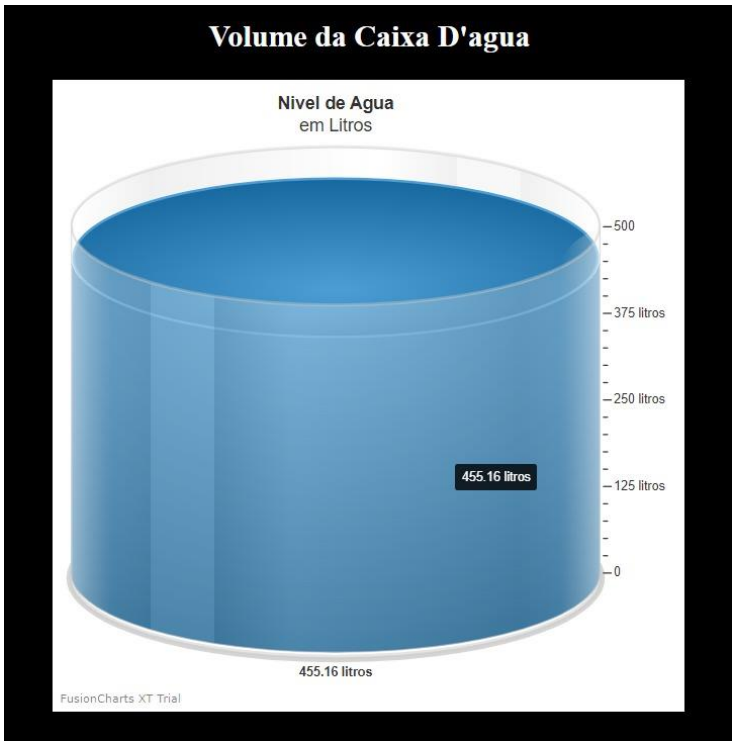
Optou-se por implementar dois elementos visuais nos arquivos: um gráfico em linha para exibir o histórico de leituras em tempo real, marcando a hora completa e a data da leitura, e um elemento visual de cilindro volumétrico simulando uma caixa d’água, exibindo o volume também em tempo real. Para a criação do gráfico, foi utilizada a biblioteca HighCharts, enquanto a biblioteca FusionCharts foi empregada para o cilindro volumétrico. As representações visuais podem ser observadas nas Figuras 10 e 11, e o código completo da página WEB está disponível no anexo I.

Para ampliar as capacidades do sistema, implementamos uma função que armazena os valores de leitura do sensor, calculando a média do nível da caixa d'água. Visando preservar as informações mesmo em caso de desligamento ou modo de espera da ESP, optamos por armazenar as leituras na memória EEPROM do dispositivo, uma vez que não dispõe de um conector para cartão SD.

Para otimizar o armazenamento na EEPROM, convertemos as leituras do formato float para o formato int. Dado que uma variável float ocupa 4 bytes, enquanto uma variável int ocupa apenas 2 bytes, essa conversão permite economizar espaço. Na implementação da função, reservamos uma seção de 512 bytes da memória EEPROM, proporcionando a capacidade de armazenar até 256 leituras. Quando esse limite é atingido, os recursos da memória são liberados para novas leituras por meio de comandos adicionais.

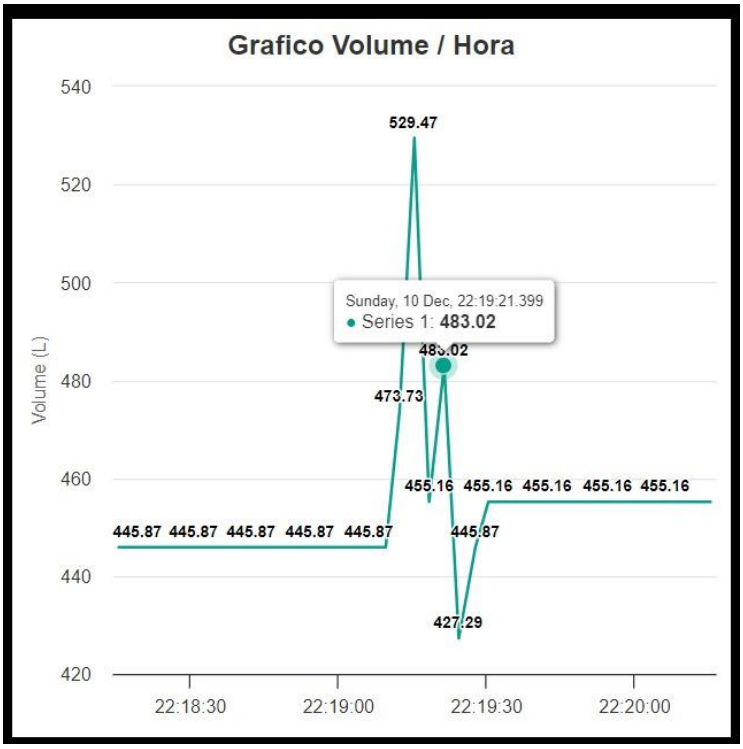
Essa abordagem garante a preservação eficiente do histórico de leituras, mesmo em situações de desligamento temporário ou em modo de espera, contribuindo para a confiabilidade e continuidade das informações sobre o nível da caixa d'água.

Figura 10 – Elemento visual FusionCharts (Cilindro)



Fonte: Autoria Própria

Figura 11 - Gráfico de tempo real Highcharts



Fonte: Autoria Própria

Para aprimorar a interface com o usuário, incorporamos a função de um Bot do Telegram. Este Bot, desenvolvido no aplicativo de mensagens instantâneas Telegram, tem a capacidade de receber mensagens automáticas enviadas por um sistema, como o implementado neste trabalho, e também aceita comandos enviados via mensagem no chat. O sistema conectado, por sua vez, responde com a informação solicitada.

No código, foram implementados dois alertas automáticos para situações anormais na caixa d'água. Quando o sensor registra uma leitura com valor acima de 5% do volume máximo, é enviada uma mensagem de alerta informando "Volume acima do normal". Se o sensor detectar um volume 10% acima do normal ou mais, um alerta "Perigo de Transbordar" é enviado, ambas as mensagens contendo os valores de volume lidos.

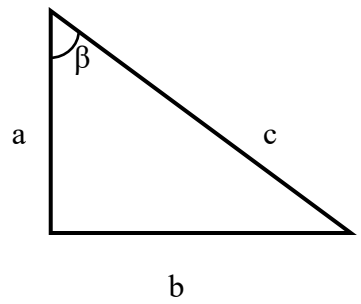
Adicionalmente, foram introduzidas outras funcionalidades para conectar o monitoramento da página web em tempo real e o bot do Telegram. Ao enviar o comando "Nivel" no chat, o bot retorna o valor do nível lido pelo sensor naquele momento. O comando "Media" retorna a média das leituras armazenadas na EEPROM da ESP. Por fim, o comando "Ip" proporciona uma mensagem de boas-vindas com o IP da ESP, que exibe a página web.

Era imperativo verificar a consistência da precisão da distância medida pelo sensor em superfícies sólidas e no espelho d'água. Isso foi inicialmente comprovado por meio de um teste

realizado em um balde com água completamente imóvel. Em seguida, um teste simulando o tubo de entrada, gerando um distúrbio na água, confirmou que a leitura do sensor permanecia precisa.

Embora esse teste inicial tenha sido crucial para verificar a resposta do sensor ao espelho d'água, era necessário realizar mais testes para determinar a calibração do sensor, avaliar variações na leitura com base na distância do sensor, identificar possíveis erros de leitura, entre outros aspectos. Para isso, foi estabelecida uma estrutura com uma superfície de água e diferentes níveis de altura, simulando o mais próximo possível uma caixa d'água de diversos tamanhos. O teste não foi realizado em uma caixa d'água real devido à falta de acesso a uma estrutura controlada. Como substituto, uma piscina foi utilizada.

Conforme as especificações do sensor indicam um ângulo de medição de 15º, para assegurar uma leitura precisa, é essencial que a área do espelho de água seja adequada, minimizando interferências. Por meio de cálculos simples de ângulos, é possível determinar o raio necessário do espelho d'água para diferentes níveis de altura. A título de exemplo, podemos calcular o raio necessário para uma altura de medição de 1 metro (100 cm).



$$Temos\ a = altura\ de\ medição$$

$$\beta = Ângulo\ de\ medição\ do\ sensor$$

$$c = hipotenusa$$

$$b = raio\ (espelho\ de\ água)$$

Fazendo o cálculo da hipotenusa c, temos:

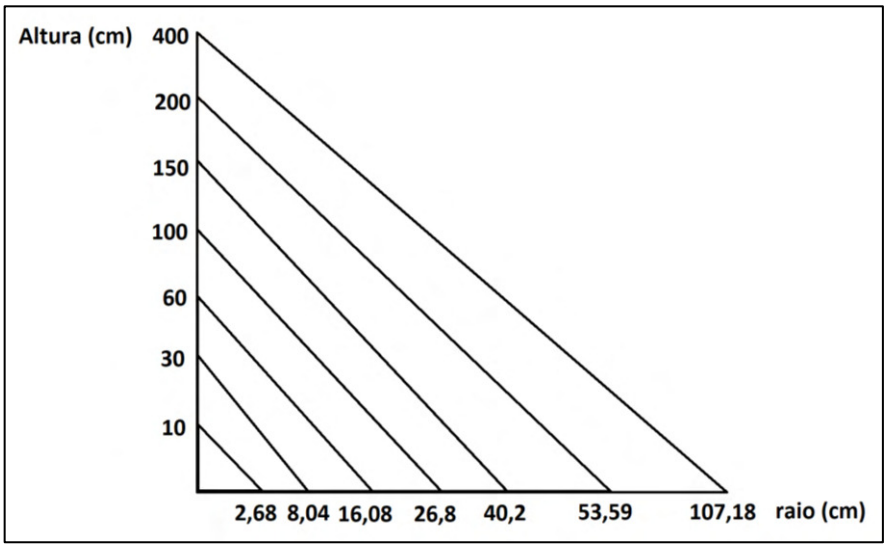
$$c = \frac{a}{\cos(\beta)} = \frac{100}{0{,}966} = 103{,}51$$

Agora calculando o valor de b, temos:

$$b = sen(\beta) \,.\left(\frac{a}{\cos(\beta)}\right) = 0{,}259 \,.\left(\frac{100}{0{,}966}\right) = 0{,}259 \,.\, 103{,}51 = 26{,}8\, cm$$

Com base nesse cálculo, podemos determinar que o raio necessário para o espelho d'água, a fim de garantir que o sensor realize a leitura com o mínimo de interferências possível, conforme recomendado pelo fabricante, é de aproximadamente 26,8 cm. A Figura 12 ilustra algumas medidas de raio calculadas para diferentes níveis de altura de medição.

