UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO" FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

TRABALHO DE GRADUAÇÃO

PROJETO DE UM SISTEMA BASEADO EM ARDUINO PARA MONITORAMENTO DE NÍVEL DE ÁGUA EM BARRAGENS HIDRELÉTRICAS

Aluno: Eduardo Henrique Pescaroli
Orientador: Prof. Dr. Carlos Antonio Alves

Ilha Solteira
2024

EDUARDO HENRIQUE PESCAROLI

PROJETO DE UM SISTEMA BASEADO EM ARDUINO PARA MONITORAMENTO DE NÍVEL DE ÁGUA EM BARRAGENS HIDRELÉTRICAS

Trabalho de graduação apresentado à Faculdade de Engenharia do Campus de Ilha Solteira – UNESP como parte dos requisitos para obtenção do grau de engenheiro eletricista.

Orientador: **Prof. Dr. Carlos Antonio Alves**

Ilha Solteira

2024

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Pescaroli, Eduardo Henrique.
P473p Projeto de um sistema baseado em arduino para monitoramento de nível de água em barragens hidrelétricas / Eduardo Henrique Pescaroli. -- Ilha Solteira: [s.n.], 2024
57 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2024

Orientador: Carlos Antônio Alves

Inclui bibliografia

1. Arduino. 2. ESP32. 3. Automação. 4. Aquisição de dados. 5. Internet das coisas.

Elaborado por Raiane da Silva Santos - CRB: 8/9999

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos seis dias do mês de dezembro do ano de dois mil e vinte e quatro, o discente ***Eduardo Henrique Pescaroli,*** matriculado sob o nº171052684, tendo como banca examinadora o seu orientador, *o Prof. Dr. Carlos Antonio Alves, o Prof. Dr. Alexandre César Rodrigues da Silva e a Profª. Drª. Suely Cunha Amaro Mantovani,* apresentou o Trabalho de Graduação intitulado **"Projeto de um Sistema Baseado em Arduino para Monitoramento de Nível de Água em Barragens Hidrelétricas",** obtendo a nota 8,0 (oito, virgula zero) conceito **APROVADO**.

Prof. Dr. Carlos Antonio Alves
- Orientador-

Eduardo Henrique Pescaroli
- Discente -

Prof. Dr. Alexandre César Rodrigues da Silva
- Membro da Banca -

Profª. Drª. Suely Cunha Amaro Mantovani
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.feis.unesp.br

EDUARDO HENRIQUE PESCAROLI

PROJETO DE UM SISTEMA BASEADO EM ARDUINO PARA MONITORAMENTO DE NÍVEL DE ÁGUA EM BARRAGENS HIDRELÉTRICAS

Trabalho de graduação apresentado à Faculdade de Engenharia do Campus de Ilha Solteira – UNESP como parte dos requisitos para obtenção do grau de engenheiro eletricista.

Orientador: **Prof. Dr. Carlos Antonio Alves**

BANCA EXAMINADORA

Orientador: Prof. Dr. Carlos Antonio Alves

Departamento de Engenharia Elétrica
Faculdade de Engenharia de Ilha Solteira
UNESP

Profa. Dra. Suely Cunha Amaro Mantovani

Departamento de Engenharia Elétrica
Faculdade de Engenharia de Ilha Solteira
UNESP

Prof. Dr. Alexandre Cesar Rodrigues da Silva

Departamento de Engenharia Elétrica
Faculdade de Engenharia de Ilha Solteira
UNESP

AGRADECIMENTOS

Agradeço primeiramente à Deus, que me abençoou com saúde e permitiu que eu buscasse pouco a pouco meus sonhos e objetivos.

À minha avó Maria, minha mãe Zeila e meu irmão Reinaldo, por todo o suporte que me deram e toda confiança depositada em mim.

À Associação Acadêmica Batera do Inferno, que foi meu abrigo de desenvolvimento pessoal, musical e profissional.

Aos meus grandes amigos de turma e banda, Allan Rodrigues, Hugo Guedes, Fayner Aciole, Guilherme Martins, Daniel Anderlini, Pedro Sismoto, Pedro Augusto, Pedro Barboni, Amanda Ricardi, Guilherme Carfora e Larissa Tallita. Aos meus amigos Thamer Leme, Gabriely Mizga, Henrique Lucchini e Lucas Di Niro pelos ensinamentos e direcionamentos na área de programação.

Aos docentes que participaram da minha formação profissional, em especial aos professores doutores Carlos Alves, Guilherme Azevedo e Jean Ribeiro.

RESUMO

Promoveu-se neste trabalho de graduação um projeto de um sistema de monitoramento de nível de água baseado em Arduino e ESP32, em conformidade com a Lei Federal nº 12.334/2010, visando garantir a segurança e estabilidade de barragens. O sistema proposto utiliza sensores ultrassônicos para medir com boa precisão o nível de água e é capaz de enviar alertas aos operadores quando níveis críticos são detectados, como situações de inundação atípica de um local ou inconformidade dos níveis de montante ou jusante do empreendimento. A implementação foi feita com um microcontrolador Arduino MEGA 2560, complementado por um módulo ESP32 com conexão Wi-Fi, permitindo envio remoto dos dados coletados para o armazenamento no banco de dados SQL e a possibilidade de visualização em tempo real. O sistema contempla avaliação em três contextos diferentes com três níveis de alerta para as regiões: montante, jusante e interior de galerias em casas de força de barragens hidrelétricas. O módulo ESP32 Wi-Fi integra o sistema a uma plataforma de banco de dados, o MySQL, para gerenciamento de dados e análise. Este projeto oferece uma solução flexível e personalizável para o monitoramento automatizado de barragens, contribuindo para a prevenção de acidentes e a gestão eficiente dos recursos hídricos. Os testes práticos realizados confirmaram o funcionamento esperado do sistema, evidenciando sua capacidade de captar com precisão as medidas dos sensores ultrassônicos e transmitir os dados de forma remota para o banco de dados SQL em tempo real. As avaliações foram conduzidas em um ambiente controlado, onde as distâncias entre o sensor e um objeto sólido próximo foram medidas, simulando condições operacionais. Embora os testes não tenham sido realizados em uma barragem hidrelétrica, os resultados obtidos indicam o potencial do sistema para aplicações em contextos reais, garantindo a confiabilidade e a eficácia do monitoramento proposto.

Palavras-chave: Aquisição de dados, ESP32, Internet das Coisas (IoT), Banco de dados, Comunicação sem fio, Automação.

ABSTRACT

This undergraduate project developed a water level monitoring system based on Arduino and ESP32, aligned with Federal Law No. 12.334/2010, aimed at ensuring the safety and stability of hydraulic structures. The proposed system utilizes ultrasonic sensors to accurately measure water levels and can alert operators when critical levels are detected, such as atypical flooding or discrepancies in upstream or downstream levels of the facility. Implementation involved an Arduino MEGA 2560 microcontroller, complemented by an ESP32 module with Wi-Fi connectivity, enabling remote transmission of collected data for storage in an SQL database and real-time visualization. The system is designed to evaluate three different contexts with three alert levels for upstream, downstream, and interior galleries in the powerhouse of hydroelectric dams. The ESP32 Wi-Fi module integrates the system with a MySQL database platform for data management and analysis. This project offers a flexible and customizable solution for automated dam monitoring, contributing to accident prevention and efficient water resource management. Practical tests confirmed the system's expected performance, demonstrating its ability to accurately capture ultrasonic sensor measurements and transmit data remotely to the SQL database in real time. Evaluations were conducted in a controlled environment, measuring distances between the sensor and a nearby solid object, simulating operational conditions. Although tests were not conducted on an actual hydroelectric dam, the results indicate the system's potential for real-world applications, ensuring the proposed monitoring's reliability and effectiveness.

Palavras-chave: Water level monitoring, SQL database, Real-time visualization, Safety, Data management, Wireless communication, Arduino.

LISTA DE ABREVIAÇÕES

3D	Three-Dimensional (Tridimensional)
ANA	Agência Nacional de Águas
ANEEL	Agência Nacional de Energia Elétrica
CAN	Alf and Vegard's RISC processor (um tipo de microcontrolador)
DBMS	Database Management System (Sistema de Gerenciamento de Banco de Dados)
DC	Direct Current (Corrente Contínua)
GND	Ground (Terra, em circuitos elétricos)
GOV	Governo
GPL	General Public License (Licença Pública Geral)
HD	Hard Drive (Disco Rígido)
HTML	HyperText Markup Language (Linguagem de Marcação de Hipertexto)
I2C	Inter-Integrated Circuit (Circuito Inter-Integrado)
IBM	International Business Machines
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
JSON	JavaScript Object Notation (Notação de Objetos JavaScript)
LED	Light Emitting Diode (Diodo Emissor de Luz)
LGPL	Lesser General Public License (Licença Pública Geral Menor)
PNSB	Política Nacional de Segurança de Barragens
PHP	PHP: Hypertext Preprocessor (originalmente Personal Home Page)
RAM	Random Access Memory (Memória de Acesso Aleatório)
RX	Receive (Receber, em comunicação de dados)

SCRIPT	Não é uma sigla, mas refere-se a um programa ou sequência de instruções
SNISB	Sistema Nacional de Informações sobre Segurança de Barragens
SPI	Serial Peripheral Interface (Interface Periférica Serial)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
SSD	Solid State Drive (Unidade de Estado Sólido)
TPLINK	Não é uma sigla, mas uma marca de equipamentos de rede
TX	Transmit (Transmitir, em comunicação de dados)
UART	Universal Asynchronous Receiver/Transmitter (Transmissor/Receptor Assíncrono Universal)
USB	Universal Serial Bus (Barramento Serial Universal)
USACE	United States Army Corps of Engineers
USBR	United States Bureau of Reclamation
VCC	Voltage Common Collector (Tensão do Coletor Comum, em circuitos elétricos)
XAMPP	X (para qualquer sistema operacional), Apache, MySQL, PHP e Perl

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O projeto de um sistema de monitoramento de nível de água baseado em Arduino e utilizando sensores ultrassônicos surge da visualização de uma dificuldade enfrentada por grande parte dos empreendedores do ramo de barragens hidrelétricas em coletar, tratar e armazenar dados referentes ao nível de água e transformar esses dados em informações que possam ser utilizadas para uma assertiva avaliação de segurança realizada por órgãos fiscalizadores do Estado em conformidade com a Lei Federal nº 12.334, de 20 de setembro de 2010, que estabelece a Política Nacional de Segurança de Barragens (PNSB).

A Agência Nacional de Águas (ANA) define barragem como qualquer estrutura em um curso permanente ou temporário de água para fins de contenção ou acumulação de substâncias líquidas ou misturas de líquidos e sólidos, compreendendo o barramento e as estruturas associadas (ANA, 2013). Como são diversos os tipos de barragens existentes, os projetos, materiais e estruturas envolvidas também são distintos, dificultando padronizações para monitoramento de condições indesejadas que podem gerar impactos econômicos, sociais e ecológicos.

Segundo dado publicado no relatório anual de segurança de barragens de 2021, o Brasil possuía aproximadamente 22700 barragens cadastradas por 33 órgãos fiscalizadores no Sistema Nacional de Segurança de Barragens (SNISB). Desse número, apenas metade possui informações sobre o empreendedor, 87% possuem informações sobre capacidade, 54% sobre altura e somente 51% possuem informações sobre autorização, outorga ou licenciamento. Em barragens de acúmulo de água para fins de geração de energia elétrica, é extremamente importante o monitoramento do nível de água montante (acima da barragem) e jusante (abaixo da barragem).

O monitoramento e controle de nível de água é uma atividade essencial para garantir a segurança de barragens. A Lei Federal nº 12.334, supracitada, determina que os empreendimentos com barragens devem implementar sistemas de monitoramento para garantir a segurança e a estabilidade das estruturas, expresso em seu Título IV, artigo 6º, incisos IX e X.

Um dos sistemas de monitoramento para barragens são as estações hidrométricas. Segundo Jaccon & Cudo (1989), uma estação hidrométrica,

responsável por obter descargas instantâneas ou médias diárias de um curso d'água, possui sempre uma escala limnimétrica, que é considerado o único elemento permanente e estritamente indispensável. A seção de réguas é a seção vertical que compreende o elemento inferior da escala, comumente utilizadas para medir a variação de nível d'água, são fixadas no perfil montante e jusante da barragem, onde um profissional realiza as leituras periodicamente.

Um resultado de investigações sobre rupturas catastróficas de barragens publicado no livro "Dams and Public Safety", mostra que 23% das rupturas são causadas por galgamento das barragens em função de eventos hidrológicos excepcionais subavaliados (USBR, 1983). Esse tipo de ocorrência (galgamentos) pode sinalizar diferentes tipos de falhas, seja no monitoramento de nível montante, nos dispositivos do órgão extravasor ou outros mecanismos envolvidos em controle de nível do reservatório. No ano de 1999, na Venezuela, a barragem de terra de El Guapo, por incapacidade de vazão dos dispositivos do sistema do órgão extravasor, rompeu após um galgamento (USACE - U.S. Army Corps of Engineers, 2014). Apresenta-se na Figura 1 a imagem aérea da barragem de El Guapo no momento de sua ruptura.

Figura 1 – El Guapo. Ruptura da barragem

Fonte: USACE, 2014

O controle do nível montante do reservatório utilizando os dados de um sistema de monitoramento baseado em um Arduino pode proporcionar maior confiabilidade,

uma vez que é possível realizar o monitoramento em tempo real, enviando informações para responsáveis técnicos em situações que podem indicar iminente perigo de galgamentos, e assim, permitir que ações preventivas sejam executadas. O Arduino é uma opção de baixo custo em comparação a outros dispositivos automatizados para monitoramento de níveis de água, podendo ser equipado com sensores que realizem as medições e que facilitem o envio de dados por meio de conexão em rede, com uma programação relativamente simples. Na seção 2.1.6 será abordado com maior riqueza de detalhes sobre o microcontrolador e todos os hardwares e softwares que são utilizados para compor o sistema do projeto proposto, com o objetivo de coletar os dados referentes ao nível de água de um reservatório.

O sistema alvo deste projeto se limita na coleta em tempo real de dados de distância em centímetros ou metros (a depender da configuração em programação) do referencial até o alvo, como por exemplo crista (topo da barragem) e o nível de água montante. Outro processo importante é o tratamento dos dados que são obtidos em tempo real, que podem conter amostras comprometidas por fatores ambientais e aleatórios (como por exemplo um pássaro passar entre o sensor e o alvo no momento exato da medida), e por isso é necessário para garantir acurácia na avaliação, um tratamento e modelagem dos dados obtidos, que não faz parte do escopo do projeto deste trabalho, mas que será mencionado para fins de sugestão de trabalhos futuros e para auxiliar na compreensão do tema.

A linha de pensamento ao construir um sistema baseado em automação de um processo passa pela visualização e entendimento de um problema ou processo repetitivo, objetivando algum tipo de vantagem em solucionar ou melhorar o processo. No presente trabalho o processo de realizar medições periódicas em instrumentos como régua limnimétricas para monitorar o nível de água de uma barragem hidrelétrica, provocou a ideia de automatizar utilizando um microcontrolador e conseguir coletar uma maior quantidade de dados com os sensores para que o resultado obtido fosse o ganho de tempo, monetário e dados que possam ser traduzidos e interpretados para desenvolvimento de planos estratégicos para o negócio. Nos próximos tópicos serão abordados sobre cada software e hardware que é utilizado para compor o sistema, e também, como o conjunto integralizado consegue entregar o resultado desejado, proporcionando melhoria para o processo, com exemplos de simulações e testes práticos.

Para uma compreensão mais abrangente e precisa do sistema proposto, faz-se necessário, a título de introdução, abordar os conceitos fundamentais e os objetivos primordiais das ferramentas empregadas no desenvolvimento do projeto. A integração dessas ferramentas oferece um vasto espectro de soluções de automação em diversos campos de aplicação, possibilitando a padronização de procedimentos e a otimização de processos. Esta abordagem multidisciplinar não apenas enriquece o escopo do projeto, mas também proporciona uma base sólida para a implementação de sistemas de monitoramento eficazes e adaptáveis a diferentes contextos operacionais.

