**UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"**
**FACULDADE DE ENGENHARIA**
**CAMPUS DE ILHA SOLTEIRA**

**BEATRIZ PEREIRA**

# MONITORAMENTO REMOTO DE NÍVEL DE OXIGÊNIO E TEMPERATURA DA ÁGUA NA CRIAÇÃO DE TILÁPIA

Ilha Solteira
2023

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

**BEATRIZ PEREIRA**

# MONITORAMENTO REMOTO DE NÍVEL DE OXIGÊNIO E TEMPERATURA DA ÁGUA NA CRIAÇÃO DE TILÁPIA

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – UNESP como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Prof. Dr. Carlos Antônio Alves
**Orientador**

Ilha Solteira
2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

P436m

Pereira, Beatriz.
Monitoramento remoto de nível de oxigênio e temperatura da água na criação de tilápia / Beatriz Pereira. -- Ilha Solteira: [s.n.], 2023
55 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Carlos Antônio Alves

Inclui bibliografia

1. Internet das coisas. 2. ESP32. 3. Protocolo MQTT.

Amanda Sertori dos Santos
Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

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

# ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte e sete dias do mês de novembro do ano de dois mil e vinte e três, a discente ***Beatriz Pereira,*** matriculada sob o nº 161052797, tendo como banca examinadora o seu orientador, *o Prof. Dr. Carlos Antonio Alves, o Prof. Dr. Alexandre Cesar Rodrigues da Silva e o Doutorando Lucas do Carmo Yamaguti,* apresentou o Trabalho de Graduação intitulado **"MONITORAMENTO REMOTO DE NÍVEL DE OXIGÊNIO E TEMPERATURA DA ÁGUA NA CRIAÇÃO DE TILÁPIA",** obtendo a nota 9.0 ( NOVE ) e conceito APROVADO.

Prof. Dr. Carlos Antonio Alves
- Orientador-

Beatriz Pereira
- Discente -

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

Doutorando Lucas do Carmo Yamaguti
- 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

## AGRADECIMENTOS

Agradeço a minha família por ser o alicerce sobre o qual construí minha conquista acadêmica. Agradeço ao Prof. Dr. Carlos Antônio Alves por ter exercido a função de orientador com empenho e consideração. Agradeço ao meu supervisor Lucas Machado de Oliveira pela oportunidade de aprendizado e auxílio com os recursos necessários para desenvolver o presente trabalho.

## RESUMO

Neste trabalho apresenta-se um protótipo de sistema de monitoramento remoto da temperatura e oxigênio dissolvido da água como um exemplo de aplicação da Internet das Coisas na piscicultura. Manter a qualidade da água com temperatura e oxigênio dissolvido em níveis adequados para o cultivo dos peixes auxilia na prevenção de doenças, diminuição da mortalidade e aumento da produtividade. A fim de monitorar remotamente os parâmetros de qualidade da água através do protótipo, foi utilizado o módulo ESP32 para obter a leitura dos sensores e enviar os valores via Wi-Fi através do protocolo de comunicação MQTT para um banco de dados InfluxDB. Os dados recebidos são acessados pela plataforma Grafana para acompanhar a variação dos parâmetros por meio de análise gráfica. O protótipo foi submetido a um teste em ambiente fechado sem sistema de climatização e com acesso à Internet via WiFi que simulou a variação dos parâmetros ocorrida em tanques de produção e obteve resultados satisfatórios ao exibir na interface gráfica a variação dos parâmetros realizada em cada intervalo do teste. A partir dos resultados obtidos no teste, foi possível inferir que o protótipo pode ser futuramente aplicado no monitoramento de qualidade da água dos tanques de produção das pisciculturas. Com a implementação do sistema, as pisciculturas terão acompanhamento frequente da variação dos parâmetros com o intuito de auxiliar nas tomadas de decisões a fim de evitar prejuízos maiores ao identificar e agir rapidamente em condições adversas, além de registrar o histórico das leituras para analisar o desempenho dos tanques conforme a fase climática. O custo inicial e a estabilidade da conexão com a Internet necessária podem dificultar a implementação do sistema em todos os tanques de produção, recomenda-se escolher pontos de leitura estratégicos com acesso à Internet e que envolvam um conjunto de tanques em condições semelhantes.

.