Figura 12 – Níveis de altura de medição e raio do espelho de água necessário



Fonte: Autoria Própria

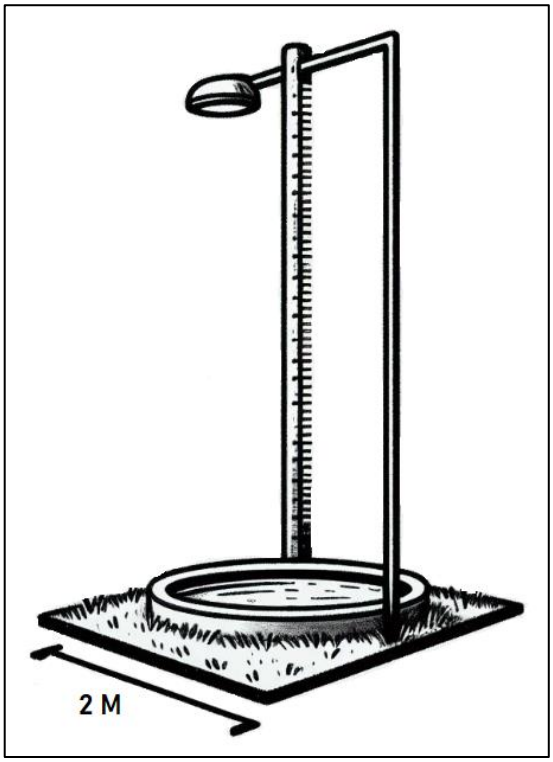
Com as informações necessárias para conduzir os testes, a estrutura foi montada para conduzir a leitura do sensor em diversos níveis de altura, como evidenciado na Figura 13. Para proporcionar uma compreensão mais clara da estrutura, a Figura 14 apresenta uma representação gráfica. Na ponta do braço, o sensor foi instalado alinhado com o centro da piscina, proporcionando assim um raio de espelho d'água de aproximadamente 1 metro. Isso permitiria a realização de testes até o valor do limite máximo de leitura estabelecido pelo fabricante do sensor, que é de 4 metros.

Figura 13 – Estrutura real utilizada nos testes

Fonte: Autoria própria

Os testes foram restritos a uma altura de 2,26 metros devido à limitação da estrutura disponível, que atingia essa altura máxima. Além disso, qualquer tentativa de realizar leituras em alturas superiores resultaria em instabilidade na posição do sensor, uma vez que seria necessário segurá-lo manualmente. Considerando o risco de uma possível queda do protótipo e do computador utilizado, a altura do teste foi limitada para garantir a segurança e estabilidade dos equipamentos, evitando a geração de dados incorretos.

Figura 14 – Representação gráfica da estrutura de testes



Fonte: Autoria própria

A piscina foi preenchida até atingir um nível considerado adequado para os testes, alcançando uma altura de aproximadamente 7,5 cm, conforme mostrado na Figura 15. No primeiro nível da estrutura, o sensor foi posicionado a uma altura de 29,3 cm em relação ao chão, ou seja, abaixo do nível da água de 7,5 cm, como indicado na Figura 16. Posteriormente, o sensor foi ativado, apresentando uma leitura de distância até o nível da água de 21 cm conforme evidenciado na Figura 16[1].

Foram realizadas 30 leituras, divididas em 3 conjuntos de testes com 10 leituras cada, em todos os testes de diferentes alturas. Essa primeira medição será referida como Teste 1, nessas condições a cada 10 leituras, 9 apresentavam o valor de 21 cm, e 1 leitura apresentava o valor de 20cm, como é possível conferir na Tabela 5, tendo como valor médio de leitura do sensor igual a 20,9. O desvio padrão também foi calculado para determinar o grau de dispersão

[1] As figuras, incluindo a Figura 16, que exibem a leitura do sensor na porta serial da IDE do Arduino, apresentam informações sobre volume, número de leituras e a distância/altura de leitura. No entanto, essas informações relacionadas à volume e número de leituras são irrelevantes no momento, sendo impressas apenas para fins de teste. Neste contexto, apenas a informação de leitura de altura é relevante. Na figura 19 a variável distancia é que deve ser considerada.

das leituras, para indicar o quanto o conjunto de leituras é uniforme. Quanto mais próximo de 0 for o desvio padrão, mais homogêneo são as leituras. O cálculo é demonstrado a seguir:

$$DP = \sqrt{\frac{\sum_{i=1}^{n}(x_i - M_a)^2}{n}}$$

Sendo,

∑: símbolo de somatório. Indica que temos que somar todos os termos, desde a primeira posição (i=1) até a posição n

x_i: valor na posição **i** no conjunto de dados

M_A: média aritmética dos dados

n: quantidade de dados

Assim,

$$DP\ (\sigma) = \sqrt{\frac{\sum_{i=1}^{30}(x_i - 20{,}9)^2}{30}}$$

O cálculo do desvio padrão foi realizado em uma ferramenta online, endereço eletrônico nas referências.

No teste 1, encontramos o valor do desvio padrão de σ = 0,3.

A figura 17 apresenta algumas dessas leituras na porta serial. Esse teste demonstrou que, nessa distância, o sensor apresentou precisão e exatidão em suas leituras, uma vez que 29,3 cm - 7,5 cm = 21,8 cm, com erro relativo percentual de 4,32%.

Tabela 5 - Resultados Teste 1 – Conjunto 1

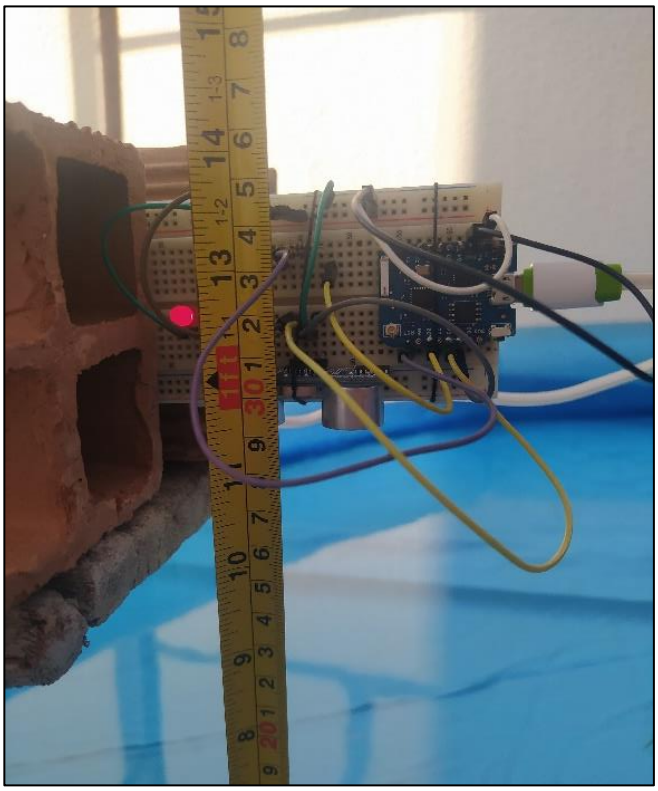
Leitura nº	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Valor Medido (cm)	21	21	20	21	21	21	21	21	21	21

Fonte: Autoria Própria

Figura 15 – Altura do nível da água

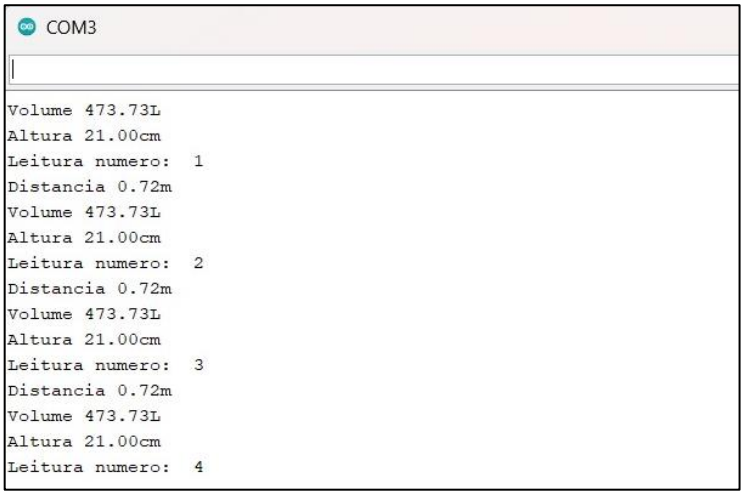
Fonte: Autoria Própria

Figura 16 – Altura do sensor Teste 1



Fonte: Autoria Própria

Figura 17 - Leitura do sensor Teste 1



Fonte: Autoria Própria

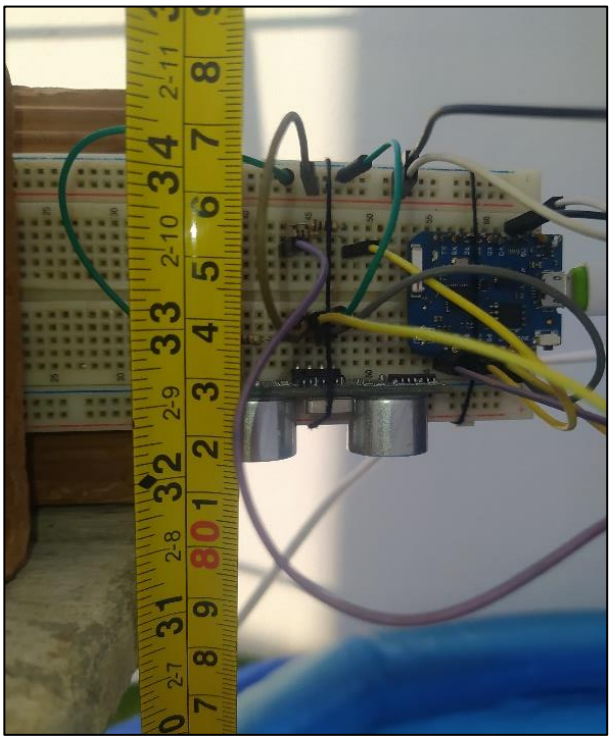
No segundo nível de leitura, denominado Teste 2, o sensor foi posicionado a uma altura de aproximadamente 81,6 cm do chão, conforme ilustrado na Figura 18. O mesmo nível de água foi mantido, e a leitura do sensor em operação mostrou aproximadamente 72 cm, conforme visto na Figura 19. A cada 10 leituras, 7 apresentavam o valor de 72 cm, e 3 leituras apresentavam variações de 1 cm para mais ou para menos, como é possível conferir na Tabela 6, tendo como valor médio de leitura do sensor igual a 71,9. Com isso, observamos que, ao aumentar a distância, um pequeno erro relativo percentual já pode ser verificado, sendo de aproximadamente 3,06%, dado que 81,6 cm - 7,5 cm = 74,1 cm. Com isso, podemos afirmar que o sensor continua apresentando uma precisão satisfatória, como no Teste 1, porém a sua exatidão foi menor comparada ao Teste 1.