1.1 OBJETIVO GERAL

Desenvolver um sistema de monitoramento de nível de água utilizando Arduino e sensores ultrassônicos, com o propósito de coletar dados em tempo real para avaliação da segurança de barragens, em conformidade com a Política Nacional de Segurança de Barragens (PNSB).

1.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos deste trabalho têm-se:

I. Implementar um sistema de coleta de dados em tempo real.
II. Garantir a precisão dos dados coletados.
III. Armazenar os dados coletados em um banco de dados para análise.
IV. Contribuir para a segurança das barragens integrando tecnologia e automação em tarefas rotineiras.

2 MATERIAIS UTILIZADOS

A lista de materiais desse projeto pode ser dividida em duas partes, visto que a parte de testes também se dividiu de maneira igual, em simulação e testes práticos. Contudo, alguns tópicos teóricos sobre os hardwares e softwares utilizados são relevantes para o completo entendimento do projeto.

2.1 BANCOS DE DADOS RELACIONAIS, MYSQL E PHP

2.1.1 BANCOS DE DADOS RELACIONAIS

Um banco de dados é essencialmente uma coleção sistematizada de informações estruturadas, geralmente armazenadas em sistemas eletrônicos como computadores, celulares e tablets. Os dados são frequentemente organizados em linhas e colunas dentro de tabelas ou conjuntos de tabelas, o que facilita o processamento e a consulta, tornando a pesquisa ou análise mais eficiente.

Há uma variedade de tipos de bancos de dados disponíveis para usuários individuais ou grupos, incluindo os bancos de dados relacionais, que são os mais comuns e se estruturam em tabelas com colunas e linhas. Os bancos de dados orientados a objetos, por outro lado, representam dados na forma de objetos, em linha com a programação orientada a objetos. Já os bancos de dados distribuídos consistem em dois ou mais arquivos localizados em diferentes fontes, podendo estar hospedados em múltiplos computadores, não necessariamente no mesmo local físico. Os *data warehouses* são coleções centralizadas de dados, projetadas especificamente para facilitar consultas e análises. Os bancos de dados *NoSQL*, que não são relacionais, permitem o armazenamento e a manipulação de dados não estruturados e semiestruturados, sendo especialmente úteis para dados complexos provenientes de aplicativos *web*. Além desses, existem outros modelos, como bancos de dados multimodal, em nuvem, *JSON*, autônomos, entre outros.

2.1.2 INTERNET DAS COISAS (IoT)

A Internet das Coisas, termo derivado da língua inglesa *Internet of things* (IoT) refere-se à interconexão de dispositivos físicos através da internet, permitindo que eles coletem, compartilhem e processem dados. Esses dispositivos, que podem variar de eletrodomésticos a sensores com aplicações industriais e empresariais, são equipados com tecnologia que lhes permite comunicar-se entre si e com sistemas centrais, sem a necessidade de intervenção humana direta. A IoT transforma objetos do cotidiano em "inteligentes", possibilitando automação e controle remoto.

A importância da IoT se instala em sua capacidade de otimizar processos, aumentar a eficiência e melhorar a qualidade de vida. Em ambientes industriais, a IoT facilita a manutenção preditiva, reduzindo o tempo de inatividade e os custos operacionais. Em processos inteligentes, a IoT melhora a gestão de recursos, como energia e água, e aprimora a alocação de recursos (mão de obra, ferramentas ou monetário) através de uma base sólida de dados para análise precisa do desempenho de cada processo.

Além disso, a IoT impulsiona a inovação, criando novas oportunidades de negócios e modelos de serviço. Ao fornecer dados em tempo real, a IoT permite uma tomada de decisão mais informada e ágil, essencial em um mundo cada vez mais dinâmico e interconectado. Com o crescimento contínuo da IoT, espera-se que sua influência se expanda ainda mais, transformando setores e impactando profundamente a sociedade.

A engenharia elétrica e a engenharia de software são duas das principais formações relacionadas com a criação de soluções em IoT. A primeira é responsável por soluções mais voltadas a hardware e a segunda a softwares (DURAES, 2002). O principal desafio é interconectar essas soluções de forma eficiente.

2.1.3 MYSQL

Para gerenciar bancos de dados, é comum o uso de um sistema de gerenciamento de banco de dados (DBMS), que atua como uma interface entre o usuário final e o banco de dados. Isso possibilita a atualização, gerenciamento e recuperação de dados, facilitando tarefas de supervisão, backup e ajustes. O *MySQL Server* é um dos sistemas de gerenciamento de banco de dados mais populares. Este programa de código aberto, baseado em SQL, é ideal para aplicativos *web* e pode ser

executado em qualquer plataforma. Ele se destaca como uma ferramenta eficiente para o gerenciamento de dados em larga escala, sendo amplamente utilizado por empresas como *Uber, LinkedIn, Facebook* e *YouTube*.

A *Structured Query Language* (SQL) ou linguagem de consulta estruturada é uma linguagem de programação utilizada por muitos bancos de dados relacionais para realizar consultas, manipular dados e gerenciar controle de acessos. Segundo informações encontradas de forma fácil na página virtual da Oracle Brasil, a linguagem supracitada foi desenvolvida pela primeira vez na IBM na década de 1970, sendo a própria Oracle a principal contribuinte (ORACLE, 2024).

2.1.4 PHP

O PHP é um acrônimo recursivo para *hypertext preprocessor*, que nada mais é do que um pré-processador de hipertexto, uma linguagem de programação de grande utilização para o desenvolvimento de *websites* e aplicações *web* (PHP,2024). É uma poderosa linguagem de programação de fácil entendimento para iniciantes em geral, com uma grande gama de recursos avançados para programadores mais experientes. O PHP com poucos comandos pode mostrar HTML (*HyperText Markup Language)* que é uma linguagem de marcação de hipertexto. Em sua essência construtiva, o PHP pode ter embutido em seu código uma mescla de código HTML, que resulta em executar uma ação bem definida. Dada a sua característica de programação bem definida com símbolos de início e fim, o PHP consegue proporcionar ao usuário a possibilidade de entrar e sair do modo PHP em seu código, utilizando como ferramenta secundária para executar uma tarefa.

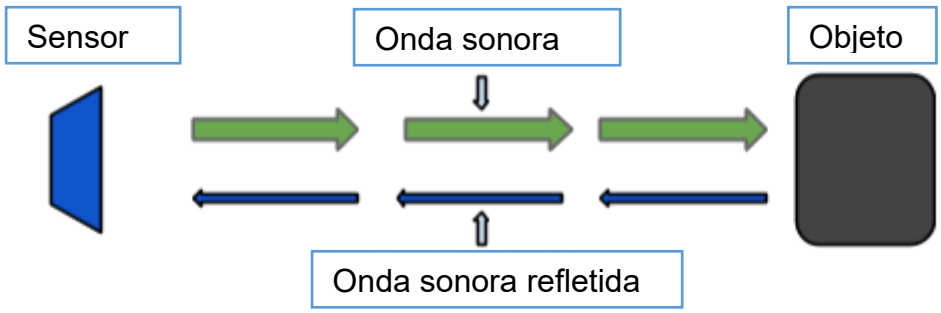
Segundo o *PHP Documentation Group*, responsáveis pela patente da linguagem, o PHP é focado principalmente nos scripts do lado do servidor, permitindo fazer tarefas como coletar dados de formulários, gerar páginas com conteúdo dinâmico ou enviar e receber cookies.

2.1.5 AQUISIÇÃO DE DADOS PARA UM SISTEMA DE MONITORAMENTO COM SENSOR ULTRASSÔNICO

Um sensor ultrassônico é um dispositivo eletrônico que possui capacidade de mensurar distâncias entre ele e o objeto mais próximo através de ondas de frequências sonoras acima do que o ser humano pode escutar (MORGAN, 2014). Na

Figura 2 é ilustrado o funcionamento de um sensor deste tipo, que emite uma onda sonora até o objeto, que reflete parte de volta para o dispositivo e assim o sensor é capaz de determinar a distância entre ele e o objeto com o tempo que a onda sonora demorou para ir e voltar.

Figura 2 – Ilustração de funcionamento do sensor ultrassônico HC-SR04



Fonte: Adaptado de MORGAN, 2014

Na Figura 3 é ilustrado a representação de um sensor HC-SR04, que possui aproximadamente 10 gramas, trigger input de 10 microssegundos, alimentação de 5 Volts em corrente contínua (DC) e uma capacidade de mensurar com alta precisão (aproximadamente 0,3 centímetros) distâncias entre 2 e 400 centímetros. Um sensor desse calibre em um processo típico é capaz de operar em uma faixa de temperatura de 40 ºC negativos à 180 ºC positivos, com pressão de até 40 bar (HAUPTMANN, 2002).

Ainda de acordo com o estudo realizado por Hauptmann e publicado em seu artigo científico "*Application of ultrasonic sensors in the process industry*", os sensores ultrassônicos possuem atributos bem atrativos como uma medição não invasiva, resposta rápida, boa precisão, baixo consumo de energia e alta resolução. Contudo, há de se considerar os fatores negativos, como complexidade no processamento digital do sinal, fácil perturbação das medições quando há bolhas de gás em líquidos e que a atenuação do som aumenta com a frequência (HAUPTMANN, 2002).

Figura 3 – Sensor ultrassônico HC-SR04

Fonte: MORGAN, 2014

Um sensor ultrassônico pode ser conectado a um microcontrolador para aquisição de dados, coletando as distâncias mensuradas e direcionando de forma extremamente rápida para ser armazenado em um banco de dados. Sua principal informação é uma medida numérica, que pode ser modelada para apresentar leituras em tempo real de determinado objeto em relação ao sensor e garantir um histórico ao longo do tempo.

2.1.6 MICROCONTROLADOR ARDUINO

O microcontrolador Arduino, que possui hardware composto por um processador Atmel AVR com suporte embutido de entrada/saída, tem como maior vantagem sobre outras plataformas de desenvolvimento de microcontroladores a facilidade de sua utilização, isso porque o seu software com linguagem de programação padrão facilita o aprendizado, permitindo implementação de projetos básicos ou avançados por pessoas sem conhecimento prévio em um curto espaço de tempo (ROBERTS, 2011)

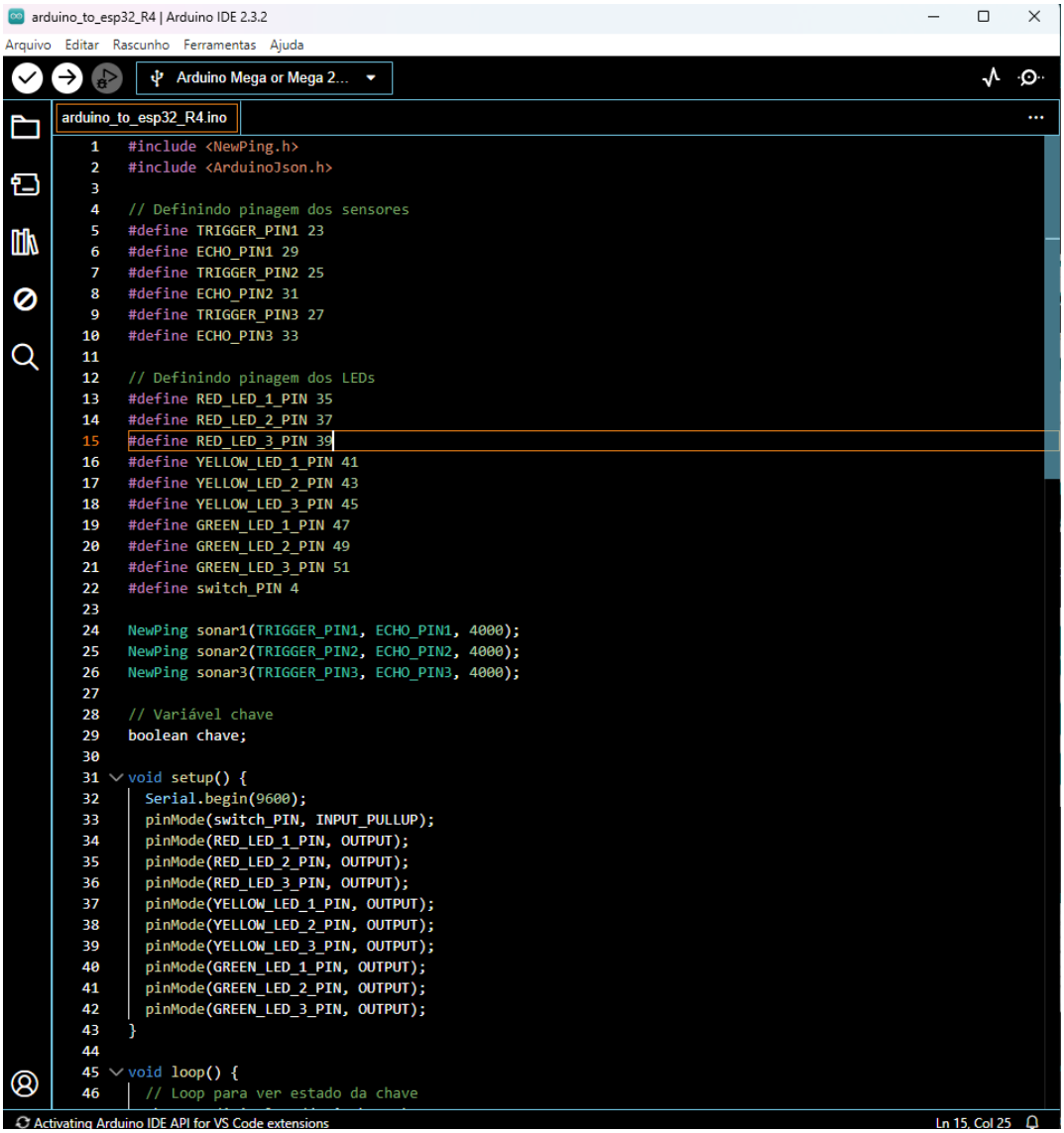
O avanço da tecnologia permitiu alcançar o patamar em que atividades repetitivas realizadas por seres humanos fossem facilitadas ou totalmente substituídas por sistemas automatizados. Conforme Roberts (2011)

> ” O Arduino pode ser utilizado para desenvolver objetos interativos independentes, ou pode ser conectado a um computador, a uma rede, ou até mesmo à Internet para recuperar e enviar dados do Arduino e atuar sobre eles. Em outras palavras, ele pode enviar um conjunto de dados recebidos de alguns sensores para um site, dados estes que poderão, assim, ser exibidos na forma de um gráfico”

O Arduino é uma plataforma aberta de prototipação eletrônica, composta pelo *hardware* (placa controladora) e *software* (ambiente de desenvolvimento integrado).

O ambiente de desenvolvimento integrado (IDE) foi desenvolvido em 2005, na Itália, e tem compatibilidade com qualquer sistema operacional sob licença GPL/LGPL (*General Public License/Lesser General Public License*) para software. A plataforma permite a rápida compilação e transferência de código em C/C++ pela interface USB para a placa controladora, de forma bem intuitiva e prática, é possível visualizar um exemplo da interface conforme Figura 4.

Figura 4 – Visualização da IDE Arduino



Fonte: Elaborado pelo autor

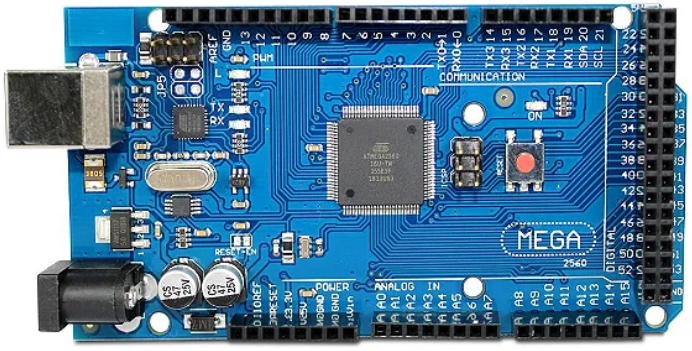
O Arduino é uma plataforma de prototipagem eletrônica que oferece diversas versões, cada uma com características específicas. Essas versões diferem em aspectos como o número de portas de entrada/saída (I/O) analógicas e digitais, opções de alimentação, módulos de conectividade e o tipo de microcontrolador empregado. Tais características permitem que o Arduino seja utilizado tanto para a leitura quanto para o processamento de dados.