**Palavras-chave:** Internet das Coisas. ESP32. Protocolo MQTT.

## ABSTRACT

This work presents a prototype of a remote monitoring system for water temperature and dissolved oxygen like an example of application of Internet of Things in fish farming. Ensuring water quality at adequate levels for fish cultivation helps prevent disease, reduce mortality and increase productivity. In order to remotely monitor the water quality parameters, the ESP32 module was used to read sensors and record values in an influxDB database via Wi-Fi through MQTT communication protocol. The data recorded by the sensors can be accessed by the Grafana platform to graphically monitor the variation of parameters. The prototype was subjected to a test in a closed environment with Internet access via WiFi and without an air conditioning system. The test simulated the variation of parameters that occurred in production tanks and obtained satisfactory results by displaying in the graphical interface the variation of parameters done at each interval of the test. From the results obtained, it was possible to infer that the prototype can be applied in the future to monitor water quality in fish farming production tanks. Through the implementation of the system, fish farms will frequently monitor the variation of parameters in order to help in decision making and consequently avoid greater losses by identifying and acting quickly in adverse conditions, moreover recording the history of readings to analyze performance of tanks according to the climatic phase. The initial cost and required stability of the Internet connection may hinder to implement the system in all production tanks. It is recommended to choose strategic reading points with Internet access and involving a group of tanks in similar conditions.



**Keywords:** Internet of Things. ESP32. MQTT protocol.

## LISTA DE FIGURAS

## LISTA DE QUADROS E TABELAS

## LISTA DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| AES | Advanced Encryption Standard |
| CAN | Controller Area Network |
| FPGA | Field Programmable Gate Array |
| GPIO | General Purpose Input/Output |
| IoT | Internet of Things |
| UIT | União Internacional de Telecomunicações |
| MQTT | Message Queuing Telemetry Transport |
| PWM | Pulse Width Modulation |
| RFID | Radio Frequency Identification |
| RNG | Random Number Generator |
| RSA | Rivest-Shamir-Adleman |
| SHA | Secure Hash Algorithm |
| SSL | Secure Sockets Layer |
| TCP | Transmission Control Protocol |
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| TLS | Transport Layer Security |
| USB | Universal Serial Bus |

**SUMÁRIO**

# 1 INTRODUÇÃO

Em 1999, o pesquisador Kevin Ashton do Instituto de Tecnologia de Massachusetts tornou-se um dos pioneiros a abordar o conceito de Internet das Coisas ao apresentar a ideia de empregar o sistema de identificação *Radio Frequency Identification* (RFID) na cadeia de suprimentos da empresa Procter & Gamble. Segundo Kevin Ashton, a tecnologia da informação da época dependia das pessoas para a obtenção dos dados. A Internet das Coisas (na língua inglesa, *Internet of Things* – IoT) representou um avanço tecnológico que permitiu a observação, a identificação e o entendimento dos dados do ambiente físico por parte das máquinas em ambiente virtual sem depender exclusivamente do lançamento por ação humana (ASHTON, 2009).

A Internet das Coisas permite que os elementos do ambiente físico se conectem à Internet através de sensores que monitoram as condições e coletam os dados utilizando conexões de Internet com e/ou sem fio. Devido ao controle e automação das unidades físicas, à comunicação das informações geradas e à redução dos custos, os sistemas de IoT estão inovando diversas áreas do mercado, entre elas a indústria, a saúde e o agronegócio (LOPEZ RESEARCH LLC, 2013).

Um exemplo de aplicação da IoT na indústria é o sensor que monitora a vibração dos equipamentos com o objetivo de identificar anomalias e desgastes dos ativos e enviar alertas para programar a manutenção adequada antes que ocorra a falha que gere uma parada total de máquina e impacte negativamente na produtividade da empresa (SANTOS, 2020).

Para exemplificar uma aplicação da IoT na área da saúde, pode-se citar os sensores de glicemia utilizados para monitorar o nível de açúcar no sangue dos pacientes através de um leitor colado na pele que coleta os dados para serem disponibilizados em tempo real no aplicativo (TOLEDO, 2019).

Um dos segmentos do agronegócio em ascensão é a piscicultura com a criação de peixes. Um exemplo aplicado da IoT no arraçoamento diário dos peixes são os sistemas de alimentação inteligentes que rastreiam o comportamento do animal durante o trato com mecanismos de feedback para que os peixes se alimentem até estarem saciados sem ocorrer desperdício de alimento (TAVARES, 2021)

Nos últimos anos, a piscicultura é o empreendimento que apresenta maior crescimento no mercado de produção animal no Brasil. Em 2021, o país alcançou o

marco de 840.005 toneladas de peixes de cultivo. Ao comparar com a produção de 802.930 toneladas em 2020, a produção de peixes no Brasil em 2021 obteve um aumento de 4,7%. A tilápia é a espécie responsável pela maior parte da produção brasileira de 2021 ao representar 63,5% do cultivo de peixes com a produção de 534.005 toneladas. O resultado apresenta um crescimento de 9,8% em relação ao total de 486.155 toneladas de tilápias produzidas em 2020 (PEIXE BR, 2022).

Garantir a boa qualidade da água na piscicultura possibilita melhor desempenho e qualidade dos peixes de cultivo, previne a proliferação de doenças, permite o aumento da produtividade e impacta positivamente nos lucros. A qualidade da água pode ser avaliada através da ação conjunta de parâmetros físicos, biológicos e químicos no ambiente aquático. A temperatura e o oxigênio dissolvido fazem parte dos principais fatores que influenciam na qualidade da água (AYROZA et al., 2011).