Para o desvio padrão, temos:

$$DP\ (\sigma) = \sqrt{\frac{\sum_{i=1}^{30}(x_i - 71{,}9)^2}{30}}$$

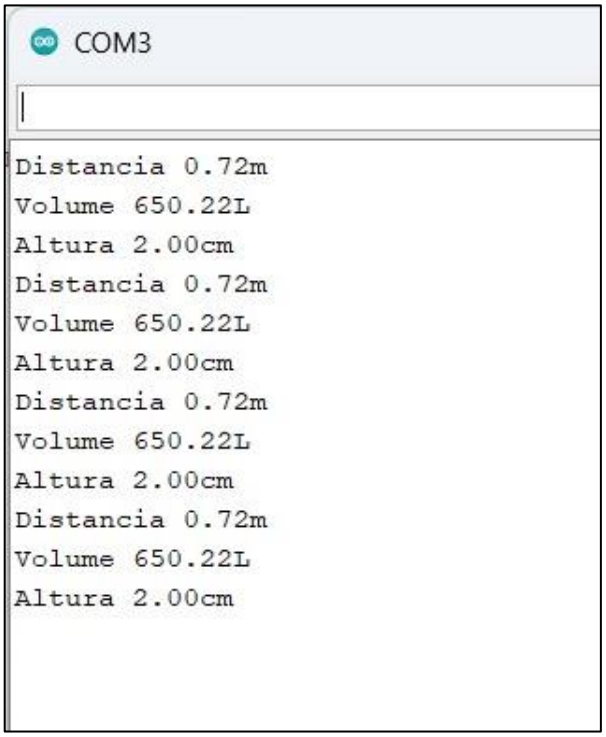
No teste 2, encontramos o valor do desvio padrão de σ = 0,538

Figura 18 – Altura do sensor Teste 2



Fonte: Autoria Própria

Figura 19 - Leitura do sensor Teste 2



Fonte: Autoria Própria

Tabela 6 - Resultados Teste 2 – Conjunto 3

Leitura nº	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Valor Medido (cm)	72	71	71	72	72	72	73	72	72	72

Fonte: Autoria Própria

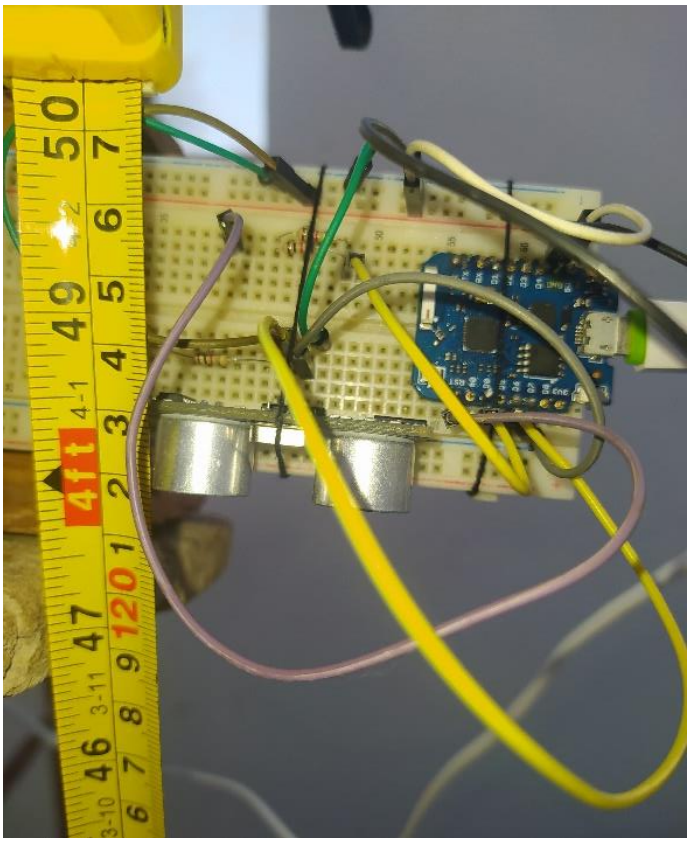
No Teste 3, o sensor foi posicionado a uma distância do chão de aproximadamente 121,9 cm, como mostrado na Figura 20. A leitura exibida pelo sensor na porta serial foi de aproximadamente 110 cm, conforme ilustrado na Figura 21. A média das leituras nos 3 conjuntos de 10 leituras se mostrou bem parecida com o Teste 2, onde aproximadamente 3 leituras a cada 10 apresentavam uma variação de 1 cm para mais ou para menos, resultando em uma média de 109,7 cm. Com isso observamos que a diferença entre a altura real (121,9 cm - 7,5 cm = 114,4 cm) e a leitura do sensor foi de aproximadamente 4,7 cm, representando um erro relativo percentual de cerca de 4,2%, e confirmando que o sensor continua apresentado precisão, porém a sua exatidão diminuiu ainda mais com o aumento da distância de leitura. O conjunto 1 de leituras do Teste 3 pode ser conferido na Tabela 7.

Para o desvio padrão, temos:

$$DP\ (\sigma) = \sqrt{\frac{\sum_{i=1}^{30}(x_i - 109{,}7)^2}{30}}$$

No teste 2, encontramos o valor do desvio padrão de σ = 0,458

Figura 20 - Altura do sensor Teste 3



Fonte: Autoria Própria

Figura 21 - Leitura do sensor Teste 3

COM3

```
Distancia 0.72m
Volume -352.98L
Altura 110.00cm
Distancia 0.72m
Volume -352.98L
Altura 110.00cm
Distancia 0.72m
Volume -343.69L
Altura 109.00cm
Distancia 0.72m
Volume -343.69L
Altura 109.00cm
Distancia 0.72m
Volume -352.98L
Altura 110.00cm
```

Fonte: Autoria Própria

Tabela 7 - Resultados Teste 3 – Conjunto 1

Leitura nº	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Valor Medido (cm)	110	110	109	110	110	110	109	109	110	110

Fonte: Autoria Própria

No Teste 4, o sensor foi posicionado a uma distância do chão de aproximadamente 225,8 cm, conforme mostrado na Figura 22. A leitura exibida pelo sensor na porta serial foi de aproximadamente 206 cm, ilustrada na Figura 23. No conjunto de leituras, o sensor mantém a sua precisão, tendo variação de 1 cm para mais ou para menos em 3 a cada 7 leituras, dispondo uma média de distância de leitura de 206,3 cm, como exemplificado na Tabela 8 com o conjunto 2 do Teste 4. Com isso, observamos que a diferença entre a altura real (225,8 cm - 7,5 cm = 218,3 cm) e a leitura do sensor foi de aproximadamente 12 cm, representando um erro relativo percentual de cerca de 5,82%. Isso evidencia que a margem de erro acompanhou o aumento da distância do sensor, essa diferença pode representar uma quantidade significativa de água não detectada pelo sensor, além disso, nessa distância podemos ver que o sensor não apresenta uma exatidão satisfatória para os propósitos desse trabalho, mas continua mantendo a sua precisão.

Para o desvio padrão, temos:

$$DP\ (\sigma) = \sqrt{\frac{\sum_{i=1}^{30}(x_i - 206{,}3)^2}{30}}$$

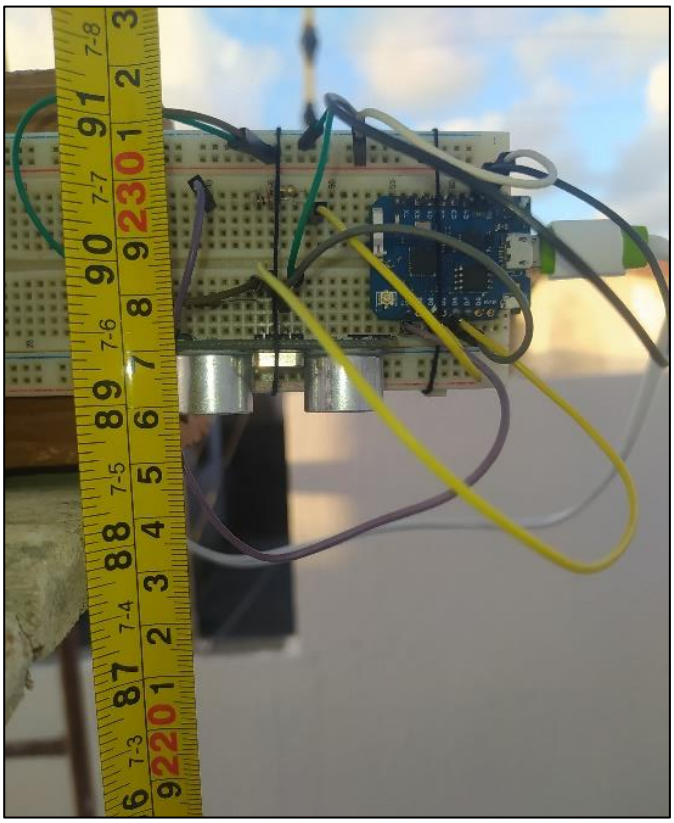
No teste 2, encontramos o valor do desvio padrão de σ = 0,458