Os modelos Arduino Uno e Arduino Mega 2560 são particularmente notáveis por sua capacidade de leitura analógica com resolução de 10 bits e uma taxa de amostragem de até 10.000 leituras por segundo, equivalente a uma frequência máxima de 10 kHz. Essa capacidade é mais do que suficiente para implementar sistemas de telemetria robustos, como discutido anteriormente.

As diferenças fundamentais entre o Arduino Uno e o Arduino Mega 2560 residem no tamanho da memória, na quantidade de portas de I/O disponíveis e na capacidade de processamento. O Mega 2560, por exemplo, oferece mais memória e um maior número de portas de I/O, tornando-o mais adequado para projetos que exigem maior complexidade e conectividade.

Na Figura 5 é ilustrado um microcontrolador Arduino Mega 2560, semelhante ao utilizado no projeto em questão, destacando suas características físicas e funcionais que o tornam uma escolha versátil para uma ampla gama de aplicações acadêmicas e profissionais.

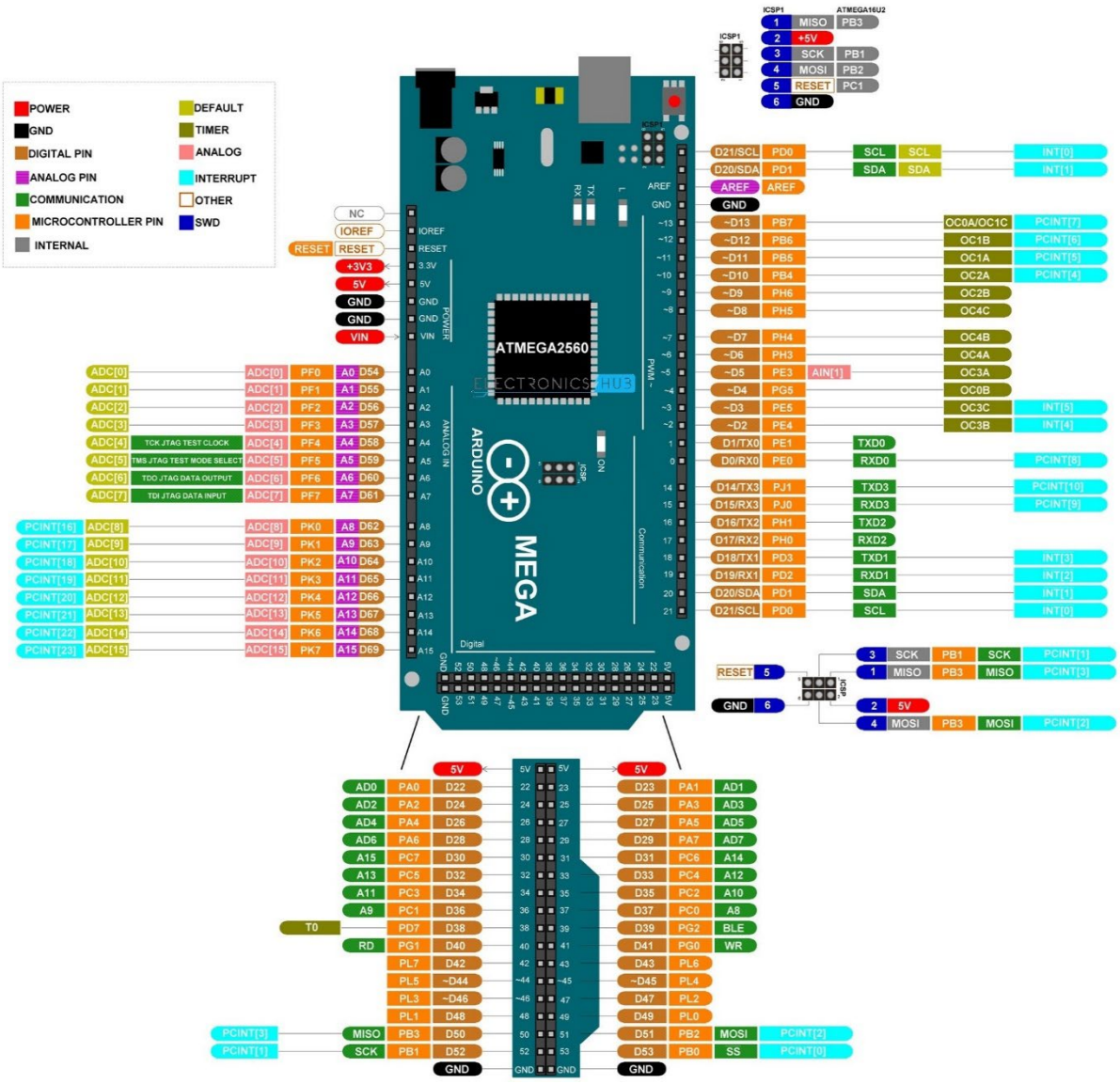
Figura 5 – Arduino MEGA



Fonte: (GTMax3D, 2024)

Na Figura 6, é ilustrada a pinagem completa de um microcontrolador Arduino MEGA 2560, para que facilite a consulta com as menções sobre conexões mencionadas no tópico 2.3 deste trabalho.

Figura 6 – Pinos de conexão do Arduino MEGA 2560



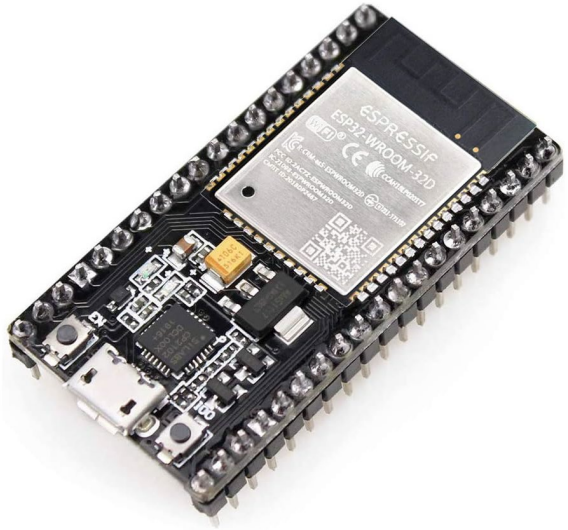
Fonte: (USINAINF, 2024)

2.1.7 MICROCONTROLADOR ESP-32

Com o avanço da tecnologia, microcontroladores ficaram cada vez mais baratos e mais simples de serem implementados, inclusive para o envio e recebimento de informações. O ESP-32 é uma placa de desenvolvimento que possui módulo de

conexão Wi-Fi e Bluetooth, além das interfaces I2C, CAN, SPI, UART, leitura de sinais analógicos e digitais, com uma confiabilidade razoável e custo de aquisição baixíssimo, que pode ser visualizado na Figura 7.

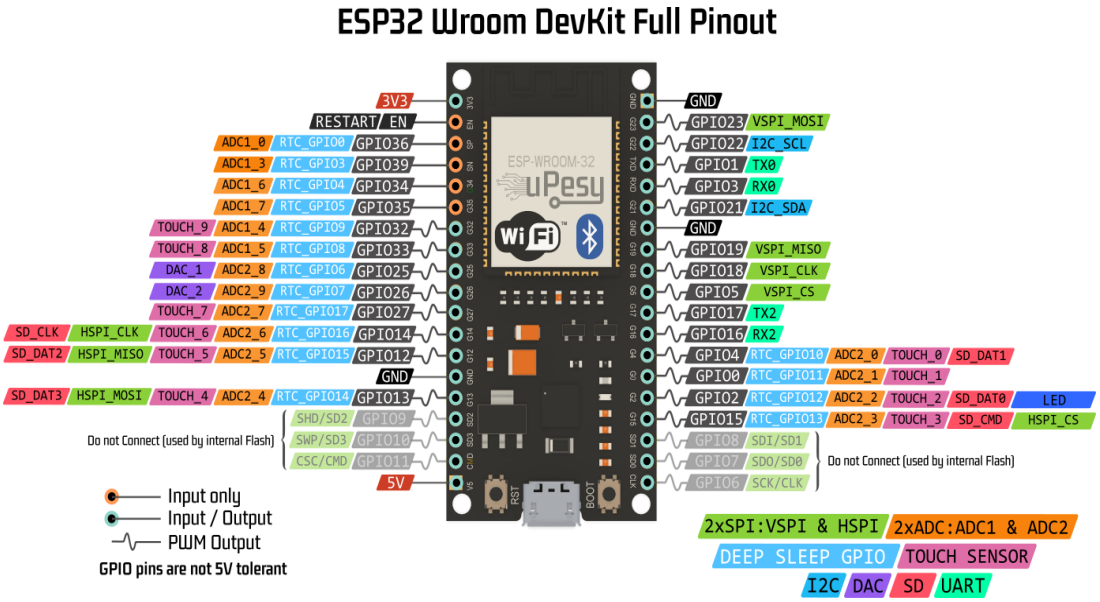
Figura 7 – Placa de desenvolvimento ESP-32



Fonte: (AMAZON, 2024)

Na Figura 8 é ilustrada a pinagem completa de um microcontrolador ESP32 idêntico ao utilizado no projeto, para que facilite a consulta com as menções sobre conexões mencionadas adiante neste trabalho.

Figura 8 – Pinagem ESP32



Fonte: (UPESY,2024)

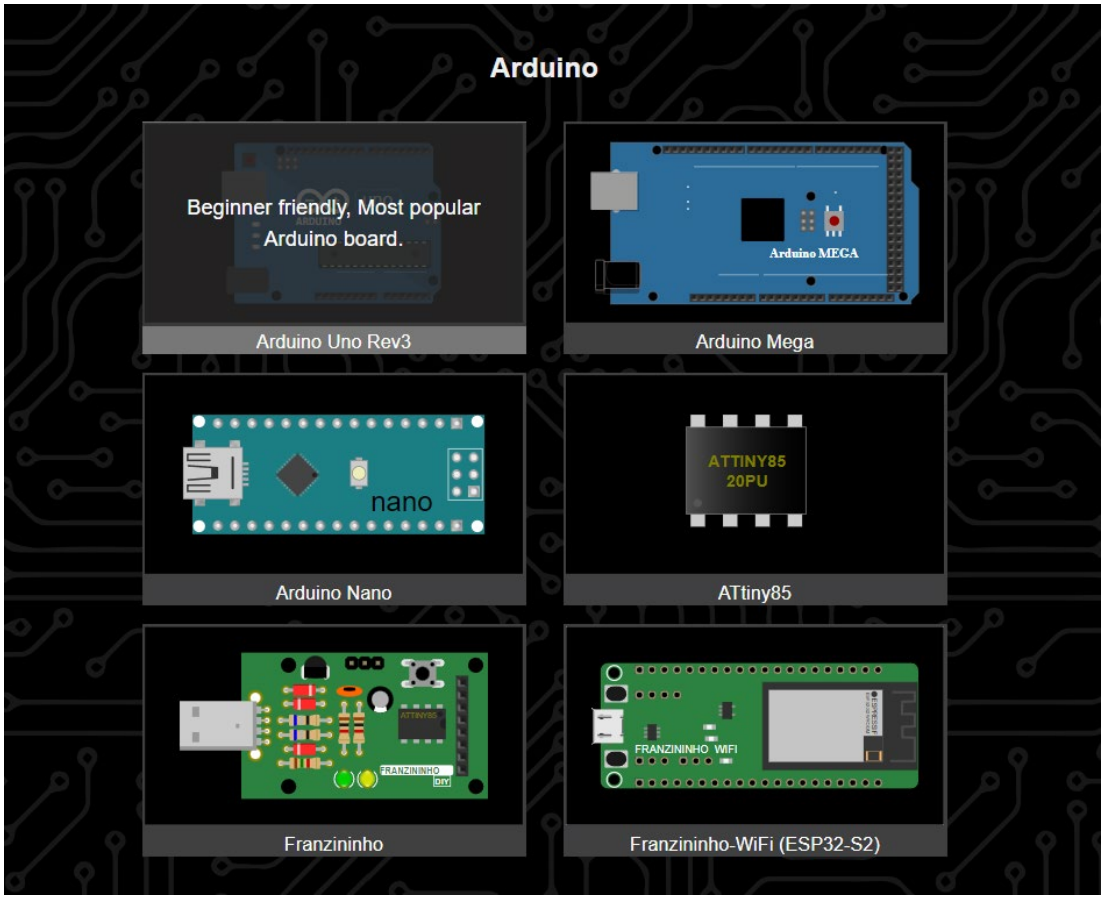
O ESP-32 é um microcontrolador com módulo Wi-Fi integrado, o que permite a transmissão de dados para servidores de maneira eficiente. Este dispositivo é equipado com 18 canais que podem ser configurados como entradas analógicas ou digitais. Ele opera com uma taxa de amostragem de 6 kHz e oferece uma resolução de 12 bits, o que é adequado para uma variedade de aplicações que exigem precisão.

Para programar o ESP-32, o ambiente de desenvolvimento integrado (IDE) do Arduino é amplamente utilizado, permitindo a compilação e o upload de códigos escritos em C/C++. Essa flexibilidade facilita a implementação de projetos básicos até complexos e a integração com outros dispositivos e sensores.

Essas características fazem do ESP-32 uma escolha versátil para projetos que necessitam de conectividade de rede e processamento eficiente de dados.

Para a simulação do sistema, utilizou-se o software online Wokwi, que possui um ambiente com diversas placas de microcontroladores para testes, permitindo uma vasta quantidade de opções. Conforme pode ser visualizado na Figura 9, é possível escolher entre algumas opções na categoria de interesse (Arduino).

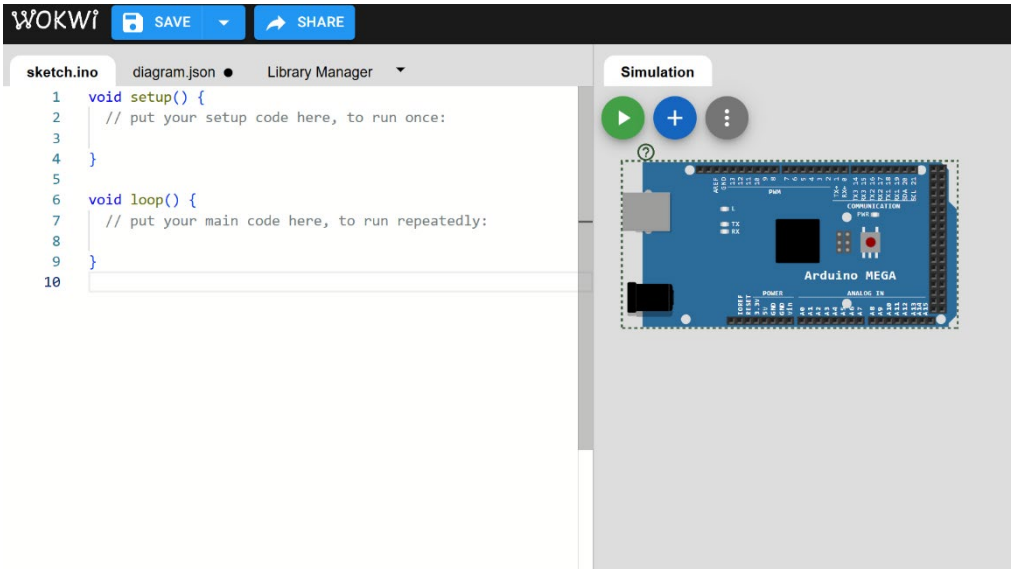
Figura 9 – Opções de escolha no software de simulação Wokwi



Fonte: Elaborado pelo autor

Escolhendo a opção de Arduino Mega, a página que o usuário é redirecionado permite a manipulação de dois modos: programação em código ou montagem do circuito, conforme Figura 10.