Por serem animais pecilotérmicos, a temperatura dos peixes pode mudar conforme a variação da temperatura do ambiente aquático. O conforto térmico para a tilápia e os peixes tropicais de água doce está entre 26ºC e 30ºC. As temperaturas fora da faixa de conforto térmico acarretam na diminuição do apetite e o desenvolvimento de doenças (AYROZA et al., 2011).

O oxigênio dissolvido quantifica o oxigênio disponível na água sob condições específicas de temperatura e pressão. O ambiente aquático apresenta maior disponibilidade de oxigênio dissolvido ao longo do dia devido à fotossíntese realizada pelas algas. No decorrer da noite a quantidade de oxigênio dissolvido na água diminui devido ao consumo de oxigênio pelos processos oxidativos. O aumento da quantidade de algas pode promover maior fertilização do ambiente aquático, interferir nas oscilações diárias do oxigênio dissolvido e, consequentemente, aumentar o estresse dos animais devido à escassez do gás no período noturno. Manter o oxigênio dissolvido em níveis adequados contribui para um melhor desempenho dos peixes ao evitar desperdícios de alimento e aumentar a resistência contra toxinas e doenças (AYROZA et al., 2011).

Diante da importância do acompanhamento dos parâmetros da qualidade da água doce para garantir o melhor desenvolvimento dos peixes de cultivo, o monitoramento remoto da temperatura e oxigênio dissolvido torna-se necessário para as pisciculturas que almejam maximizar o lucro e minimizar os custos ao diminuir prejuízos com a mortalidade dos peixes, garantir a qualidade do pescado para

comercialização e evitar o agravamento do impacto ambiental devido à atividade exercida (AYROZA et al., 2011).

Neste trabalho apresenta-se um protótipo que pode servir de solução para acompanhar os parâmetros de qualidade da água em tanques-rede na criação de tilápia a fim de alertar as variações que podem colocar em risco a vida do animal. Um dos dispositivos mais utilizados para projetos de monitoramento remoto escolhido para ser empregado neste trabalho é o módulo ESP32 por ser um microcontrolador dual *core* de baixo custo com memória *flash*, *Bluetooth* 4.2 e *Wi-Fi* integrados. Com uma arquitetura de programação independente de outras placas, o dispositivo apresenta baixo consumo de energia, desempenho elevado em potência, multifuncionalidade e alta credibilidade. As plataformas de prototipagem baseadas no ESP32 incluem um conversor USB (*Universal Serial Bus*) e uma porta micro USB para energização na faixa de 2,2 volts a 3,6 volts e permitem que a programação do módulo seja compatível com o ambiente de desenvolvimento integrado do Arduino para utilizar módulos e sensores compatíveis com o Arduino (OLIVEIRA, 2017).

## 1.1 Objetivos

O objetivo geral deste trabalho de graduação é desenvolver um protótipo de sistema de monitoramento remoto do nível de temperatura e oxigênio dissolvido da água para estudar a possibilidade de uma futura aplicação em tanques-rede de criação de tilápia.

Os objetivos específicos para o desenvolvimento do protótipo incluem a pesquisa dos equipamentos adequados, a montagem do circuito experimental, a instalação e configuração dos *softwares* e serviços utilizados e a construção de uma interface visual para acompanhamento das medições.

# 2 FUNDAMENTOS TEÓRICOS

Foi levantado um estudo sobre o conceito e aplicação da IoT, microcontroladores, sensores e *softwares* para a construção do protótipo.

## 2.1 Internet das Coisas - IoT

O conceito de IoT pode ser associado à ação conjunta de tecnologias e protocolos sobre objetos em ambiente físico através de redes de comunicação para monitorar as condições físicas envolvidas. A interação entre computadores, sensores e objetos permite o armazenamento, processamento e análise de grandezas físicas. Para possibilitar a interação entre máquinas, os dispositivos envolvidos são capazes de atuar na área da computação, comunicação e controle (MAGRANI, 2018).

Para realizar a integração do ambiente físico ao ambiente virtual, a Internet das Coisas pode ser construída por etapas, conforme a Figura 1.