Tabela 8 - Resultados Teste 4 – Conjunto 2

Leitura nº	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Valor Medido (cm)	206	206	206	207	207	207	206	206	206	206

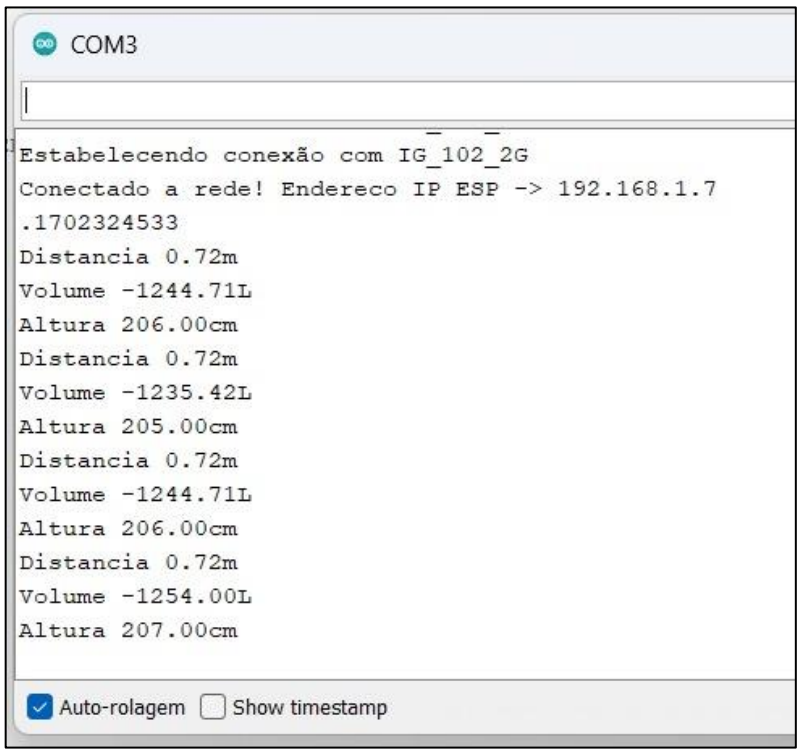
Fonte: Autoria Própria

Figura 22 - Altura do sensor Teste 4



Fonte: Autoria Própria

Figura 23 - Leitura do sensor Teste 4



Fonte: Autoria Própria

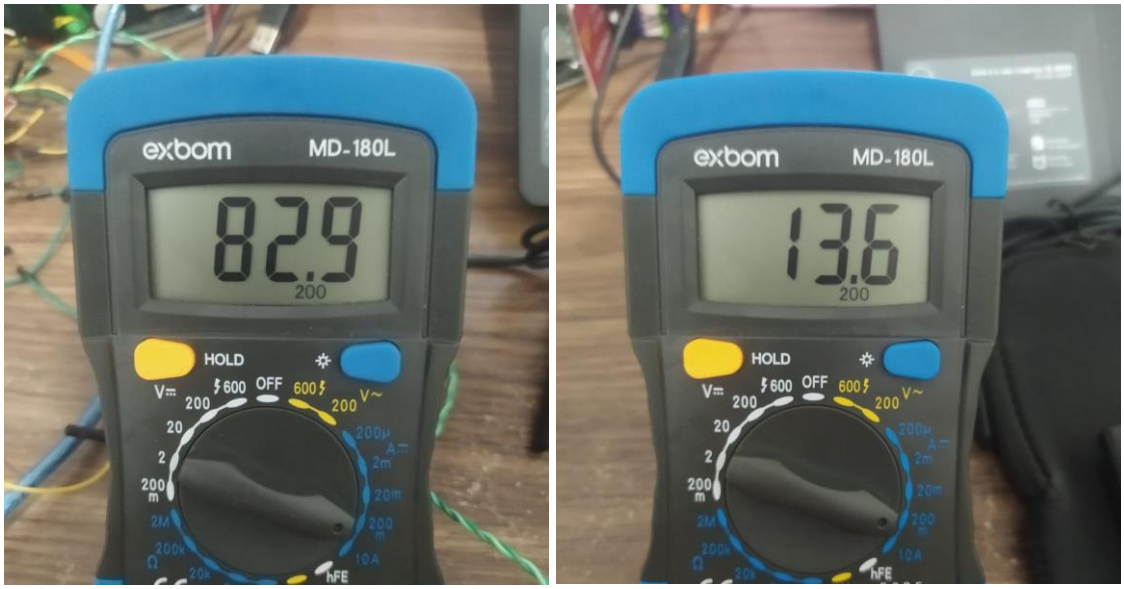
A altura máxima que se mostrou possível a medição, diante do que estava disponível, foi a mostrada no teste 4.

Tendo em vista os testes realizados, é possível afirmar que a leitura do sensor na água apresenta resultados satisfatórios e que pode ser utilizado na aplicação, levando em conta as informações de altura de caixas d'água da Fortlev, onde apenas caixas com volume igual ou superior à 5000L apresentam altura total de 2 m ou mais, faixa de altura onde o sensor já não apresenta a exatidão exigida.

Considerando que o sistema embarcado estava localizado em uma área de difícil acesso e alimentado por uma bateria, foi incorporada ao código a função "*sleep*", que coloca a ESP em modo de espera nos momentos em que não é necessário realizar processamento. O código efetua uma leitura a cada aproximadamente 4 segundos, e foi definido realizar 30 leituras de nível antes de a placa entrar no modo *sleep* por 1 minuto. Após esse intervalo, o sistema é reativado para realizar novamente 30 leituras. É importante observar que durante o modo *sleep*, a conexão com a rede é interrompida, tornando a página web indisponível.

Essa função se mostrou extremamente eficiente, visto as leituras de consumo do circuito durante as leituras e no modo *sleep*, ilustradas na Figura 24.

Figura 24 - Consumo em processamento (a esquerda) e no modo sleep (a direita)

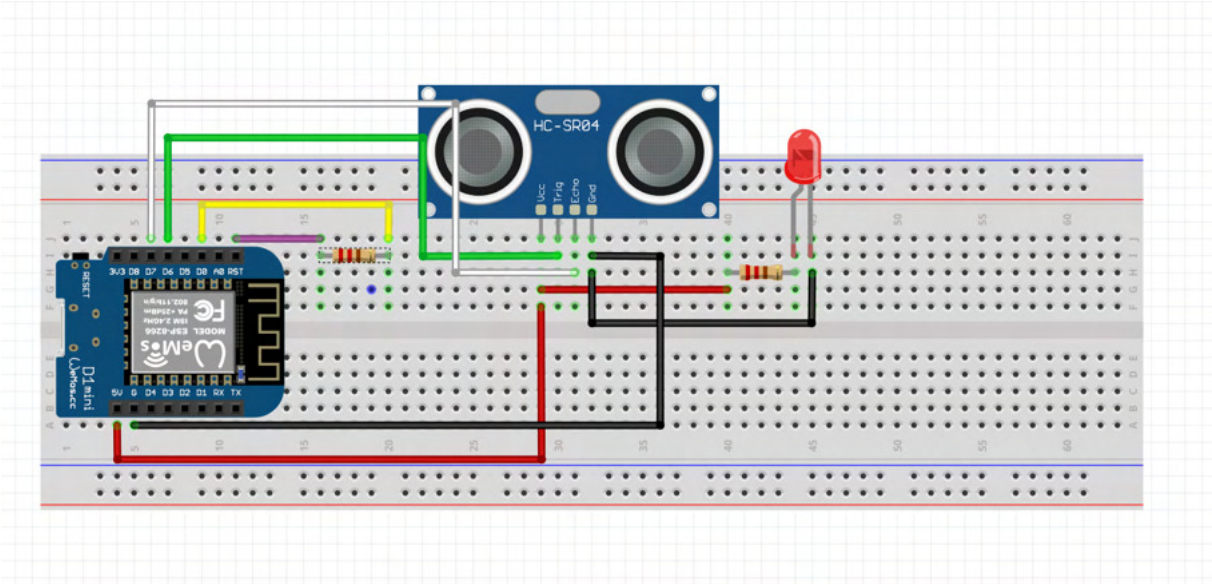


Fonte: Autoria Própria

A função *sleep* coloca o Arduino em modo "standby", mas pode apresentar a questão de não acordar automaticamente após o término do tempo definido na função. Para contornar esse problema, uma solução encontrada nos fóruns do Arduino envolve a conexão da porta GPIO 16 (D0) à porta RST por meio de um resistor de aproximadamente 500 ohms. Isso ocorre porque, na ESP8266, o estado do pino RST é sempre HIGH durante o funcionamento, e quando recebe um sinal LOW, o microcontrolador é reiniciado. Ao atribuir um temporizador com a função Deep Sleep, ao término do tempo determinado, o pino D0 emite um pulso LOW. Portanto, ao conectar um resistor entre a porta D0 e RST, cria-se uma espécie de interruptor que desperta a placa.

Com isso estabelecido, o circuito ficou na configuração mostrada na Figura 25.

Figura 25 - Configuração do circuito com a função sleep



Fonte: Autoria Própria (Software Fritzing)

Para calcular o tempo necessário para a caixa d'água retornar ao seu estado de 100% de capacidade após o uso de uma determinada quantidade de água, foi necessário realizar um teste em uma caixa d'água real.

Esse teste foi conduzido de duas maneiras diferentes. Primeiro, com a caixa d'água completamente cheia, utilizando uma descarga com aproximadamente 7 litros de água. Em seguida, ligando duas torneiras para manter um fluxo constante de uso de água e verificar se existia uma variação no nível que poderia ser detectada para calcular o volume de água utilizado e determinar o consumo.