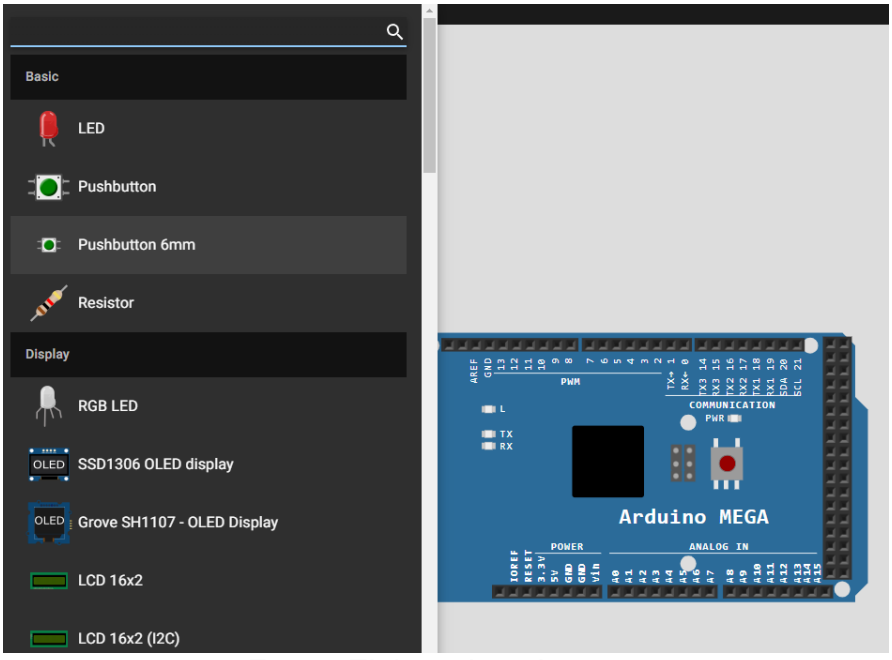
Figura 10 – Ambiente de desenvolvimento Wokwi



Fonte: Elaborado pelo autor

Na parte superior, um pouco para a direita do centro têm-se três ícones, um verde, um azul e um cinza. O símbolo verde inicia a simulação do circuito que obedecerá a programação situada à esquerda da tela. O botão no círculo azul, que parece um símbolo de soma, permite a inserção no circuito de componentes, conforme pode ser visualizado na Figura 11.

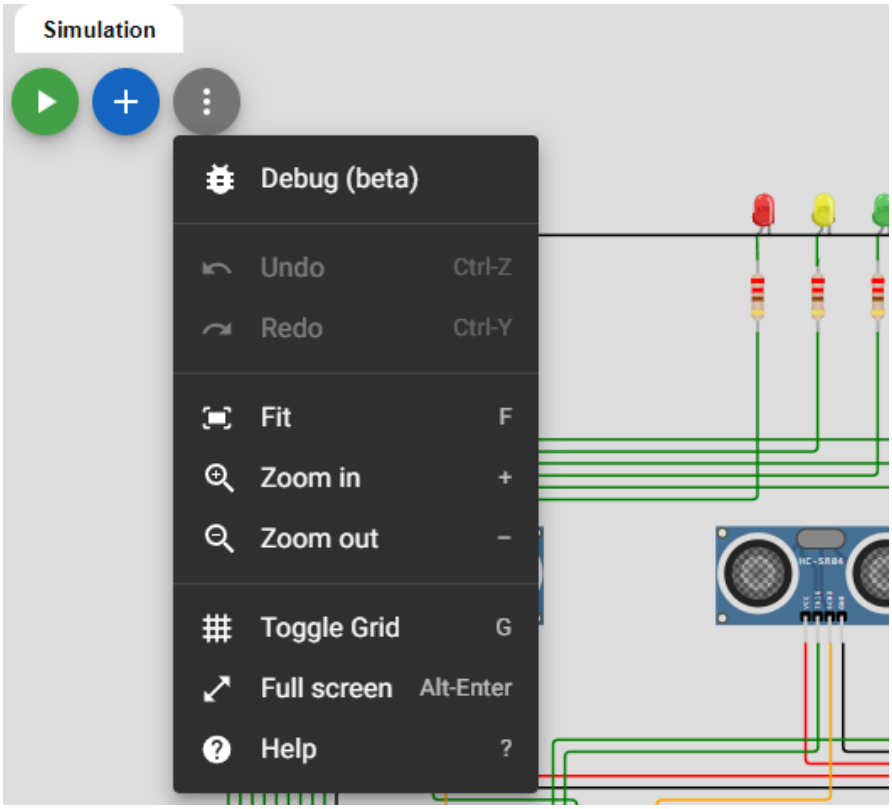
Figura 11 – Opções de componentes no ambiente de desenvolvimento Wokwi



Fonte: Elaborado pelo autor

O último botão de interação permite realizar debugs, ajustar à tela, exibir linhas de grade e outras funcionalidades para auxiliar no desenvolvimento, conforme Figura 12.

Figura 12 – Opções para o usuário no ambiente de desenvolvimento Wokwi



Fonte: Elaborado pelo autor

Com relação à parte prática, utilizou-se a seguinte lista de materiais:

- Arduino Mega 2560;
- ESP32 WROOM;
- Resistores de 1000 ohms;
- Sensores ultrassônicos HC-SR04;
- LEDs de cor vermelha 5 Vcc;
- LEDs de cor amarela 5 Vcc;
- LEDs de cor verde 5 Vcc;
- 1 Protoboard 10cm x 16 cm;
- 1 Modem Wi-Fi TPLINK 100 mbps;

- 1 Computador com configurações de processador i3 11º geração, HD SSD 512 gigabytes, 16 gigabytes de memória RAM e placa mãe ASUS ZEN PRIME;
- Fios e cabos para conexão;
- Cabo USB para alimentação do Arduino e ESP32.

3 RESULTADOS DE SIMULAÇÃO E TESTE EXPERIMENTAL

3.1 SIMULAÇÃO

Esta etapa do projeto tem por finalidade verificar utilizando o ambiente virtual Wokwi o correto funcionamento do código e a conexão dos componentes antes da implementação prática. O objetivo é garantir que o sistema funcione corretamente com componentes reais, permitindo ajustes conforme necessário.

Utilizou-se três sensores ultrassônicos para medir o nível de água. As conexões simuladas estão numericamente indicadas na frente de cada componente:

- **Trigger:** Conectados às portas digitais 23, 25 e 27 do Arduino Mega;
- **Echo:** Conectados às portas digitais 29, 31 e 22;
- **Alimentação:** VCC e GND do Arduino.

Cada sensor possui três LEDs (vermelho, amarelo e verde) para indicar o estado:

- **Vermelhos:** Portas 35, 37 e 39;
- **Amarelos:** Portas 41, 43 e 45;
- **Verdes:** Portas 47, 49 e 51.

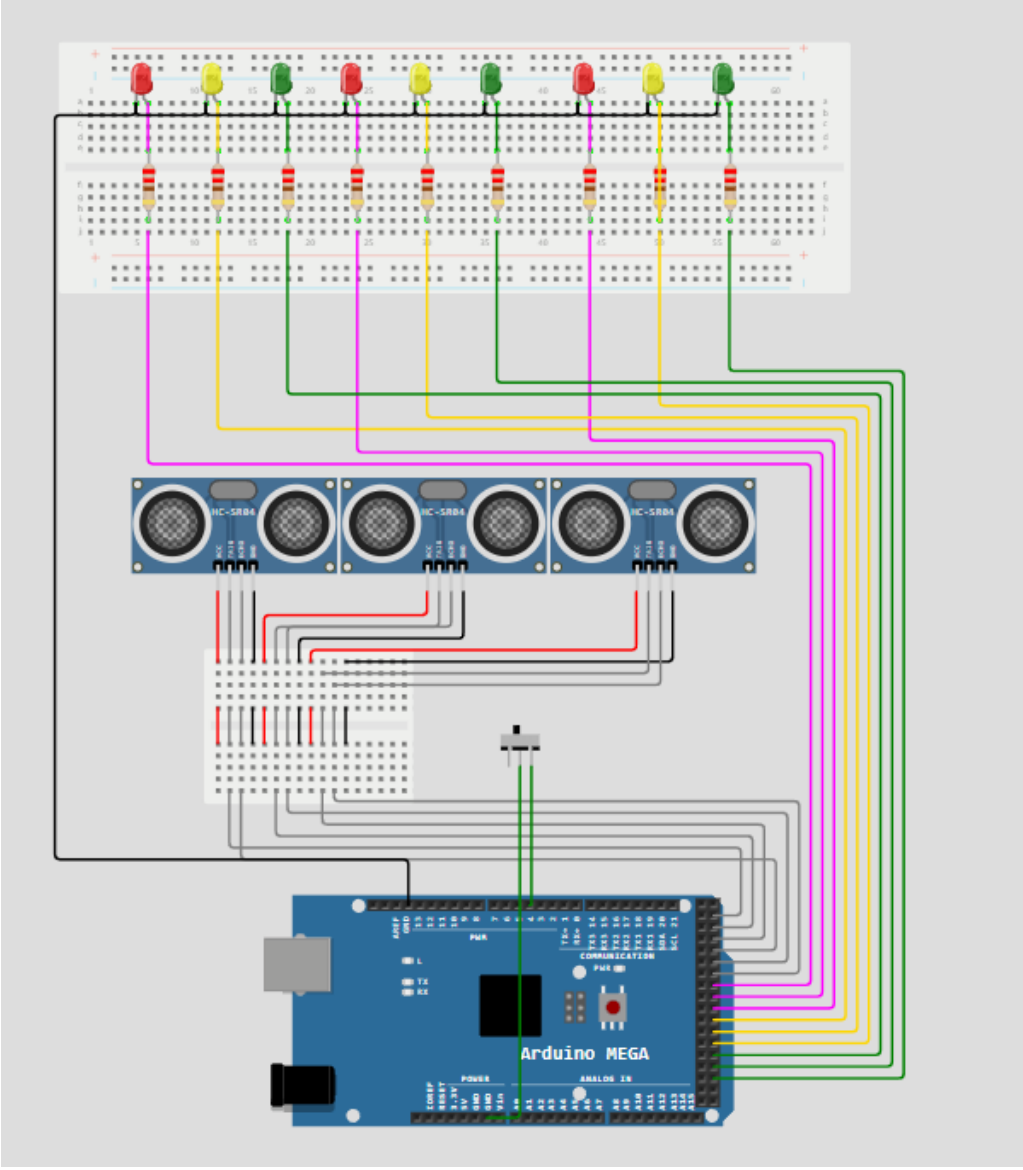
Os LEDs estão conectados a resistores para limitar a corrente que circula pelos componentes.

Colocou-se o slide-switch com o intuito de simular a verificação de conexão Wi-Fi, que chamaria uma rotina de backup interno. Contudo, o único resultado deste recurso na simulação é a sinalização do sistema estar ou não conectado ao banco de dados (por meio da rede Wi-Fi). Como na simulação não foi possível realizar esse procedimento, o slide-switch operou somente como sinalizador lógico.

- Nível lógico 1: Conexão simulada.
- Nível lógico 0: Desconexão simulada, indicando uma rotina de backup offline (não implementada).

É possível visualizar o circuito completo utilizado na simulação através da Figura 13 abaixo.

Figura 13 – Esquemático do circuito da simulação

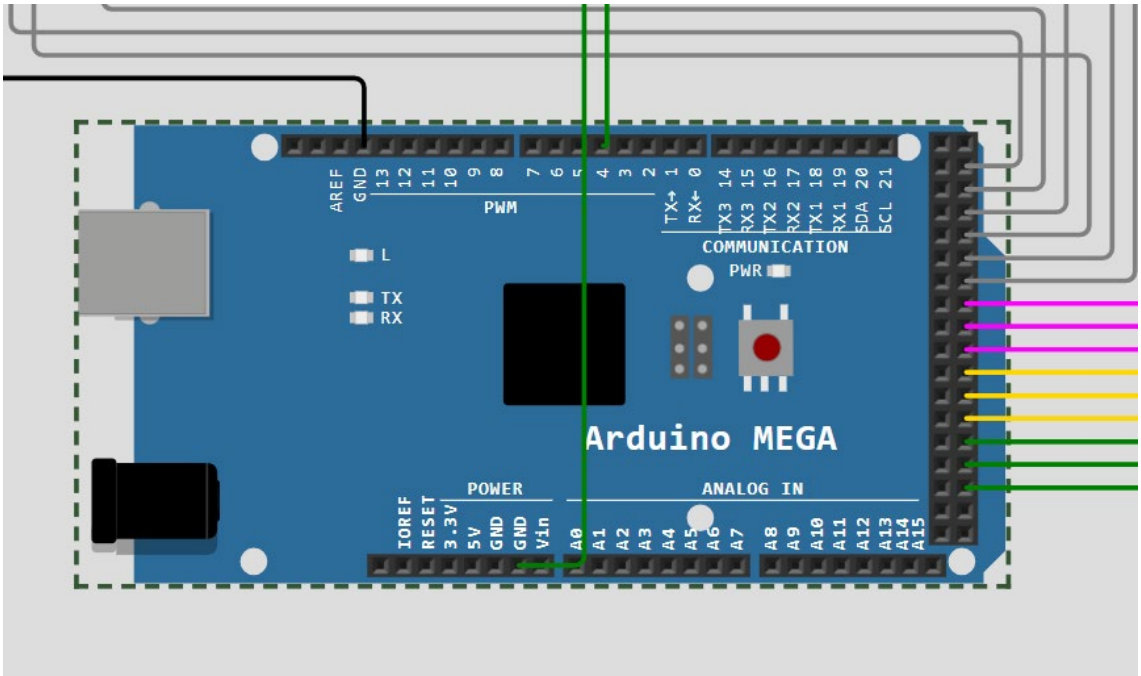


Fonte: Elaborado pelo autor

As conexões feitas no Arduino podem ser visualizadas conforme mencionadas as pinagens na Figura 14, onde os fios de cor cinza indicam as conexões do ECHO e TRIGGER dos sensores, os fios de cor rosa indicam as conexões dos anodos dos LEDs vermelho, os fios de cor amarela indicam as conexões dos LEDs de cor amarela, os fios de cor verde indicam as conexões dos LEDs de cor verde, os fios pretos

indicam as conexões dos catodos com o GND (Ground) e o verde os fios da chave slide-switch. Note que não foi mencionado os fios de alimentação, pois como se trata de uma simulação, não foi obrigatório a utilização, visto que o ambiente de simulação indica alimentação padrão dos componentes mesmo sem conexão.

Figura 14 – Vista aproximada das conexões no Arduino Mega no Wokwi



Fonte: Elaborado pelo autor

A simulação visa testar a lógica de funcionamento e a integração dos componentes. A implementação prática pode exigir ajustes nas conexões e no código para atender às condições reais e necessidades específicas do projeto.

Este ambiente virtual permite identificar e corrigir problemas antes da montagem física, garantindo maior eficiência e precisão na fase prática. A flexibilidade do projeto com simulações e teste prático (tópico seguinte) permite um leque adaptações futuras para aprimorar a funcionalidade e a robustez do sistema.

3.2 TESTE EXPERIMENTAL

O sistema proposto segue um fluxo de dados estruturado em quatro etapas principais, conforme ilustrado no fluxograma da Figura 15. O processo começa com a captação de dados por meio de sensores ultrassônicos HC-SR04, reconhecidos por sua precisão e confiabilidade em medições de distância. Esses sensores são amplamente utilizados em diversas aplicações devido à sua capacidade de fornecer leituras precisas e consistentes, o que é essencial para a integridade dos dados coletados. Em seguida, esses dados brutos são processados pelo microcontrolador Arduino Mega 2560, que possui capacidade computacional adequada para essa tarefa. O Arduino Mega 2560 é uma plataforma robusta e versátil, ideal para projetos que exigem um processamento eficiente e confiável. Ele é equipado com um microcontrolador ATmega2560, que oferece uma ampla gama de funcionalidades e suporte para múltiplos sensores e módulos.