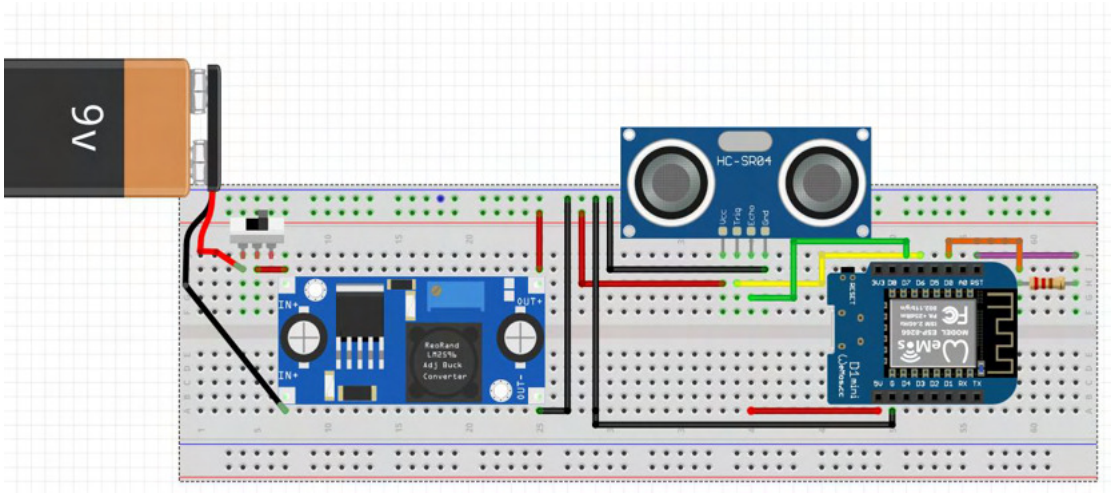
Ambos os testes concluíram que não foi possível determinar o volume de água gasto. A caixa d'água, do Modelo Fortlev 500L, possui um diâmetro de abertura de 1,22m. Durante o teste com a descarga de 7 litros, a variação no nível da caixa não foi detectada pelo sensor, devido à boia mecânica de controle de nível que inicia o reabastecimento imediatamente após a variação mínima de volume de água.

No segundo teste com ambas as torneiras ligadas, o nível de água leva alguns segundos adicionais para diminuir. Contudo, assim que a boia sofre a variação novamente, o processo de reabastecimento da caixa é imediatamente iniciado.

Ao desligar uma torneira e manter apenas uma ligada, buscando equilibrar o volume de água que sai da caixa com o volume permitido pela boia, foi crucial para evidenciar a inviabilidade do cálculo do consumo, pois não ocorria variação no nível do volume de água. Mesmo se houvesse um ponto onde a vazão do consumo fosse ligeiramente maior que o volume de entrada, como por exemplo 10% ou 20%, essa alteração não seria detectada, conforme indicado pelos resultados dos testes anteriores, nos quais um consumo instantâneo de aproximadamente 7 litros de água não causou variações consideráveis no sensor.

Após os testes, foi desenvolvido o protótipo do circuito final, que incluiria a integração da bateria de alimentação. A escolha para o projeto recaiu sobre uma bateria de lítio recarregável, modelo 6sp753868, com 3.7V, 2100mAh e 7.77Wh. Dado que tanto a placa quanto o sensor demandam uma alimentação de 5V, optou-se por utilizar duas baterias conectadas em série, resultando em uma tensão de alimentação de aproximadamente 6V, medida por meio de um multímetro. Para ajustar essa tensão, incorporou-se o regulador de tensão LM 2596 DC-DC. Concluindo a fase de testes e ajustes, foi concebido o circuito final a ser implementado. Nesse processo, incluiu-se o regulador de tensão LM 2596, eliminando o LED e o resistor originalmente destinados à visualização do funcionamento do circuito. Essa modificação se deu em virtude da presença de um LED no próprio LM 2596, indicando seu funcionamento. Portanto, o LED anterior seria um consumo desnecessário de bateria. Além disso, foi adicionada uma chave switch no circuito para controlar a alimentação. A configuração final pode ser visualizada na Figura 26.

Figura 26 - Configuração Final do Circuito



Fonte: Autoria Própria (Software Fritzing)

Os testes de consumo de bateria foram conduzidos nesse circuito, revelando um consumo de 90 mA durante o funcionamento e 13,6 mA em modo de espera. Com esses dados em mente, é possível implementar algumas configurações na programação da função *sleep* para otimizar o uso da bateria. A ESP8266 tem um intervalo de aproximadamente 4 segundos entre cada leitura registrada. Em outras palavras, ao realizar 30 leituras, a placa permanecerá ativa por 120 segundos antes de entrar no modo *sleep* pelo tempo determinado, considerando, neste caso, 1 minuto. Para calcular a duração da bateria, empregamos a seguinte fórmula:

$$T = \frac{C}{I}$$

Onde,

T = Tempo (h)

C = Capacidade (mAh)

I – Corrente (I)

Na configuração mencionada, a capacidade requerida da bateria é dada por:

$$C = (T_1 . I_1) + (T_2 . I_2) = (120s . 90mA) + (60s \, . 13{,}6mA)$$

$$= (10800\ mAs) + (816\ mAs) = 11616\ mAs$$

Com isso, podemos calcular a corrente utilizada pelo circuito:

$$I = \frac{C}{T_1 + T_2} = \frac{11616\ mAs}{120\ s + 60\ s} = 64{,}53\ mA$$

Fazendo o cálculo do tempo de capacidade da bateria, temos:

$$T = \frac{2100\ mAh}{64{,}53\ mA}$$

$$T\ = 32{,}54\ horas$$

Essa seria a duração estimada da bateria nessa configuração, assumindo condições ideais. Entretanto, fatores como a eficiência do regulador de tensão, autodescarga da bateria, variações de tensão, condições ambientais, ciclo de vida da bateria, entre outros, podem influenciar esse tempo de duração. A Lei de Peukert (PEUKERT, 1897) define o expoente de Peukert como uma medida mais precisa para ajustar o cálculo da bateria, mas, quando as especificações detalhadas da bateria não são conhecidas, um fator é utilizado com base nas condições específicas do sistema. Neste caso, utilizaremos um fator de 0,8, equivalente a 80%. Portanto, teremos um tempo estimado da bateria de 26,03 horas. A Tabela 9 exemplifica alguns modelos de configurações do sistema, apresentando o tempo em execução, o tempo em modo sleep, o consumo em cada perfil, o tempo de duração ideal e a duração real esperada, considerando o fator de 0,8. O tempo de execução utilizado foi de 4 segundos para cada leitura, somado ao tempo que a ESP8266 demora para se conectar à rede WiFi, onde foi considerado um tempo de 5 segundos.

Tabela 9 - Configurações de consumo da bateria

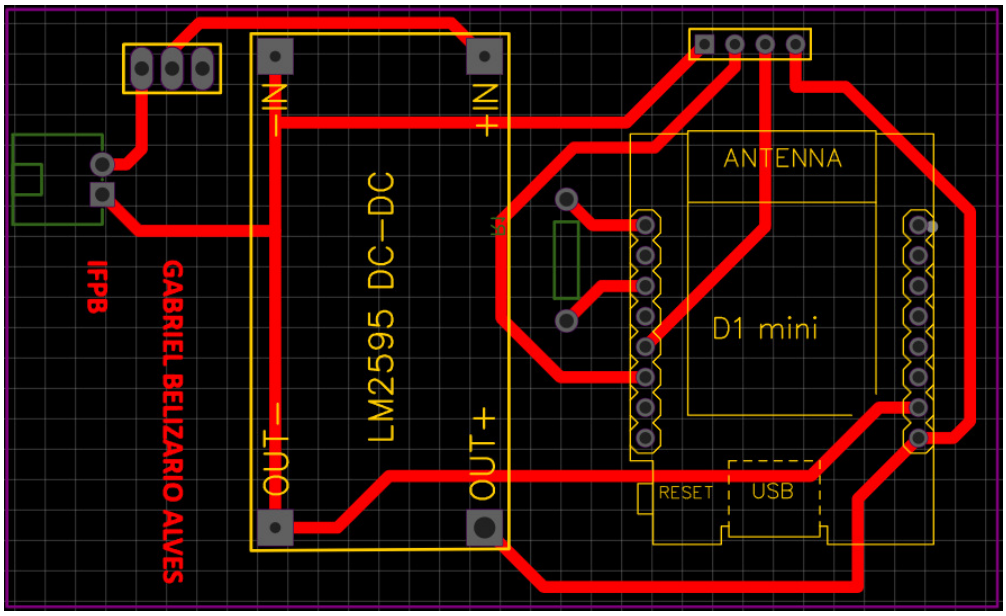
PERFIS DE CONSUMO

Perfil	Leituras	Execução	*Sleep*	Consumo	Duração Ideal	Duração Real
01	5	25 s	60 s	36,07 mA	58,22 horas	46,58 horas
02	5	25 s	120 s	26,77 mA	78,44 horas	62,75 horas
03	10	45 s	60 s	46,34 mA	45,32 horas	36,26 horas
04	10	45 s	120 s	34,44 mA	60,97 horas	48,78 horas
05	20	85 s	60 s	58,39 mA	35,96 horas	28,77 horas
06	20	85 s	120 s	45,28 mA	46,37 horas	37,10 horas
07	30	125 s	60 s	65,22 mA	32,20 horas	25,76 horas
08	30	125 s	120 s	52,58 mA	39,94 horas	31,95 horas

Fonte: Autoria Própria

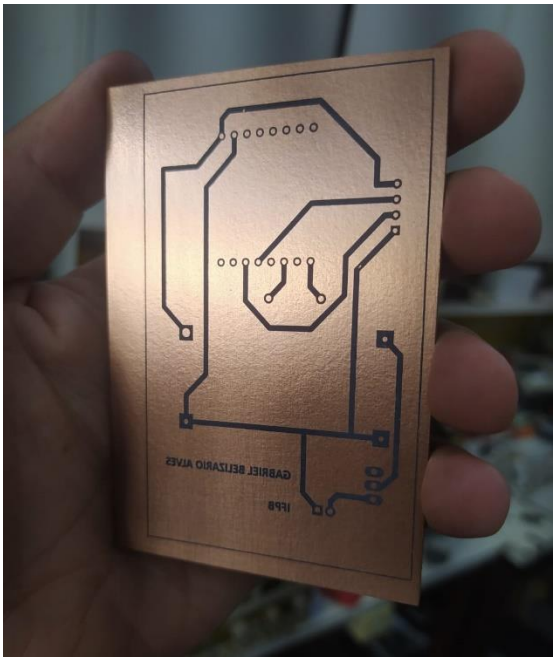
Em seguida, avançamos para o desenvolvimento do projeto da placa de circuito impresso que integraria todos os componentes do circuito. Optamos pelo software EasyEDA Standart devido à sua interface simples e às ferramentas adequadas para a elaboração da placa, conforme ilustrado na Figura 27. Após a concepção, a placa foi produzida em uma folha de cobre de uma única face, como evidenciado na Figura 28, utilizando percloreto de ferro para corrosão e definição das trilhas, cujo resultado pode ser observado na Figura 29. Por fim, o protótipo definitivo foi instalado em uma caixa d'água, e o monitoramento remoto foi realizado com sucesso.

Figura 27 - Projeto da PCI no EasyEDA



Fonte: Autoria Própria (Software EasyEDA)

Figura 28 - Circuito Impresso no cobre



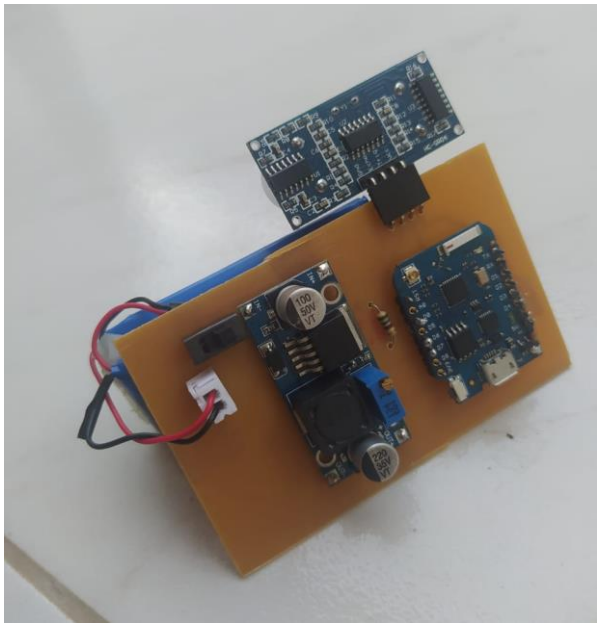
Fonte: Autoria Própria

Figura 29 - Circuito pós corrosão

Fonte: Autoria Própria

Em seguida, procedemos com o teste de continuidade em todas as trilhas utilizando o multímetro, não identificando qualquer curto ou falha. Com essa validação, prosseguimos para a transferência dos componentes do circuito da Protoboard para a placa, conforme mostrado na Figura 30. Finalmente, foram realizados os mesmos testes com o circuito na placa, onde os mesmo resultados foram obtidos.

Figura 30 - Projeto Final na PCI (Placa de Circuito Impresso)



Fonte: Autoria Própria

6. RESULTADOS E DISCUSSÕES

A realização dos testes com o sensor de nível proporcionou insights significativos quanto à sua precisão e desempenho em diferentes alturas. Foram conduzidos 30 testes divididos em três baterias, cada uma compreendendo 10 leituras, abrangendo variados níveis de altura. Esta seção destaca os principais resultados obtidos e discute as implicações dessas descobertas.

Os testes iniciais, denominados como Teste 1, realizados a uma altura aproximada de 21,8 cm do espelho de água, revelaram uma notável consistência nas leituras do sensor. Em cada bateria de 10 leituras, 9 apresentaram um valor de 21 cm, enquanto apenas 1 leitura indicou 20 cm. A média dessas leituras foi de 20,9 cm, demonstrando uma precisão e exatidão, de aproximadamente 0,9 cm, satisfatórias nas leituras do sensor a essa distância específica.

Ao progredir para alturas maiores, observou-se que as leituras do sensor mantiveram uma consistência notável. No entanto, a exatidão, definida como a diferença entre os valores medidos (reais) e os valores esperados (teóricos), apresentou algumas variações. No Teste 2, realizado a uma altura de aproximadamente 74,1 cm, a leitura média foi de aproximadamente 71,9 cm, indicando uma exatidão de aproximadamente 2,2 cm em relação ao valor teórico. No Teste 3, a uma altura de aproximadamente 114,4 cm, a leitura média foi de cerca de 109,7 cm, representando uma exatidão de aproximadamente 4,7 cm. No Teste 4, a uma altura de aproximadamente 218,3 cm, a leitura média foi de cerca de 206,3 cm, resultando em uma exatidão de aproximadamente 12 cm. Essas variações indicam uma tendência de aumento na diferença entre os valores medidos e os valores esperados à medida que aumenta a distância do sensor.

Esses resultados destacam a importância de uma calibração cuidadosa e considerações sobre a distância do sensor em aplicações específicas. O sensor de nível mostrou-se adequado para determinadas faixas de altura, proporcionando leituras confiáveis e consistentes. No entanto, é imperativo compreender suas limitações, especialmente em contextos que envolvem distâncias consideráveis.

7. CONCLUSÕES

Diante da realização dos testes e análises conduzidas no contexto do medidor de nível, é possível extrair conclusões significativas sobre a performance e viabilidade do dispositivo. A estrutura experimental montada permitiu a aquisição de dados em diferentes alturas, proporcionando uma compreensão mais abrangente do comportamento do sensor.

Os resultados obtidos nos testes indicam que o medidor de nível, baseado em microcontroladores, apresenta consistência e precisão em suas leituras. A avaliação estatística dos dados coletados se faz essencial para compreender a variabilidade nas medições, considerando fatores como a altura, possíveis interferências e a estabilidade do sensor.

Uma análise detalhada das leituras em diferentes alturas revelou que o dispositivo manteve um erro percentual aceitável, evidenciando sua confiabilidade em condições controladas. É crucial abordar questões estatísticas, como médias, desvios padrão e possíveis variações entre os testes, para oferecer uma visão mais completa da precisão do medidor. A análise estatística foi imprescindível para determinar a confiabilidade das leituras, e mensurar a exatidão do sensor.

A limitação da altura dos testes a 2,26 metros foi estrategicamente estabelecida para garantir a estabilidade da estrutura e a segurança do equipamento. A preocupação com possíveis variações nas leituras em alturas superiores e a preservação da integridade do protótipo foram considerações relevantes na definição desse limite.

Com base nos dados coletados, é possível afirmar que o medidor de nível, utilizando microcontroladores, mostra-se promissor para aplicações práticas que demandam medições precisas em diferentes alturas de líquidos. A continuidade do desenvolvimento desse dispositivo pode ser orientada pelos insights obtidos nesta pesquisa, direcionando futuras investigações para aspectos como efeitos de temperatura, desgaste, consumo de energia e possíveis aprimoramentos na conectividade.

A integração de tecnologias adicionais, como a implementação de um sistema de boia eletrônica operando em conjunto com o sensor de nível, abre a possibilidade de gerenciar o abastecimento da caixa d'água, permitindo, em teoria, a quantificação do consumo de água. A inclusão de ferramentas e integrações com outras tecnologias de interação com o usuário, como o assistente virtual Alexa da Amazon, por exemplo, torna-se atrativa para aprimorar o conceito de automação residencial e Internet das Coisas (IoT).

8. **REFERÊNCIAS**

ALMEIDA, R. M. A. de; MORAES, C. H. V.; SERAPHIM, T. F. P. Programação de sistemas embarcados: Desenvolvendo software para microcontroladores em linguagem C. 1. ed. [S.l.]: GEN LTC, 2016.

ANTONIO, F. G. Multiflow WiFi utilizando Software Defined Networking. Tese (Doutorado) — Universidade de Coimbra, 2014.

Banzi, M. (2011). Getting Started with Arduino. Maker Media, Inc.

BARKUSOV, Yevgen; QIAN. Jinrong; Battery Power Management for Portable Devices. 2013.

BENTLEY, J. P. Principles of Measurement Systems. Harlow (England): Pearson, 2005.

CHARTS, Fusion; https://www.fusioncharts.com/about-us - Acesso em 06 de dezembro de 2023.

CHUNG, Taddy; BAGUR, José; https://docs.arduino.cc/learn/electronics/low-power - Acesso em 01 de dezembro de 2023.

CLARKE, Robin T.; KING, Jannet. O atlas da água: o mapeamento completo do recurso mais precioso do Planeta. São Paulo: Publifolha, 2005.

COLLISCHONN, W; DORNELLES, F; Hidrologia para engenharia e ciências ambientais.

COMPONENTES, Hobby; https://hobbycomponents.com/development-boards/864-wemos-d1-mini-pro-esp8266-development-board; Acesso em 15 de novembro de 2023.

DATASHEET, ESP8266; https://www.openimpulse.com/blog/wp-content/uploads/wpsc/downloadables/0A-ESP8266__Datasheet__EN_v4.3.pdf - Acesso em 05 de Dezembro de 2023.

DATASHEET, ESP8266; https://www.makerhero.com/img/files/download/ESP8266EX-Datasheet.pdf - Acesso em 15 de novembro de 2023.

EADY, Fred; Implementing 802.11 with Microcontrollers: Wireless Networking for Embedded Systems Designers *Embedded Technology* - Elsevier, 2005.

GAST, M. S; Wi-Fi Handbook: Building 802.11b Wireless Networks. McGraw-Hill, 2005.

FLANAGAN, D. JavaScript: the Definitive Guide. [S.l.]: "O'Reilly Media, Inc.", 2011.

FIELDING, R. et al. Hypertext transfer protocol–HTTP/1.1. [S.l.], 1999.

HERO, Maker; https://www.makerhero.com/blog/conheca-a-wemos-d1-mini-pro-mais-funcionalidades-para-o-esp8266/ - Acesso em 15 de novembro de 2023.

HIGHCHARTS; https://www.highcharts.com/about/ Acesso em 06 de dezembro de 2023.

HOROWITZ, P., & Hill, W. (2015). A Arte da Eletrônica. 3ª Edição.

INSTRUMENTS, Texas; https://www.makerhero.com/img/files/download/LM2596-Datasheet.pdf - Acesso em 27 de novembro de 2023.