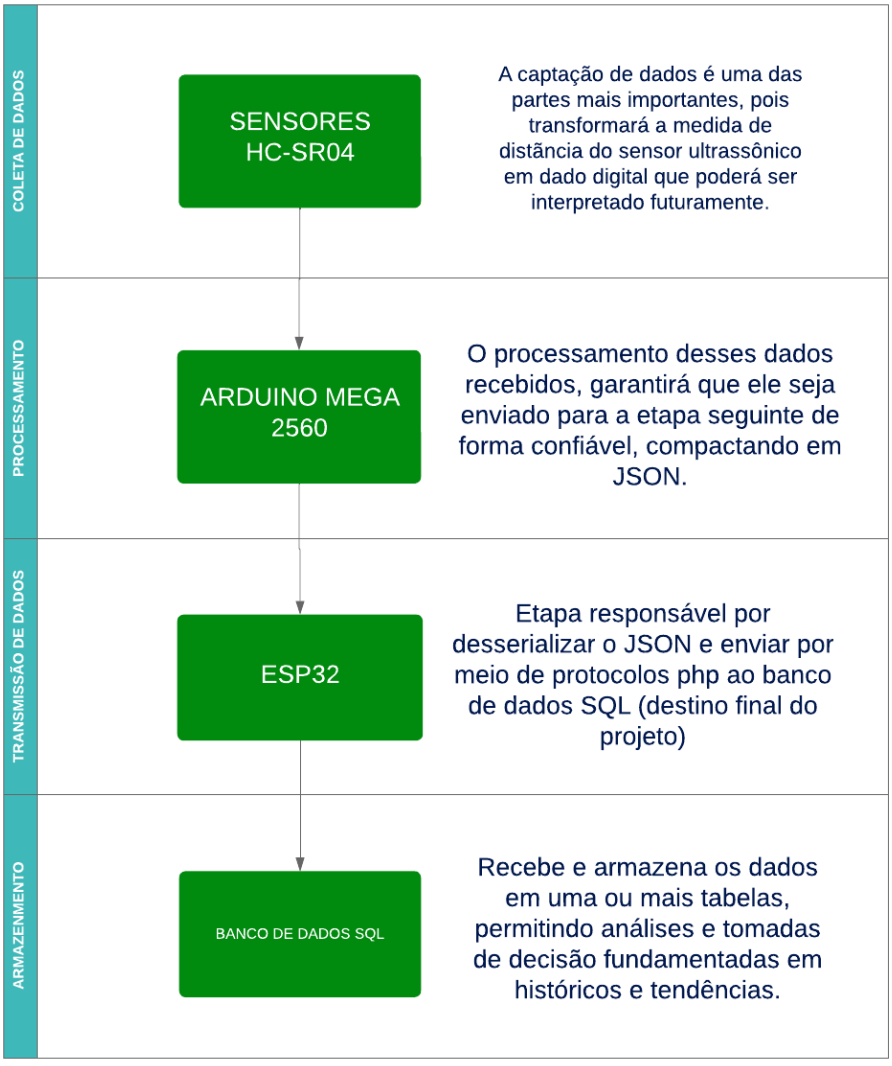
A terceira etapa envolve a transmissão dos dados processados via ESP32, um módulo de comunicação sem fio que permite a conexão com redes Wi-Fi, facilitando a transferência remota de informações. O ESP32 é conhecido por sua alta performance e baixo consumo de energia, tornando-o uma escolha excelente para aplicações de Internet das Coisas (IoT). Ele suporta tanto Wi-Fi quanto Bluetooth, proporcionando flexibilidade na comunicação e integração com outros dispositivos. A capacidade de transmitir dados sem fio é crucial para sistemas que requerem monitoramento e controle em tempo real, permitindo que os dados sejam acessados e analisados remotamente.

Finalmente, o fluxo se completa com o armazenamento dos dados em um banco de dados SQL, proporcionando uma solução robusta para a persistência e análise posterior das informações coletadas. O uso de um banco de dados SQL garante que os dados sejam armazenados de forma estruturada e segura, facilitando a recuperação e análise eficiente. Bancos de dados SQL são amplamente utilizados devido à sua capacidade de lidar com grandes volumes de dados e realizar consultas complexas de maneira rápida e eficaz.

Este fluxograma representa de forma clara a arquitetura do sistema, demonstrando a integração eficiente entre hardware de sensoriamento, processamento embarcado, comunicação sem fio e armazenamento de dados. A

integração desses componentes é fundamental para garantir que o sistema funcione de maneira coesa e eficiente. Cada etapa do processo foi cuidadosamente projetada para maximizar a precisão, confiabilidade e eficiência do sistema como um todo.

Figura 15 – Fluxo de procedimentos do projeto



Fonte: Elaborado pelo autor

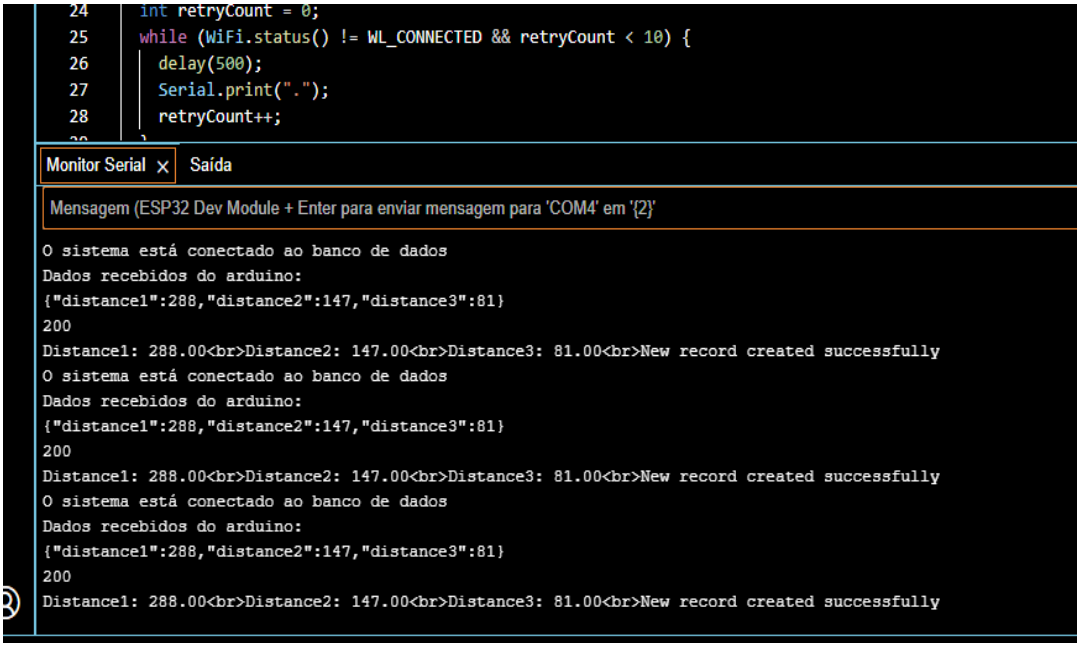
Na etapa experimental do projeto, após a conclusão da programação do Arduino durante a fase de simulação, o foco foi direcionado para o desenvolvimento da programação do ESP32 utilizando a IDE Arduino, etapa que transita entre o passo de “Processamento” e “Transmissão de dados” do fluxograma apresentado anteriormente.

O objetivo principal foi estabelecer uma comunicação eficiente entre o Arduino Mega e o ESP32. Para isso, buscou-se transferir as leituras de distância dos sensores,

que inicialmente foram exibidas no monitor serial do Arduino Mega, para o monitor serial do ESP32. A fim de assegurar a confiabilidade dessa transmissão de dados, optou-se por conectar os dois dispositivos através de uma comunicação serial pelos pinos RX (pino receptor) e TX (pino transmissor) de ambos os hardwares, utilizando cabos, e transmitir os dados dos três sensores em formato JSON.

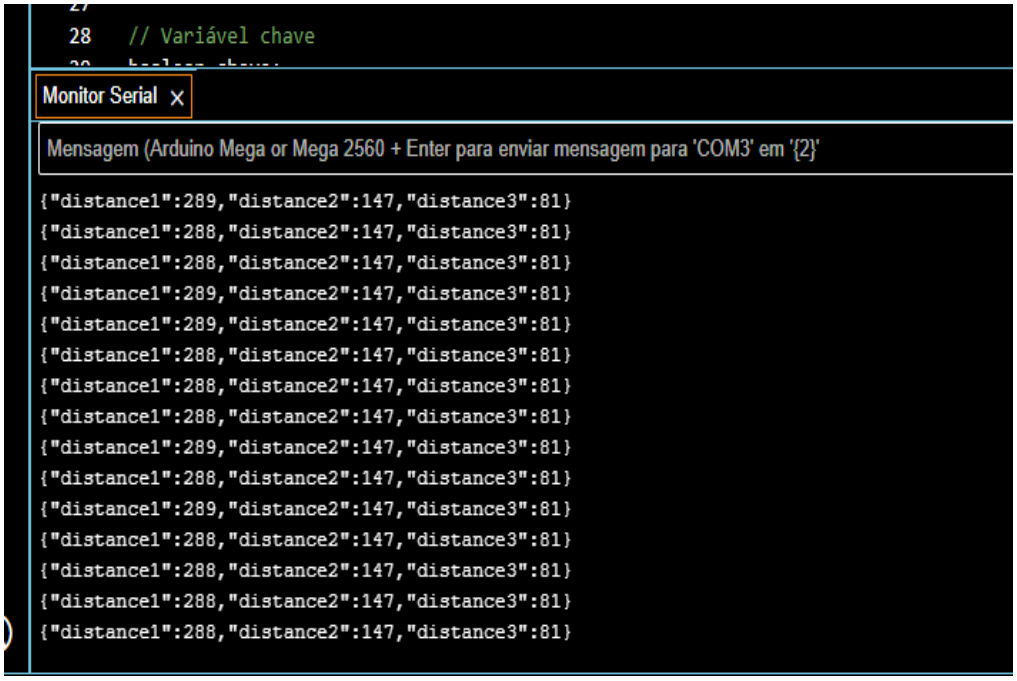
Nas Figuras 16 e 17 são ilustrados os monitores seriais do Arduino e do ESP32 durante um dos testes de captação de dados, com o intuito de verificar o bom funcionamento da transmissão de um para o outro. É possível verificar que a distância lida em tempo real pelos três sensores estão sendo recebidas pelo Arduino e enviadas para o ESP32. O trecho "*New record created successfully*" é uma mensagem que indica que o registro daquela informação foi criado no banco de dados.

Figura 16 – Monitor serial Arduino capturado no mesmo frame do ESP32.



Fonte: Elaborado pelo autor.

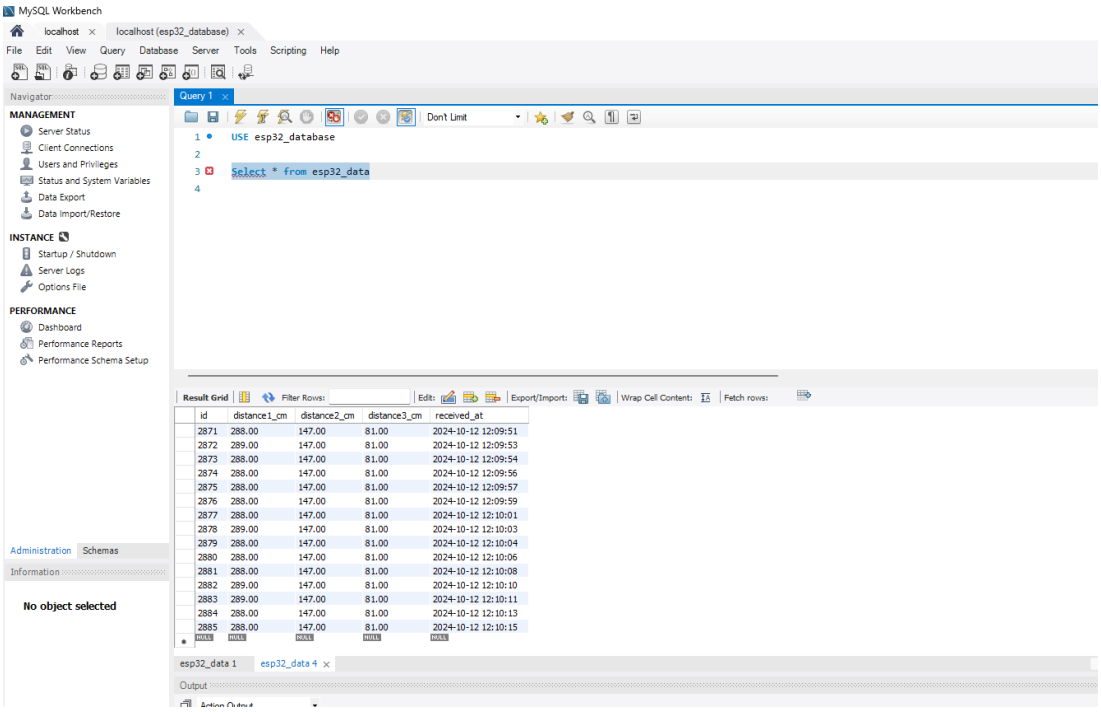
Figura 17 – Monitor serial ESP32 capturado no mesmo frame do Arduino.



Fonte: Elaborado pelo autor.

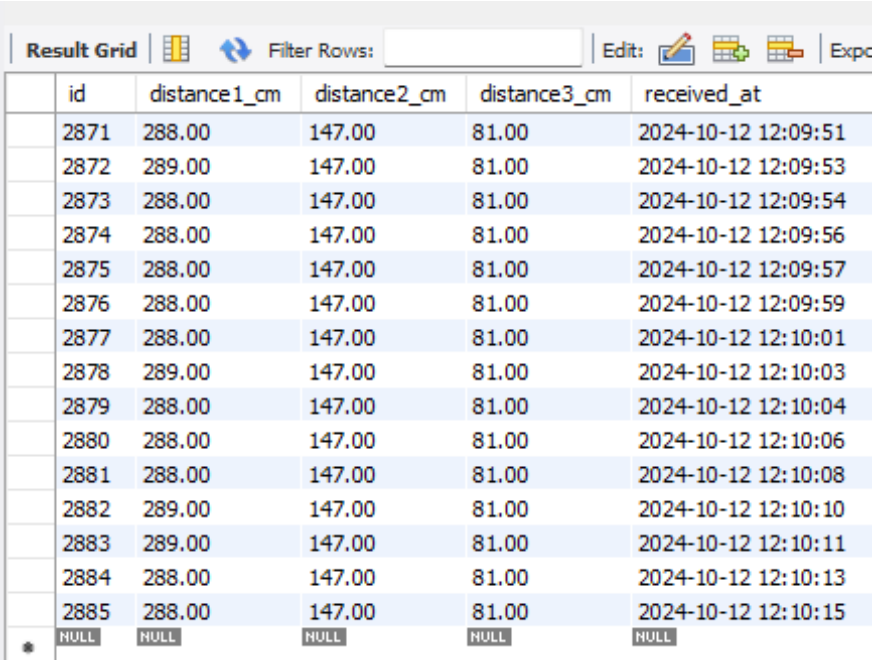
Com a comunicação entre os dispositivos estabelecida, seguindo para a próxima etapa, criou-se um banco de dados utilizando o MySQL Server e utilizou-se um computador pessoal como servidor local (*localhost*) para essa tarefa. Criou-se Um banco de dados nomeado como "esp32_database", e dentro dele, estruturou-se uma tabela chamada "esp32_data", com as colunas “id” "distance1_cm", "distance2_cm" e "distance3_cm" e “receive_at”. A coluna “id” gera automaticamente um número inteiro sequencial para cada linha de dados recebida, de forma que a combinação das três distâncias medidas seja relacionada à uma data e hora de recebimento única. A Figura 18 mostra o ambiente virtual do software MySQL com banco de dados “esp32_database” e a tabela “esp32_data” sendo acessados por um comando SQL de “USE” que realiza a tarefa de acionar o banco de dados desejado e “SELECT * FROM” que executa a tarefa de selecionar as colunas de interesse dentro de uma tabela que exista dentro do banco de dados utilizado. A Figura 19 mostra com maior ênfase a tabela criada dentro do banco de dados para armazenar os dados recebidos pelo ESP32 e acessada pelo comando SQL, já com 2885 leituras realizadas na prática com os três sensores HC-SR04, demonstrando o funcionamento adequado do projeto.

Figura 18 – MySQL server com servidor, banco de dados e tabela criada.



Fonte: Elaborado pelo autor.

Figura 19 – Registros dos testes armazenados na tabela do banco de dados.

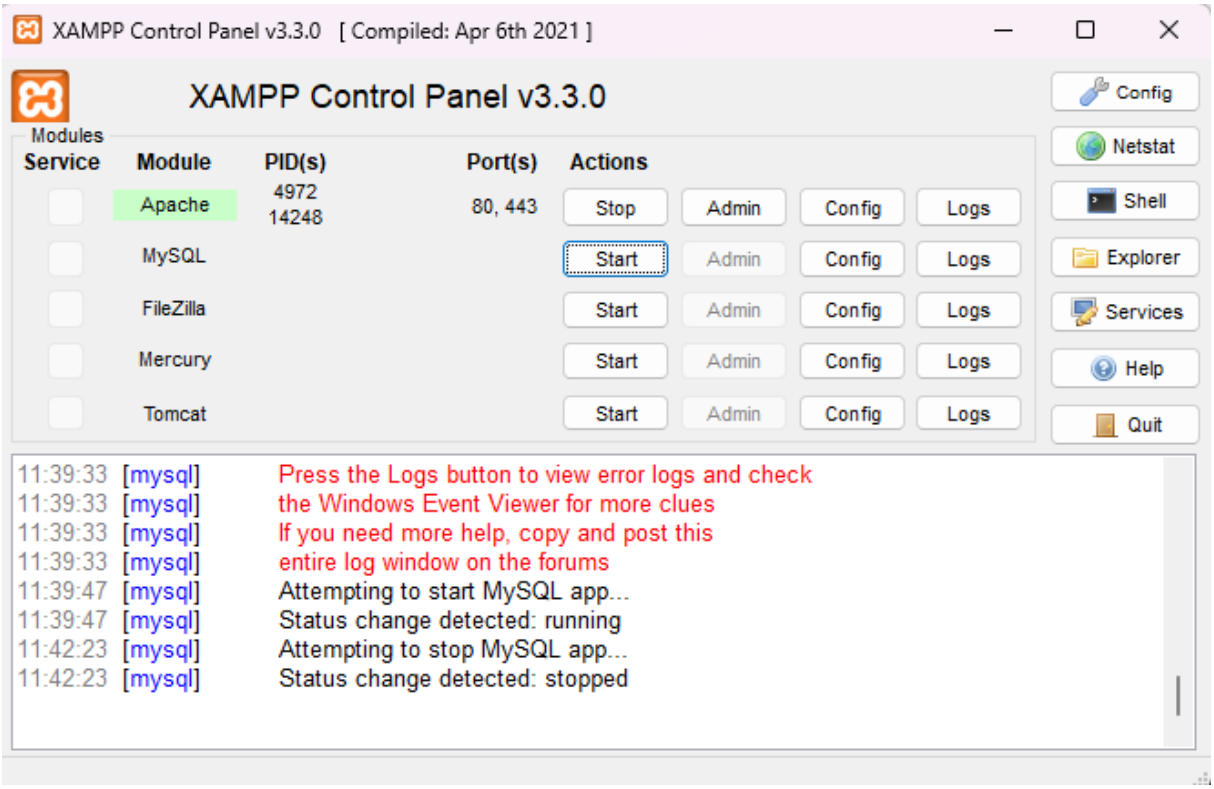


Fonte: Elaborado pelo autor.

Após a criação do banco de dados e da tabela, ajustou-se o código do ESP32 para se conectar à rede Wi-Fi e enviar os dados coletados para o banco de dados.

Nesse ponto, identificou-se a necessidade de utilizar uma programação em PHP para facilitar essa comunicação. Desenvolveu-se um script em PHP para realizar a comunicação entre o ESP32 e o banco de dados SQL. O script é um arquivo do tipo ".txt", que se encontra no Apêndice III, mas está parcialmente ilustrado pela Figura 23. O Apache no XAMPP opera como um servidor web que para desenvolvimento local é essencial para permitir comunicação entre hardware e banco de dados. O ESP32 utiliza do script PHP que é intermediado pelo Apache para enviar as informações para o banco de dados MySQL. O ambiente virtual do painel de controle do XAMPP pode ser visualizado na Figura 20.

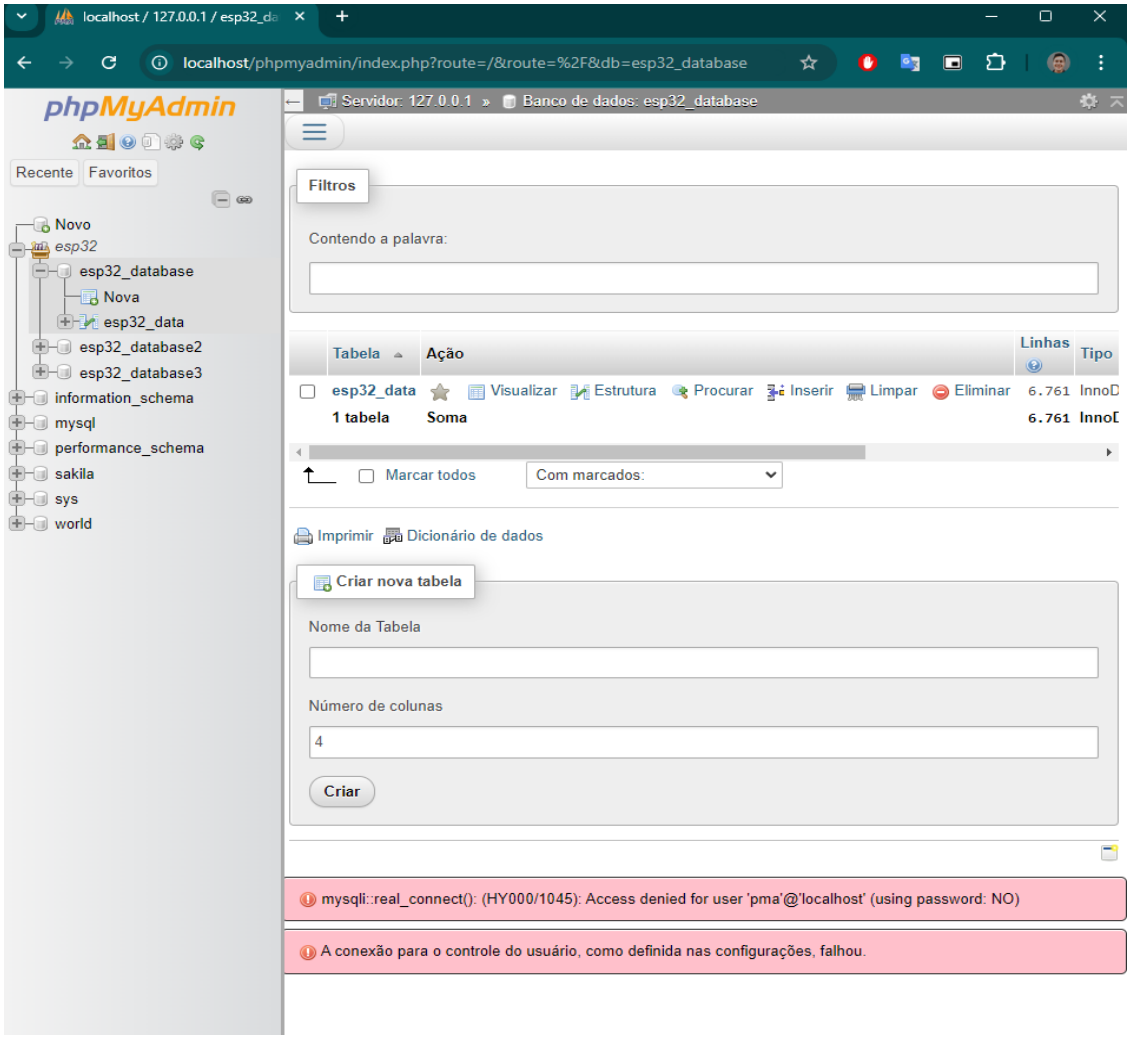
Figura 20 – Ambiente virtual do software XAMPP para execução Apache.



Fonte: Elaborado pelo autor.

O módulo Apache do XAMPP permite também acessar o banco de dados por uma página *web*, facilitando ainda mais a visualização dos resultados em tempo real. O ambiente virtual se chama PhpMyAdmin, e pode ser visualizado através das Figuras 21 e 44. Neste ambiente é possível extrair os dados das leituras armazenados na tabela "esp32_data" de forma simples para planilhas, excluir dados e até mesmo alterar de forma manual o formato da tabela utilizada.

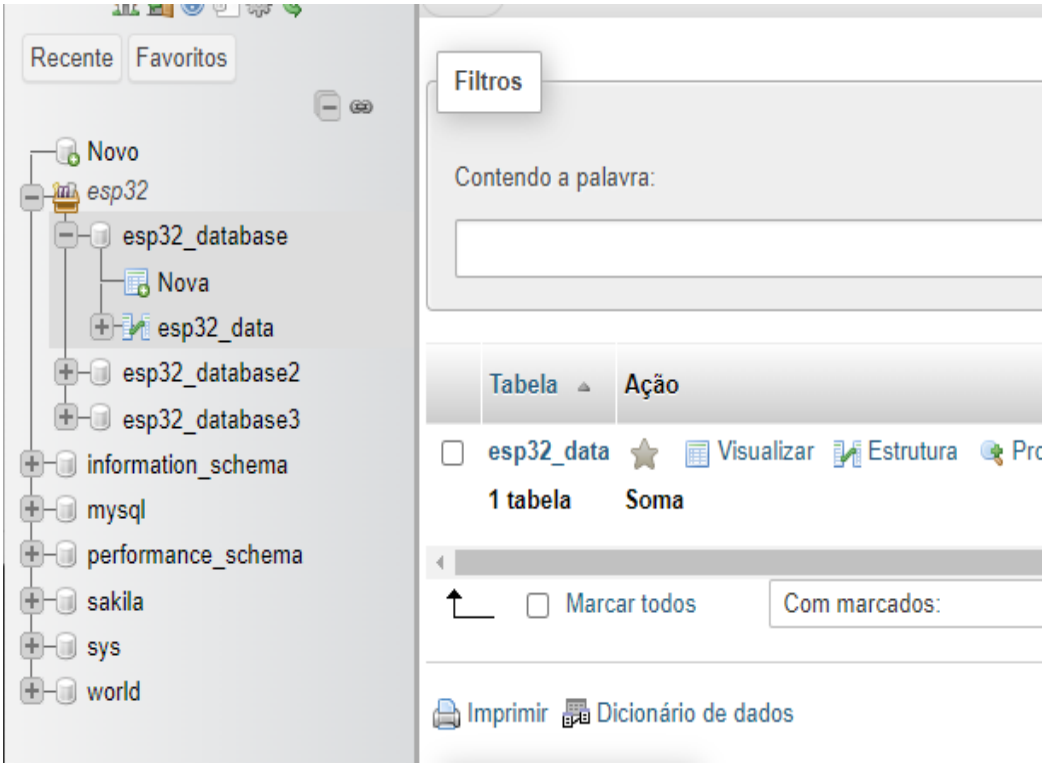
Figura 21 – PhpMyAdmin ambiente virtual



Fonte: Elaborado pelo autor.

A Figura 22 mostra com maior nível de detalhes o menu lateral para criação do banco de dados, criação e edição das tabelas do banco, além de outras opções que não foram exploradas nesse projeto.

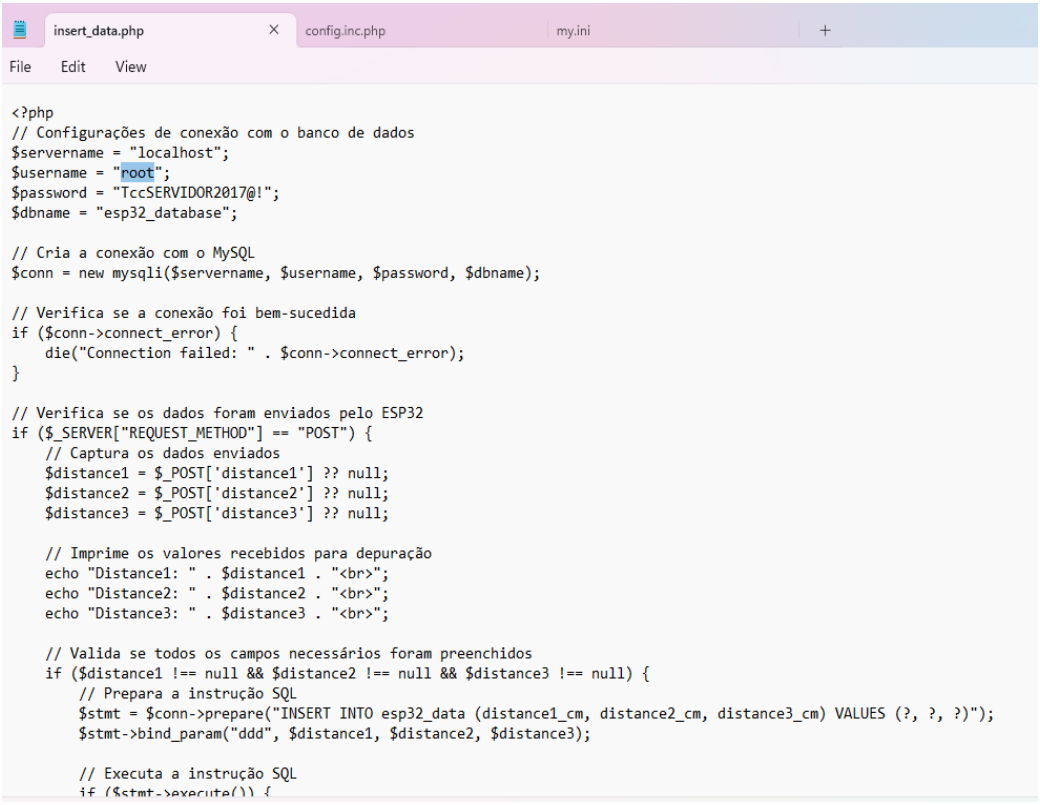
Figura 22 – PhpMyAdmin visualização da estrutura.



Fonte: Elaborado pelo autor.

O script PHP foi inserido e editado em um arquivo do tipo txt, que posteriormente é salvo no formato php, que fará a interação dos dados do ESP32 com o banco de dados SQL denominado “esp32_database”, inserindo diretamente na tabela “esp32_data”.

Figura 23 – Exemplificação do script em txt “insert_data”.



insert_data.php | config.inc.php | my.ini

File Edit View

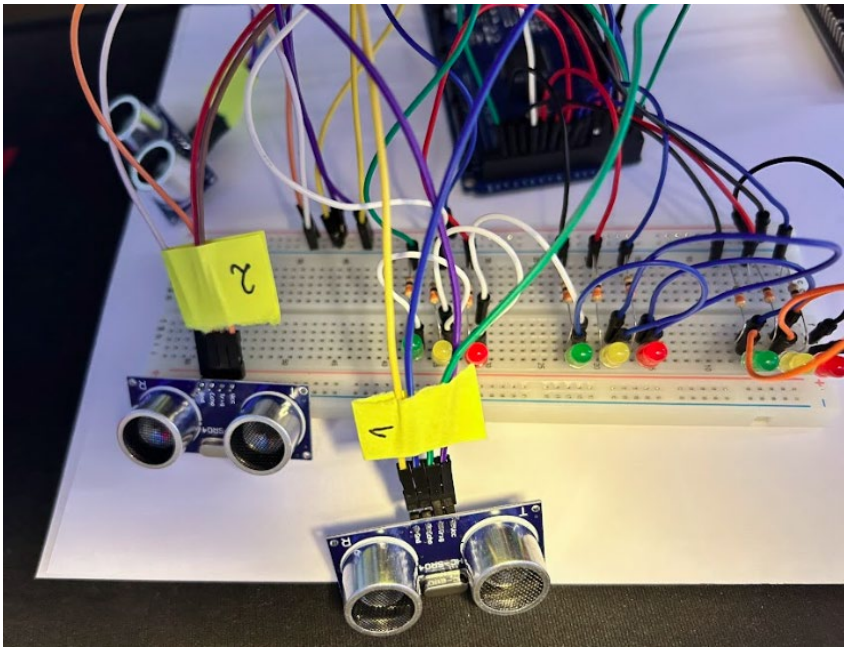
```php
<?php
// Configurações de conexão com o banco de dados
$servername = "localhost";
$username = "root";
$password = "TccSERVIDOR2017@!";
$dbname = "esp32_database";

// Cria a conexão com o MySQL
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica se a conexão foi bem-sucedida
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Verifica se os dados foram enviados pelo ESP32
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Captura os dados enviados
    $distance1 = $_POST['distance1'] ?? null;
    $distance2 = $_POST['distance2'] ?? null;
    $distance3 = $_POST['distance3'] ?? null;

    // Imprime os valores recebidos para depuração
    echo "Distance1: " . $distance1 . "<br>";
    echo "Distance2: " . $distance2 . "<br>";
    echo "Distance3: " . $distance3 . "<br>";

    // Valida se todos os campos necessários foram preenchidos
    if ($distance1 !== null && $distance2 !== null && $distance3 !== null) {
        // Prepara a instrução SQL
        $stmt = $conn->prepare("INSERT INTO esp32_data (distance1_cm, distance2_cm, distance3_cm) VALUES (?, ?, ?)");
        $stmt->bind_param("ddd", $distance1, $distance2, $distance3);

        // Executa a instrução SQL
        if ($stmt->execute()) {
```

Fonte: Elaborado pelo autor.