LINDEN, D., REDDY, T. B., & MEISSNER, P.; Battery Technology Handbook. CRC Press, 2003.

MA, Y.; ZHOU, G.; WANG, S. Wifi sensing with channel state information: A survey. ACM Computing Surveys (CSUR), ACM New York, NY, USA, v. 52, n. 3, p. 1–36, 2019.

MONK, S. (2016). Programming Arduino: Getting Started with Sketches. McGraw-Hill Education.

MOHAN, Ned, Tore M. Undeland, William P. Robbins Power Electronics: Converters, Aplications and Design, 3 Edition, October 2002.

MOREIRA, M.C. *Gestão de recursos hídricos*: sistema integrado para otimização da outorga de uso da água. 2006. 105 f. Dissertação (Mestrado em Engenharia Agrícola) - Universidade Federal de Viçosa, Viçosa, 2006.

NALON, José Aleandre – Introdução ao Processamento Digital de Sinais – Rio de Janeiro, LTC, 2009.

PEUKERT, W. (1897). Über die abhängigkeit der kapazität von der entladestromstärke bei bleiakkumulatoren. Elektrotechnische Zeitschrift.

PRESSMAN, A. I., Billings, M., Morey, D., & Deller, J. (2015). Switching Power Supply Design. McGraw-Hill Education.

RASHID, M. H. (2014). Eletrônica de Potência: Circuitos, Dispositivos e Aplicações.

RAGGETT, D. et al. Html 4.01 specification. W3C recommendation, v. 24, 1999.

SILVA, M. S. Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata. [S.l.]: Novatec Editora, 2007.

TANENBAUM, A. S.; WOODHULL, A. S. Sistemas Operacionais: Projeto e Implementação. 2. ed. [S.l.]: Bookman, 2000. ISBN 8573075309,9788573075304.

TECHNOLOGY, Handson; https://www.handsontec.com/dataspecs/HC-SR04-Ultrasonic.pdf - Acesso em 27 de novembro de 2023.

APÊNDICES

APÊNDICE I – CÓDIGO DA ESP8266

```
// TCC - MEDIÇÃO DE NÍVEL E VOLUME DA CAIXA D'ÁGUA


#include <Ultrasonic.h>             // Biblioteca para utilizar o sensor ultrassônico
#include <ESP8266WiFi.h>            // Biblioteca para conexão WiFi
#include <ESPAsyncTCP.h>            // Biblioteca para protocolo TCP assincrono
#include <ESPAsyncWebServer.h>      // Biblioteca para servidor web assincrono
#include <FS.h>                     // Sistema de arquivos SPIFFS para ESP8266
#include <Wire.h>                   // Comunicação I2C
#include <Arduino_JSON.h>           // Manipulação de dados em formato JSON
#include <EEPROM.h>                 // Biblioteca para acessar a memória EEPROM
#include <WiFiClientSecure.h>       // Cliente WiFi seguro
#include <UniversalTelegramBot.h>   // Biblioteca para integração com o Telegram
#include <TimeLib.h>                // Biblioteca para manipulação de tempo
#include <NTPClient.h>              // Cliente NTP para obter a hora atual
#include <WiFiUdp.h>                // Biblioteca para comunicação UDP
#include <ArduinoJson.h>            // Manipulação de dados em formato JSON



//  INICIA OS PINOS TRIG E ECHO DO SENSOR ULTRASÔNICO

Ultrasonic ultrasonic (D6, D7);


const long utcOff = -10800;                                     // Offset de fuso horário em segundos (UTC-3 para o horário de Brasília)
WiFiUDP ntpUDP;                                                 // Objeto UDP para comunicação com o servidor NTP
NTPClient timeClient(ntpUDP, "pool.ntp.org", utcOff);           // Cliente NTP para sincronização de tempo


float raio1 = 0.60;         // Raio interno da caixa d'água em metros
float raio2 = 0.475;        // Raio externo da caixa d'água em metros
float altura2 = 0.72;       // Altura total da caixa d'água em metros
float altura1;              // Altura calculada com base na distância medida
int capacidade = 500;       // Capacidade máxima da caixa d'água em litros

float distancia;            // Armazena a distância medida pelo sensor ultrassônico em centímetros
float distancia1;           // Armazena a distância convertida de centímetros para metros (função hcsr04)
float distancia2;           // Armazena a distância convertida de centímetros para metros (função getDistance)
String result;              // Armazena a altura (distância) como uma string
String result2;             // Armazena o volume da caixa d'água como uma string
String result3;
float volume;               // Volume calculado da caixa d'água em litros
String ip;


unsigned long lastRunTime = 0;                      // Armazena o tempo da última execução
const unsigned long runInterval = 30 * 1000;        // Intervalo de tempo desejado entre as execuções (30 segundos)
const unsigned long sleepInterval = 30 * 1000;      // Intervalo de tempo para o modo sleep (30 segundos)

unsigned long lastAlertTime = 0;                    // Armazena o tempo do último alerta
const unsigned long alertInterval = 1 * 30 * 1000;  // Intervalo desejado entre alertas (1 minuto)



// DADOS DE LOGIN NO WIFI
const char* rede = "IG_102_2G";                                         // Nome da Rede
const char* senha = "gi080615";                                         // Senha da Rede
#define BOT_TOKEN "6625058528:AAEfTGxxzstZabMOqvv6ETCXpA85YF_ioxA"      // Token utilizado para o envio de mensagens para o bot do telegram

// DADOS DE LOGIN NO WIFI
//const char* rede = "Redmi Note 8";                                    // Nome da Rede
//const char* senha = "12345678";                                       // Senha da Rede
//#define BOT_TOKEN "6625058528:AAEfTGxxzstZabMOqvv6ETCXpA85YF_ioxA"    // Token utilizado para o envio de mensagens para o bot do telegram


#define CHAT_ID "2004148826"                        // ID do chat no Telegram


X509List cert(TELEGRAM_CERTIFICATE_ROOT);           // Lista de certificados para comunicação segura com o servidor do Telegram

WiFiClientSecure client;                            // Cliente WiFi seguro para comunicação segura
UniversalTelegramBot bot(BOT_TOKEN, client);        // Objeto para manipular o bot do Telegram

int endereco = 0;                                   // Endereço na EEPROM para armazenar leituras



// CRIA O WEBSERVER

AsyncWebServer server(80);


//FUNÇÃO DE LEITURA DE DISTÂNCIA DO SENSOR PARA A WEBPAGE

String getDistance() {

    digitalWrite(D6, LOW);                          //SETA O PINO 6 COM UM PULSO BAIXO "LOW"
    delayMicroseconds(2);                           //INTERVALO DE 2 MICROSSEGUNDOS
    digitalWrite(D6, HIGH);                         //SETA O PINO 6 COM PULSO ALTO "HIGH"
    delayMicroseconds(10);                          //INTERVALO DE 10 MICROSSEGUNDOS
    digitalWrite(D6, LOW);                          //SETA O PINO 6 COM PULSO BAIXO "LOW" NOVAMENTE

    //FUNÇÃO RANGING, FAZ A CONVERSÃO DO TEMPO DE RESPOSTA DO ECHO EM CENTIMETROS, E ARMAZENA NA VARIAVEL "distancia"
```

```
95
96    distancia = ultrasonic.Ranging(CM);                                                    // Variável global recebe o valor da distância medida
97    distancia2 = distancia / 100;                                                          // Converte a distância de cm para m
98    altura1 = altura2 - distancia2;                                                        // Variável "distância" é recalculada com base na altura da caixa
99    volume = ((3.14 * altura1) / 3) * ((raio1 * raio1) + (raio1 * raio2) + (raio2 * raio2));     // Cálculo do volume da caixa d'água
100   volume = volume * 1000;                                                                // Converte de m³ para L
101   result = String(altura2);                                                              // Variável global do tipo String recebe a distância (convertido de inteiro para String)
102   result2 = String(volume);                                                              // Variável global do tipo String recebe o volume
103
104   return result2;                                                                  // Retorna o valor do volume
105   }
106
107
108
109 void setup () {
110
111   // INICIA A PORTA SERIAL
112
113   Serial.begin (115200);
114
115   // INICIA O SPIFFS, PARA CARREGAR OS DADOS DA PÁGINA WEB
116
117   if (! SPIFFS.begin ()) {
118     Serial.println ("Ocorreu um erro ao montar o SPIFFS");
119     return;
120   }
121
122   // CONEXÃO DO WIFI
123
124   WiFi.begin(rede, senha);
125   client.setTrustAnchors(&cert);
126   while (WiFi.status() != WL_CONNECTED)             //Aguarda a conexão
127   {
128     Serial.print("Estabelecendo conexão com ");
129     Serial.println(WiFi.SSID());                    //Imprime o nome da Rede
130     delay(500);
131   }
132   Serial.print("Conectado a rede! Endereco IP ESP -> ");
133   Serial.println(WiFi.localIP());                   //Imprime o IP local da ESP, que exibirá a página WEB
134
135   ip = WiFi.localIP().toString();
136
137
138    // CRIA A ASSOCIAÇÃO ENTRE O HTML E AS FUNÇÕES UTILIZADAS( MANDA AS LEITURAS PARA A PÁGINA)
139
140   server.on("/level", HTTP_GET, [] (AsyncWebServerRequest * request){          // Envia o valor do volume para ser mostrado no recurso visual da caixa d'água
141     request-> send (200, "text/plain", String(volume));
142     });
143
144   server.on ("/", HTTP_GET, [] (AsyncWebServerRequest * request) {             // Envia o arquivo index.html para ser carregado na página
145     request-> send (SPIFFS, "/index.html");
146   });
147   server.on ("/distance", HTTP_GET, [] (AsyncWebServerRequest * request) {     // Envia os valores calculado em getDistance para serem lidos pelo gráfico
148     request-> send_P (200, "text / plain", getDistance(). c_str ());
149   });
150
151    // INICIA O SERVIDOR
152
153   server.begin ();
154
155   configTime(0, 0, "pool.ntp.org"); // Obtém a hora UTC via NTP
156   time_t now = time(nullptr);
157   while (now < 24 * 3600)
158   {
159     Serial.print(".");
160     delay(100);
161     now = time(nullptr);
162   }
163   Serial.println(now);
164
165
166 }
167 void loop() {
168
169
170   int horaAtual = hour();
171   int minutoAtual = minute();
172
173   // REALIZA 10 LEITURAS ANTES DE ENTRAR NO MODO SLEEP
174
175   for (int i = 0; i < 10; i++) {
176     hcsr04();
177
178     Serial.print("Distancia ");
179     Serial.print(result);
180     Serial.println("m");
181
182     Serial.print("Volume ");
183     Serial.print(result2);
184     Serial.println("L");
185
186     Serial.print("Altura ");
187     Serial.print(result3);
188     Serial.println("cm");
```

```
    Serial.println("cm");

    Serial.print("Leitura numero:  ");
    Serial.println(i);


    mensagem();           // Verifica comandos no Telegram a cada iteração
  }

  Serial.print("Entrando no modo sleep....");


  ESP.deepSleep(1* 30 * 1000000);    // Entra no modo sleep por 30 segundos


}

void hcsr04() {

  digitalWrite(D6, LOW);                        //SETA O PINO 6 COM UM PULSO BAIXO "LOW"
  delayMicroseconds(2);                         //INTERVALO DE 2 MICROSSEGUNDOS
  digitalWrite(D6, HIGH);                       //SETA O PINO 6 COM PULSO ALTO "HIGH"
  delayMicroseconds(10);                        //INTERVALO DE 10 MICROSSEGUNDOS
  digitalWrite(D6, LOW);                        //SETA O PINO 6 COM PULSO BAIXO "LOW" NOVAMENTE

  // Função Ranging converte o tempo de resposta do echo em centímetros e armazena na variável "distancia"

  distancia = ultrasonic.Ranging(CM);                                                   // Variável global recebe o valor da distância medida
  result3 = String(distancia);                                                          // Variável global recebe o valor da distância (convertido de float para String)
  distancial = distancia / 100;                                                         // Converte a distância de cm para m
  altural = altura2 - distancial;                                                       // Variável "distância" é recalculada com base na altura da caixa
  volume = ((3.14 * altural) / 3) * ((raio1 * raio1) + (raio1 * raio2) + (raio2 * raio2));     // Cálculo do volume da caixa d'água
  volume = volume * 1000;                                                               // Converte de m³ para L
  result = String(altura2);                                                             // Variável global do tipo String recebe a distância (convertido de inteiro para String)
  result2 = String(volume);                                                             // Variável global do tipo String recebe o volume


  int volume2 = int(volume);      // Converte para int antes de salvar na EEPROM

  // SALVA AS LEITURAS NA EEPROM

  EEPROM.begin(512);
  EEPROM.put(endereco, volume2);
  endereco += sizeof(volume2);
  EEPROM.end();


  // FUNÇÕES DE AVISO DE VOLUME ACIMA E TRANSBORDA

  if (volume >= (capacidade + capacidade * 0.1)) {                                  // Verifica se o volume está acima de 10% da capacidade
    if (millis() - lastAlertTime >= alertInterval || lastAlertTime == 0) {          // Verifica se tempo suficiente passou desde o último alerta ou se é o primeiro alerta
      bot.sendMessage(CHAT_ID, "Perigo de transbordar " + result2 + " Litros", ""); // Envia a mensagem de alerta para volume maior que 10%
      lastAlertTime = millis();                                                     // Atualiza o timestamp do último alerta
    }
  } else if (volume >= (capacidade + capacidade * 0.05)) {                          // Verifica se o volume está acima de 5% e abaixo de 10%
    if (millis() - lastAlertTime >= alertInterval || lastAlertTime == 0) {          // Verifica se tempo suficiente passou desde o último alerta ou se é o primeiro alerta
      bot.sendMessage(CHAT_ID, "Volume acima do normal " + result2 + " Litros", ""); // Envia a mensagem de alerta para volume acima do normal (entre 5% e 10%)
      lastAlertTime = millis();                                                     // Atualiza o timestamp do último alerta
    }
  }
}




// FUNÇÃO DE CÁLCULO DO VOLUME MÉDIO DA CAIXA D'ÁGUA, DAS LEITURAS ARMAZENADAS NA EEPROM

  int nivelMedio() {

  int volumeTotal = 0;              // Variável para armazenar o somatório dos volumes lidos
  int contagemLeituras = 0;         // Contador de leituras armazenadas

  EEPROM.begin(512);


  for (int i = 0; i < endereco; i += sizeof(int)) {     // Loop para percorrer as leituras armazenadas na EEPROM
    int volumeLido;
    EEPROM.get(i, volumeLido);                          // Obtém o volume lido da EEPROM
    volumeTotal += volumeLido;                          // Adiciona o volume à soma total
    contagemLeituras++;                                 // Incrementa o contador de leituras
  }

  EEPROM.end();


  if (contagemLeituras > 0) {                // Calcula o nível médio com base nas leituras armazenadas
    return volumeTotal / contagemLeituras;   // Retorna a média dos volumes lidos
  }

  return 0;                                  // Se não houver leituras, retorna 0.
}
```

```
// FUNÇÃO DE MENSAGENS DO TELEGRAM

void mensagem() {

  int Bot_mtbs = 1000;                                                    // Definição do tempo mínimo entre verificações de mensagens do bot
  long Bot_lasttime;


  if (millis() > Bot_lasttime + Bot_mtbs) {                               // Verifica se tempo suficiente passou desde a última verificação
    // Obtém o número de novas mensagens
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);


    while (numNewMessages) {                                              // Processa as mensagens recebidas
      for (int i = 0; i < numNewMessages; i++) {
        String chat_id = String(bot.messages[i].chat_id);
        String text = bot.messages[i].text;
        String from_name = bot.messages[i].from_name;


        if (text == "Nivel") {                                            // Verifica se a mensagem é "Nivel"
          bot.sendMessage(CHAT_ID, "Volume: " + result2 + " Litros", ""); // Envia a mensagem com o volume atual
        }


        if (text == "Media") {                                                  // Verifica se a mensagem é "Media"
          int nivel = nivelMedio();                                             // Calcula o nível médio e envia a mensagem
          String nivelLeitura = String(nivel);
          bot.sendMessage(CHAT_ID, "Nível Médio: " + nivelLeitura + " Litros", "");   // Envia a mensagem com a média calculada
        }
        if (text == "Ip"){
          bot.sendMessage(CHAT_ID, "Bem vindo! Clique nesse link " + ip + " e acesse o monitoramento em tempo real!", "");
        }

      }


      numNewMessages = bot.getUpdates(bot.last_message_received + 1);               // Obtém o número de novas mensagens novamente
    }


    Bot_lasttime = millis();                                                // Atualiza o timestamp da última verificação
  }
}
```