Realizou-se os primeiros testes práticos com apenas um sensor, que resultou na inserção dos dados na tabela criada dentro do banco de dados do servidor MySQL, conforme esperado. Com o sistema em operação, os outros dois sensores foram integrados, e novos testes foram conduzidos. Todos os testes ocorreram em um ambiente controlado, onde as medidas dos sensores, expressas em centímetros, indicou a distância do sensor até o objeto mais próximo, geralmente uma parede.

Após a primeira etapa de testes, montou-se o circuito completo do projeto, utilizando os materiais mencionados no tópico anterior, e o protótipo montado pode ser visualizado através da Figuras 24.

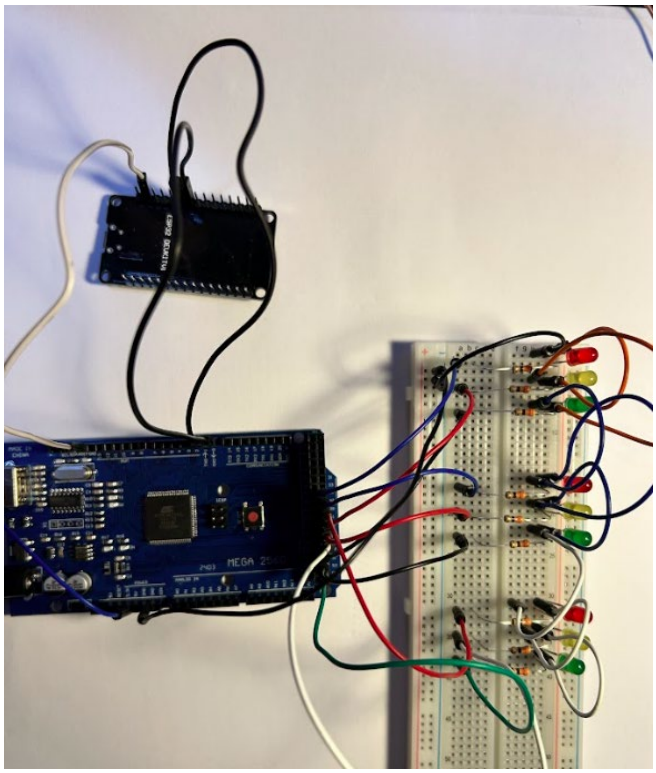
Figura 24 – Circuito utilizado para testes práticos.



Fonte: Elaborado pelo autor.

A Figura 25 ilustra em outra perspectiva o circuito utilizado para testes práticos.

Figura 25 – Circuito utilizado para testes práticos sem os sensores conectados.



Fonte: Elaborado pelo autor.

O protótipo entrega resultados visuais (ligando os LEDs) e virtuais (dados armazenados no banco de dados SQL). O critério pré-programado, conforme os códigos do Arduino e do ESP32, disponíveis nos Apêndices I e II, consiste em três diferenciações:

- Para distâncias inferiores à 100 cm, o LED vermelho é ligado.
- Para distâncias maiores que 100 cm e menores ou iguais à 200 cm, o LED amarelo é ligado.
- Para distâncias superiores à 200 cm, o LED verde é ligado.

Os LEDs podem ser considerados um sistema de redundância, para que seja sinalizado visualmente uma irregularidade ou inconformidade nas medidas. Cada um dos sensores possui três LEDs conectados em série com resistores, que jamais são ligados simultaneamente entre si, pois o gatilho de acionamento é uma faixa de operação bem definida. Ao mesmo tempo que o sinal visual é acionado, o Arduino capta essa informação e executa a sequência de tarefas descritas anteriormente.

Para os testes, realizou-se a alimentação Vcc dos componentes por meio de cabos USB das próprias placas (Arduino e ESP32), conectadas ao computador. Contudo, ressalta-se que é possível alimentar as placas de formas alternativas como fontes de corrente contínua e banco de baterias de corrente contínua.

Alguns dos resultados obtidos em relação ao armazenamento dos dados podem ser visualizados nas Figuras 26 e 27, em que os testes foram realizados colocando obstáculos em diferentes distâncias dos sensores, mostrando a resposta em tempo real, com a alteração dos valores na tabela do banco de dados para o projeto.

Figura 26 – Dados obtidos colocando obstáculos entre os sensores e a parede.

	id	distance1_cm	distance2_cm	distance3_cm	received_at
Editar Copiar Remover	6897	59.00	146.00	254.00	2024-10-16 05:43:39
Editar Copiar Remover	6898	55.00	146.00	254.00	2024-10-16 05:43:40
Editar Copiar Remover	6899	179.00	146.00	254.00	2024-10-16 05:43:42
Editar Copiar Remover	6900	179.00	146.00	254.00	2024-10-16 05:43:44
Editar Copiar Remover	6901	180.00	146.00	254.00	2024-10-16 05:43:45
Editar Copiar Remover	6902	45.00	146.00	254.00	2024-10-16 05:43:47
Editar Copiar Remover	6903	9.00	106.00	255.00	2024-10-16 05:43:49
Editar Copiar Remover	6904	4.00	110.00	253.00	2024-10-16 05:43:50
Editar Copiar Remover	6905	5.00	109.00	253.00	2024-10-16 05:43:52
Editar Copiar Remover	6906	6.00	109.00	254.00	2024-10-16 05:43:54
Editar Copiar Remover	6907	179.00	146.00	5.00	2024-10-16 05:43:55
Editar Copiar Remover	6908	180.00	146.00	5.00	2024-10-16 05:43:57
Editar Copiar Remover	6909	179.00	146.00	4.00	2024-10-16 05:43:59
Editar Copiar Remover	6910	179.00	146.00	35.00	2024-10-16 05:44:01
Editar Copiar Remover	6911	180.00	9.00	47.00	2024-10-16 05:44:03
Editar Copiar Remover	6912	179.00	8.00	37.00	2024-10-16 05:44:04
Editar Copiar Remover	6913	179.00	9.00	39.00	2024-10-16 05:44:06
Editar Copiar Remover	6914	179.00	8.00	39.00	2024-10-16 05:44:08
Editar Copiar Remover	6915	23.00	146.00	254.00	2024-10-16 05:44:09
Editar Copiar Remover	6916	180.00	146.00	253.00	2024-10-16 05:44:11

Fonte: Elaborado pelo autor.

Figura 27 – Dados obtidos tampando o trigger do sensor 1.

Editar Copiar Remover	6686	0.00	148.00	81.00	2024-10-12 14:05:48
Editar Copiar Remover	6687	287.00	147.00	81.00	2024-10-12 14:05:50
Editar Copiar Remover	6688	289.00	147.00	81.00	2024-10-12 14:05:51
Editar Copiar Remover	6689	0.00	148.00	81.00	2024-10-12 14:05:53
Editar Copiar Remover	6690	0.00	147.00	81.00	2024-10-12 14:05:55
Editar Copiar Remover	6691	288.00	147.00	81.00	2024-10-12 14:05:57
Editar Copiar Remover	6692	289.00	147.00	81.00	2024-10-12 14:05:58
Editar Copiar Remover	6693	288.00	147.00	81.00	2024-10-12 14:06:00
Editar Copiar Remover	6694	288.00	147.00	81.00	2024-10-12 14:06:01
Editar Copiar Remover	6695	289.00	147.00	81.00	2024-10-12 14:06:03
Editar Copiar Remover	6696	0.00	147.00	81.00	2024-10-12 14:06:05
Editar Copiar Remover	6697	289.00	147.00	81.00	2024-10-12 14:06:07
Editar Copiar Remover	6698	288.00	147.00	81.00	2024-10-12 14:06:08
Editar Copiar Remover	6699	288.00	147.00	81.00	2024-10-12 14:06:10
Editar Copiar Remover	6700	288.00	147.00	81.00	2024-10-12 14:06:12
Editar Copiar Remover	6701	288.00	147.00	81.00	2024-10-12 14:06:13

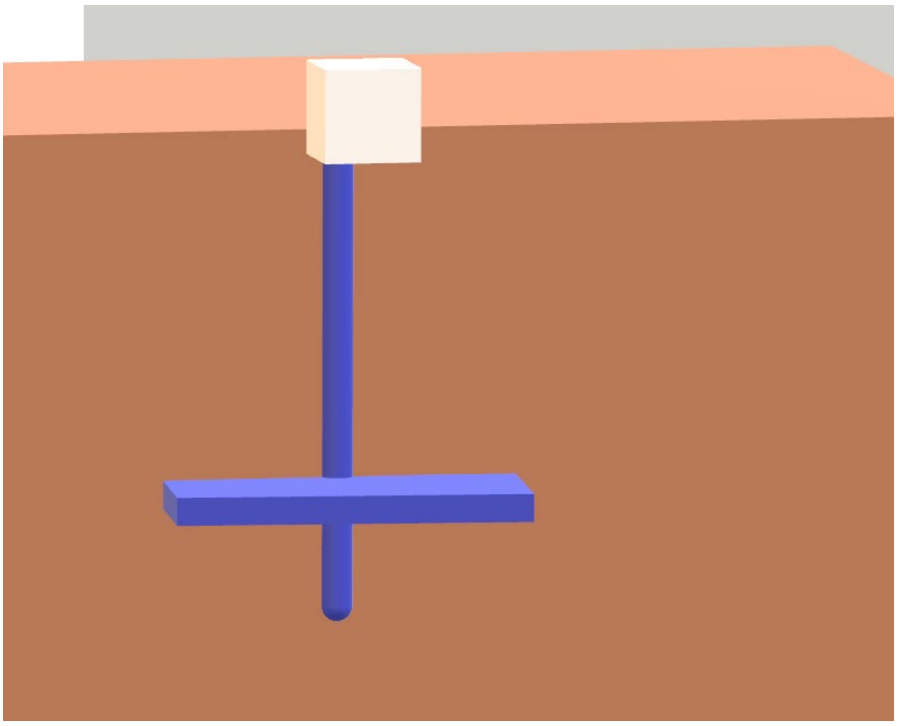
Fonte: Elaborado pelo autor.

3.3 DESAFIOS IDENTIFICADOS EM UMA IMPLEMENTAÇÃO EXPERIMENTAL REAL.

Na implementação prática do projeto de monitoramento de nível de água em uma barragem hidrelétrica, diversos desafios podem surgir, exigindo uma abordagem cuidadosa e multifacetada. Um dos principais problemas a serem enfrentados são as intempéries que podem afetar significativamente o desempenho do sensor ultrassônico HC-SR04. Condições climáticas extremas, variações de umidade e temperatura podem alterar a velocidade do som no ar, comprometendo a precisão das medições. Mudanças bruscas nas leituras, como variações superiores a 50 centímetros em um dia sem alterações climáticas significativas, podem indicar problemas no sensor ou interferências ambientais. Para mitigar esses efeitos, é possível implementar um sistema de calibração automática que considere as condições ambientais, utilizar sensores adicionais de temperatura e umidade para compensar as leituras, e instalar o sensor em um invólucro protetor resistente às intempéries.

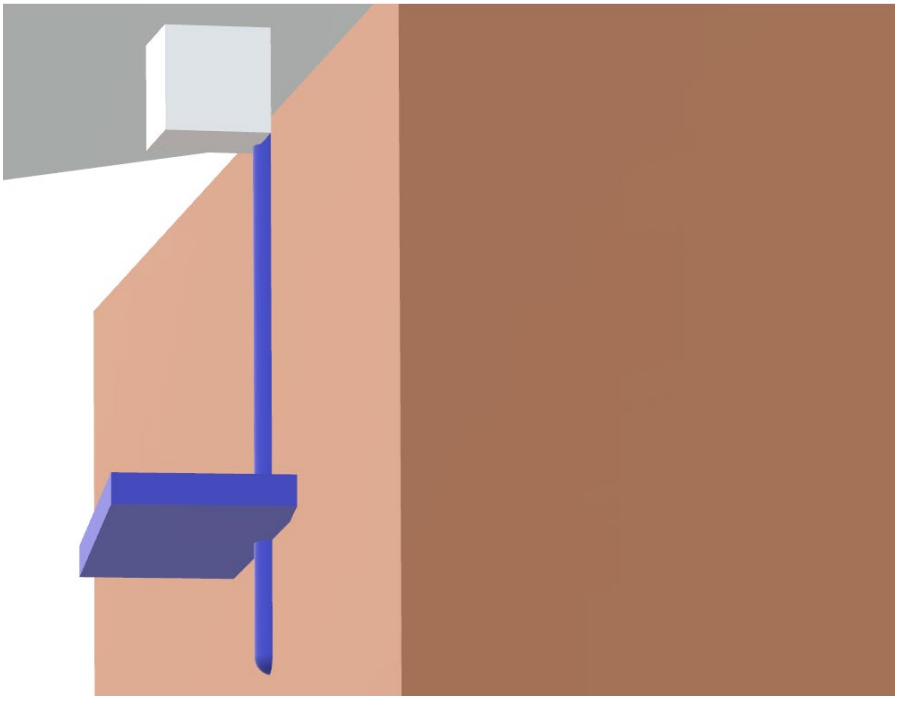
Outro desafio significativo é a dificuldade do sensor HC-SR04 em realizar leituras precisas em superfícies líquidas, uma vez que não foi projetado especificamente para esta finalidade. A solução proposta de utilizar uma estrutura mecânica com um trilho e um suporte flutuante é uma abordagem engenhosa que permite que o sensor meça a distância até uma superfície sólida (o flutuador) em vez da superfície da água diretamente. Esta configuração não só permite que o sistema se ajuste automaticamente às variações do nível de água, mas também reduz a interferência causada por ondulações na superfície. No entanto, é crucial considerar a manutenção regular dessa estrutura mecânica para garantir seu funcionamento adequado ao longo do tempo. Um exemplo dessa estrutura mecânica apenas para fins de visualização foi feito utilizando o software Paint 3D, que é padrão da Microssoft Windows e está disposto nas Figuras 28 e 29.

Figura 28 – Exemplo de estrutura mecânica para dar suporte ao sensor e confiabilidade o sistema do projeto.



Fonte: Elaborado pelo autor.

Figura 29 – Vista lateral de estrutura mecânica para dar suporte ao sensor e confiabilidade o sistema do projeto.



Fonte: Elaborado pelo autor.

A interferência por fatores biológicos, como a presença de animais, insetos ou detritos obstruindo o caminho do sensor, é outro problema potencial que pode levar a leituras incorretas. Para abordar esta questão, pode-se implementar um sistema de detecção de anomalias que identifique leituras fora do padrão esperado, instalar barreiras físicas ou repelentes para minimizar a interferência, utilizar múltiplos sensores em diferentes posições para redundância e validação cruzada das leituras, e programar rotinas de autolimpeza ou alertas para manutenção quando forem detectadas obstruções persistentes.

Além dessas medidas específicas, é fundamental estabelecer um protocolo de manutenção regular e inspeção visual do sistema para identificar e resolver problemas precocemente. A implementação de um sistema de alerta que notifique os operadores sobre leituras anômalas ou falhas no sistema também é altamente recomendada. Estas considerações demonstram a complexidade de implementar um sistema de monitoramento em um ambiente real e destacam a importância de uma abordagem holística que combine hardware robusto, software inteligente e procedimentos operacionais bem definidos. Ao abordar esses desafios de maneira abrangente, o projeto pode alcançar um nível de confiabilidade e precisão necessário para o monitoramento eficaz de barragens hidrelétricas, contribuindo significativamente para a segurança e eficiência operacional dessas estruturas críticas.

4 CONCLUSÕES

O projeto de desenvolvimento de um sistema baseado em Arduino para monitoramento de nível de água em barragens hidrelétricas usando Arduino MEGA 2560 e um ESP32 com plataforma de controle e sensores HC-SR04.

A integração de sensores ultrassônicos HC-SR04, Arduino Mega 2560, e ESP32, juntamente com um sistema de armazenamento em banco de dados SQL, resultou em uma solução robusta e confiável para o monitoramento contínuo e em tempo real dos níveis de água.

Os resultados obtidos durante os testes em ambiente controlado foram positivos, indicando a viabilidade da implementação deste sistema em cenários reais de barragens hidrelétricas. A precisão das medições, a eficiência na transmissão de dados e a confiabilidade do armazenamento demonstraram o potencial do sistema que pode contribuir significativamente com a segurança e a gestão operacional de barragens.

A utilização de tecnologias de código aberto e componentes de baixo custo pode ser uma estratégia importante para dar acessibilidade a qualquer usuário em diferentes contextos e necessidades. Além disso, a arquitetura modular do projeto permite futuras expansões e melhorias, como a incorporação de análises preditivas ou a integração com outros sistemas de monitoramento.

É importante ressaltar que, embora os resultados iniciais sejam promissores, testes adicionais em condições reais de operação são necessários para validar completamente a robustez e a longevidade do sistema. Aspectos como a resistência a intempéries, a confiabilidade em longo prazo e a integração com sistemas existentes de segurança de barragens devem ser considerados em futuros desenvolvimentos.

Conclui-se, portanto, que este projeto não apenas atendeu às expectativas iniciais, mas também abriu caminhos para futuras pesquisas e aplicações na área de monitoramento ambiental e segurança de infraestruturas hídricas. A combinação de tecnologias IoT com práticas tradicionais de engenharia demonstra o potencial de inovação no campo da gestão de recursos hídricos, contribuindo para a segurança e eficiência operacional de barragens hidrelétricas.

5 REFERÊNCIAS BIBLIOGRÁFICAS

Agência Nacional das Águas; Ministério do Meio Ambiente. Classificação de Barragens: Melhores Práticas Nacionais e Internacionais *[Documento Técnico]*. Brasília, DF: ANA; 2013.

AMAZON. *HiLetgo ESP-WROOM-32 Development Board*. Disponível em: www.amazon.com. Acesso em: 5 out. 2024.

BLUM, Jeremy. *Exploring Arduino: Tools and Techniques for Engineering Wizardry*. 1st ed. Indianapolis: Wiley, 2013.

Brasil. Lei nº 12.234, de 20 de setembro de 2010. Estabelece a Política Nacional de Segurança de Barragens, Diário Oficial da União. 21 set. 2010.

BRUNO, O. M.; ESTROZI, L. F.; NETO, J. E. S. B. Programando para a internet com PHP. 1. ed. Rio de Janeiro, RJ: Brasport, 2008. *E-book*. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 3 out. 2024.

CYTRON TECHNOLOGIES SDN. BHD. – ALL RIGHTS RESERVED. Product User's Manual – HCSR04 Ultrasonic Sensor. [s.l: s.n.]. Disponível em: https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsBx2qR4vP8saG73rE/edit Acesso em: 3 out. 2024

DEITEL, H. M.; DEITEL, P. J.C++:Como programar. 5. ed. São Paulo: Pearson, 2006.E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 8 out. 2024.

GTMax3D. Arduino Mega 2560 R3. Disponível em: www.gtmax3d.com.br. Acesso em: 5 out. 2024.

JACCON, G.; CUDO, K.J. Curva-chave: Análise e traçado. Brasília: DNAEE. p. 52-59. 1989.

McROBERTS, M. Arduino Básico. Rua Luís Antônio dos Santos 110 02460-000 – São Paulo, SP – Brasil: Novatec Editora Ltda., set. 2011.

NodeMCU ESP8266. Disponível em: https://www.amazon.com/DevelopmentWireless-Micropython-Programming-Soldered/dp/B0BVM3H46B . Acesso em: 3 out. 2024

SILVA, Eliane Lima; SILVA, Mariano Andrade da. Segurança de barragens e os riscos potenciais à saúde pública. Saúde em Debate, v. 44, 2021.

UPESY. ESP32 *Pinout Reference: Which GPIO pins should you use?* Disponível em: www.upesy.com. Acesso em: 5 out. 2024.

UNITED STATES BUREAU OF RECLAMATION - USBR. *Dams and Public Safety*. *A Water Resources Technical Publication*. *Denver*, 1983.

USACE *Best Practices in Dam and Levee Safety Risk Analysis. Corps of Engineers. Waterways Experiment Station*. 2014.

USINAINF. Arduino Mega 2560 R3 + Cabo USB. Disponível em: www.usinainfo.com.br. Acesso em: 5 out. 2024.

WARREN, John-David; ADAMS, Josh; MOLLE, *Harald*. Arduino para robótica. 1. ed. São Paulo: *Blucher*, 2019. 579 p.

APÊNDICE I – CÓDIGO ARDUINO

CÓDIGO DO ARDUINO

```
#include <NewPing.h>
#include <ArduinoJson.h>

// Definindo pinagem dos sensores
#define TRIGGER_PIN1 23
#define ECHO_PIN1 29
#define TRIGGER_PIN2 25
#define ECHO_PIN2 31
#define TRIGGER_PIN3 27
#define ECHO_PIN3 22

// Definindo pinagem dos LEDs
#define VERMELHO_LED_1_PIN 35
#define VERMELHO_LED_2_PIN 37
#define VERMELHO_LED_3_PIN 39
#define AMARELO_LED_1_PIN 41
#define AMARELO_LED_2_PIN 43
#define AMARELO_LED_3_PIN 45
#define VERDE_LED_1_PIN 47
#define VERDE_LED_2_PIN 49
#define VERDE_LED_3_PIN 51
#define switch_PIN 4

NewPing sonar1(TRIGGER_PIN1, ECHO_PIN1, 4000);
NewPing sonar2(TRIGGER_PIN2, ECHO_PIN2, 4000);
NewPing sonar3(TRIGGER_PIN3, ECHO_PIN3, 4000);


void setup() {
  Serial.begin(9600);
  pinMode(VERMELHO_LED_1_PIN, OUTPUT);
  pinMode(VERMELHO_LED_2_PIN, OUTPUT);
  pinMode(VERMELHO_LED_3_PIN, OUTPUT);
  pinMode(AMARELO_LED_1_PIN, OUTPUT);
  pinMode(AMARELO_LED_2_PIN, OUTPUT);
  pinMode(AMARELO_LED_3_PIN, OUTPUT);
  pinMode(VERDE_LED_1_PIN, OUTPUT);
  pinMode(VERDE_LED_2_PIN, OUTPUT);
  pinMode(VERDE_LED_3_PIN, OUTPUT);
}

void loop() {

  // Leitura da distância dos sensores
  unsigned int distance1 = sonar1.ping_cm();
  unsigned int distance2 = sonar2.ping_cm();
  unsigned int distance3 = sonar3.ping_cm();

  // Controle dos LEDs com base na distância
  // LED Vermelho 1
  if (distance1 <= 100) {
    digitalWrite(VERMELHO_LED_1_PIN, HIGH);
  } else {
    digitalWrite(VERMELHO_LED_1_PIN, LOW);
  }

  // LED Vermelho 2
  if (distance2 <= 100) {
    digitalWrite(VERMELHO_LED_2_PIN, HIGH);
  } else {
    digitalWrite(VERMELHO_LED_2_PIN, LOW);
  }

  // LED Vermelho 3
```

```
  if (distance3 <= 100) {
    digitalWrite(VERMELHO_LED_3_PIN, HIGH);
  } else {
    digitalWrite(VERMELHO_LED_3_PIN, LOW);
  }

  // LED Amarelo 1
  if (distance1 > 100 && distance1 <= 200) {
    digitalWrite(AMARELO_LED_1_PIN, HIGH);
  } else {
    digitalWrite(AMARELO_LED_1_PIN, LOW);
  }

  // LED Amarelo 2
  if (distance2 > 100 && distance2 <= 200) {
    digitalWrite(AMARELO_LED_2_PIN, HIGH);
  } else {
    digitalWrite(AMARELO_LED_2_PIN, LOW);
  }

  // LED Amarelo 3
  if (distance3 > 100 && distance3 <= 200) {
    digitalWrite(AMARELO_LED_3_PIN, HIGH);
  } else {
    digitalWrite(AMARELO_LED_3_PIN, LOW);
  }

  // LED Verde 1
  if (distance1 > 200) {
    digitalWrite(VERDE_LED_1_PIN, HIGH);
  } else {
    digitalWrite(VERDE_LED_1_PIN, LOW);
  }

  // LED Verde 2
  if (distance2 > 200) {
    digitalWrite(VERDE_LED_2_PIN, HIGH);
  } else {
    digitalWrite(VERDE_LED_2_PIN, LOW);
  }

  // LED Verde 3
  if (distance3 > 200) {
    digitalWrite(VERDE_LED_3_PIN, HIGH);
  } else {
    digitalWrite(VERDE_LED_3_PIN, LOW);
  }

  // Cria um objeto JSON para enviar os dados
  StaticJsonDocument<200> doc;
  doc["distance1"] = distance1;
  doc["distance2"] = distance2;
  doc["distance3"] = distance3;


  // Serializa o JSON e envia para o ESP32
  String jsonString;
  serializeJson(doc, jsonString);
  Serial.println(jsonString);

  delay(1500); // Atraso entre leituras
}
```

APÊNDICE II - CÓDIGO DO ESP-32

CÓDIGO DO ESP-32

```
#define RXp2 16
#define TXp2 17
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>

// Defina o nome da rede Wi-Fi (SSID) e a senha (PASSWORD)
const char* ssid = "APTO_704";
const char* password = "menininhos1000@!";

// URL do script PHP
const char* serverName = "http://192.168.0.6/insert_data.php";  // Substitua pelo endereço IP do seu computador

// Variável booleana para verificar a conexão Wi-Fi
bool wifiConnected = false;

// Capacidade do documento JSON
const int capacity = JSON_OBJECT_SIZE(10) + 100; // Ajuste a capacidade conforme necessário

// Função para conectar ao Wi-Fi
void connectToWiFi() {
  Serial.print("Tentando conectar ao Wi-Fi...");
  WiFi.begin(ssid, password);
  int retryCount = 0;
  while (WiFi.status() != WL_CONNECTED && retryCount < 10) {
    delay(500);
    Serial.print(".");
    retryCount++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    wifiConnected = true;
    Serial.println("");
    Serial.println("WiFi conectado.");
    Serial.println("Endereço IP: ");
    Serial.println(WiFi.localIP());
  } else {
    wifiConnected = false;
    Serial.println("");
    Serial.println("Falha na conexão Wi-Fi.");
  }
}

// Função para enviar dados via HTTP POST
void sendData(float distance1, float distance2, float distance3) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;

    http.begin(serverName);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    // Formata os dados como uma string de consulta
    String httpRequestData = "distance1=" + String(distance1) + "&distance2=" + String(distance2) + "&distance3=" +
String(distance3);

    int httpResponseCode = http.POST(httpRequestData);

    if (httpResponseCode > 0) {
      String response = http.getString();
      Serial.println(httpResponseCode);
      Serial.println(response);
    } else {
      Serial.print("Error on sending POST: ");
      Serial.println(httpResponseCode);
    }
```

```
    http.end();
  } else {
    Serial.println("Falha em conectar ao WiFi");
  }
}

void setup() {
  // Inicializa a comunicação serial para monitoramento
  Serial.begin(115200);

  // Inicializa a comunicação serial com o Arduino
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);

  // Tenta conectar ao Wi-Fi
  connectToWiFi();
}

void loop() {
  // Verifica a conexão Wi-Fi
  if (WiFi.status() == WL_CONNECTED) {
    wifiConnected = true;
    Serial.println("O sistema está conectado ao banco de dados");
  } else {
    wifiConnected = false;
    Serial.println("O sistema não está conectado ao banco de dados, verificar conexão");
  }

  // Coleta dados do Arduino
  if (Serial2.available()) {
    String data = Serial2.readStringUntil('\n'); // Lê até o final da linha
    data.trim(); // Remove espaços em branco e caracteres de nova linha
    Serial.println("Dados recebidos do arduino: ");
    Serial.println(data);

    // Verifica se os dados recebidos são válidos
    if (data.length() > 0 && data[0] == '{' && data[data.length() - 1] == '}') {
      // Deserializa o JSON recebido
      StaticJsonDocument<capacity> doc;
      DeserializationError error = deserializeJson(doc, data.c_str());

      if (error) {
        Serial.print("Erro ao deserializar JSON: ");
        Serial.println(error.c_str());
        return;
      }

      // Extrai os valores do JSON
      float distance1 = doc["distance1"];
      float distance2 = doc["distance2"];
      float distance3 = doc["distance3"];

      // Envia os dados para o servidor
      sendData(distance1, distance2, distance3);
    } else {
      Serial.println("Dados recebidos inválidos");
    }
  }

  // Se não estiver conectado, tenta reconectar a cada 30 segundos
  if (!wifiConnected) {
    delay(30000); // Aguarda 30 segundos antes de tentar reconectar
    connectToWiFi();
  } else {
    // Aguarda um pouco antes de repetir o loop
    delay(1500);
  }
}
```

APÊNDICE III – CÓDIGO DO SCRIPT PHP

CÓDIGO PHP USADO PELO ESP32

```
<?php
// Configurações de conexão com o banco de dados
$servername = "localhost";
$username = "root";
$password = "TccSERVIDOR2017@!";
$dbname = "esp32_database";

// Cria a conexão com o MySQL
$conn = new mysqli($servername, $username, $password, $dbname);

// Verifica se a conexão foi bem-sucedida
if ($conn->connect_error) {
    die("Falha na conexão: " . $conn->connect_error);
}

// Verifica se os dados foram enviados pelo ESP32
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Captura os dados enviados
    $distance1 = $_POST['distance1'] ?? null;
    $distance2 = $_POST['distance2'] ?? null;
    $distance3 = $_POST['distance3'] ?? null;

    // Imprime os valores recebidos para depuração
    echo "Distance1: " . $distance1 . "<br>";
    echo "Distance2: " . $distance2 . "<br>";
    echo "Distance3: " . $distance3 . "<br>";

    // Valida se todos os campos necessários foram preenchidos
    if ($distance1 !== null && $distance2 !== null && $distance3 !== null) {
        // Prepara a instrução SQL
        $stmt = $conn->prepare("INSERT INTO esp32_data (distance1_cm, distance2_cm, distance3_cm) VALUES (?, ?, ?)");
        $stmt->bind_param("ddd", $distance1, $distance2, $distance3);

        // Executa a instrução SQL
        if ($stmt->execute()) {
            echo "Gravação realizada";
        } else {
            echo "Error: " . $stmt->error;
        }

        // Fecha a instrução
        $stmt->close();
    } else {
        echo "Todos os campos são obrigatórios.";
    }
}

// Fecha a conexão
$conn->close();
?>
```

APÊNDICE IV – CÓDIGO DE BIBLIOTECA *ARDUINOJSON*

ArduinoJson.h

```
#pragma once

#ifdef __cplusplus

#  include "ArduinoJson.hpp"

using namespace ArduinoJson;

#else

#error ArduinoJson requires a C++ compiler, please change file extension to .cc or .cpp

#endif
```