ANEXO II – CÓDIGO HTML PARA A PÁGINA WEB

```html
<!DOCTYPE HTML>
<html>

<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="https://code.highcharts.com/highcharts.js"></script>
    <script type="text/javascript" src="http://static.fusioncharts.com/code/latest/fusioncharts.js"></script>
    <script type="text/javascript" src="http://static.fusioncharts.com/code/latest/themes/fusioncharts.theme.fint.js?cacheBust=56"></script>
</head>

<body style="background-color: black;">
    <h2 style="color: white; text-align: center; margin-top: 50px;">Volume da Caixa D'agua</h2>

    <div style="margin: auto; width: 500px;">
        <div id="chart-container" style="margin: 0 auto;"></div>
        <script type="text/javascript">
            FusionCharts.ready(function() {
                var fusioncharts = new FusionCharts({
                    "type": "cylinder",
                    "dataFormat": "json",
                    "id": "fuelMeter",
                    "renderAt": "chart-container",
                    "width": "500",
                    "height": "500",
                    "dataSource": {
                        "chart": {
                            "theme": "fint",
                            "caption": "Nivel de Agua",
                            "subcaption": "em Litros",
                            "lowerLimit": "0",
                            "upperLimit": "500",
                            "lowerLimitDisplay": "0",
                            "upperLimitDisplay": "500",
                            "numberSuffix": " litros",
                            "showValue": "1",
                            "chartBottomMargin": "25"
                        },
                        "value": "0"
                    }
                }).render("chart-container");

                setInterval(function() {
                    var xhttp = new XMLHttpRequest();
                    xhttp.onreadystatechange = function() {
                        if (this.readyState == 4 && this.status == 200) {
                            var volume = parseFloat(this.responseText);
                            fusioncharts.feedData("&value=" + volume);
                        }
                    };
                    xhttp.open("GET", "/level", true);
                    xhttp.send();
                }, 1000);
```

```
            });
        </script>
    </div>

    <div style="margin-top: 50px;"></div>

    <div id="chart-distance" style="width: 500px; height: 500px; margin: 0 auto;"></div>
    <script>
        var chartT = new Highcharts.Chart({
            chart: { renderTo: 'chart-distance' },
            title: { text: 'Grafico Volume / Hora' },
            series: [{
                showInLegend: false,
                data: []
            }],
            plotOptions: {
                line: {
                    animation: false,
                    dataLabels: { enabled: true }
                },
                series: { color: '#059e8a' }
            },
            xAxis: {
                type: 'datetime',
                dateTimeLabelFormats: { second: '%H:%M:%S' }
            },
            yAxis: {
                title: { text: 'Volume (L)' }
            },
            credits: { enabled: false }
        });
        setInterval(function() {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
                    var x = (new Date()).getTime()- 3 * 60 * 60 * 1000;
                        y = parseFloat(this.responseText);
                    //console.log(this.responseText);
                    if (chartT.series[0].data.length > 40) {
                        chartT.series[0].addPoint([x, y], true, true, true);
                    } else {
                        chartT.series[0].addPoint([x, y], true, false, true);
                    }
                }
            };
            xhttp.open("GET", "/distance", true);
            xhttp.send();
        }, 3000);
    </script>
</body>

</html>
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, Joao Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

Trabalho de Conclusão de Curso

Assunto:	Trabalho de Conclusão de Curso
Assinado por:	Gabriel Belizario
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Gabriel Belizário Alves**, **ALUNO (20141610411) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA**, em 27/12/2023 13:04:31.

Este documento foi armazenado no SUAP em 27/12/2023. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1039417
Código de Autenticação: 716fad1485