Figura 1 – Etapas da IoT

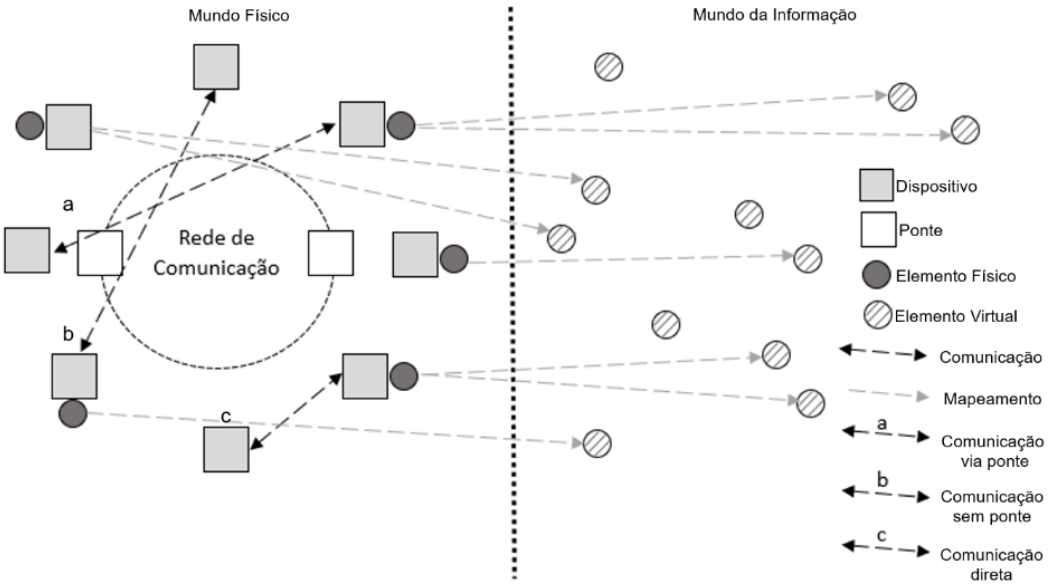

Fonte: (SANTOS, 2016)

A identificação é a etapa de reconhecimento dos objetos a fim de estabelecer uma conexão deles com a Internet. Os sensores fazem parte da etapa de coleta dos dados. Os dados geralmente são armazenados em centros de armazenamento em nuvem ou data Warehouse. Os atuadores controlam o ambiente físico ao reagir de acordo com os dados lidos pelos sensores. A comunicação é responsável por conectar os dispositivos envolvidos. O *Wi-Fi* e o *Bluetooth* fazem parte das principais tecnologias da comunicação que realizam a conexão entre máquinas. A etapa da computação atua diretamente nos dispositivos com unidade de processamento através dos algoritmos locais para permitir a interação entre máquinas. Os processadores, microcontroladores e FPGA (*Field-programmable Gate Array*) são equipamentos geralmente utilizados para computar a interação. A semântica da IoT possibilita a geração das classes de serviços aos extrair as informações dos dispositivos envolvidos para utilização eficiente dos recursos presentes (SANTOS, 2016).

Entre as classes de serviços que a IoT pode gerar estão os serviços de identificação, os serviços de agregação de dados, os serviços de colaboração e inteligência e os serviços de ubiquidade. Os serviços de identificação permitem o monitoramento das entidades físicas através das entidades virtuais. Os serviços de colaboração e inteligência realizam a tomada de decisão a partir dos dados heterogêneos e homogêneos compilados pelos serviços de agregação de dados. Os serviços de ubiquidade garantem a contínua disponibilidade dos serviços de colaboração e inteligência (SANTOS, 2016).

A União Internacional de Telecomunicações (UIT) estabeleceu uma visão técnica geral da IoT ao descrever a relação do mundo físico com o mundo digital da informação, conforme representado na Figura 2.

Figura 2 – Visão técnica da IoT




Fonte: Adaptado de Faccioni Filho (2016)

Conforme observado na Figura 2, todo objeto presente no ambiente físico corresponde a um elemento virtual no mundo da informação, mas existem elementos no mundo da informação que não representam objetos físicos (FACCIONI FILHO, 2016).

Os dispositivos retratados na Figura 2 referem-se aos equipamentos responsáveis por coletar e enviar diferentes tipos de dados para as redes de comunicação. São capazes de se comunicarem com outros dispositivos e podem responder de acordo com as informações recebidas da rede de comunicação. O caminho “a” da Figura 2 corresponde à comunicação por rede de comunicação através de *gateways* (pontes), o caminho “b” passa pela rede de comunicação sem *gateways* e o caminho “c” trata-se da comunicação direta sem a presença da rede de comunicação (FACCIONI FILHO, 2016).

Os dispositivos da IoT podem ser categorizados em transporte e captura de dados, sensor/atuador e geral. Os dispositivos de transporte de dados estão conectados aos objetos físicos para promoverem a comunicação entre o objeto e a rede de comunicação de forma indireta. Os dispositivos de captura de dados são capazes de ler e registrar as informações ao conectarem-se no objeto. Os dispositivos sensores captam e medem parâmetros físicos para transformar as medições em sinais digitais. Os dispositivos atuadores podem reagir no ambiente físico de acordo com os sinais digitais enviados pelas redes de comunicações. Os dispositivos gerais

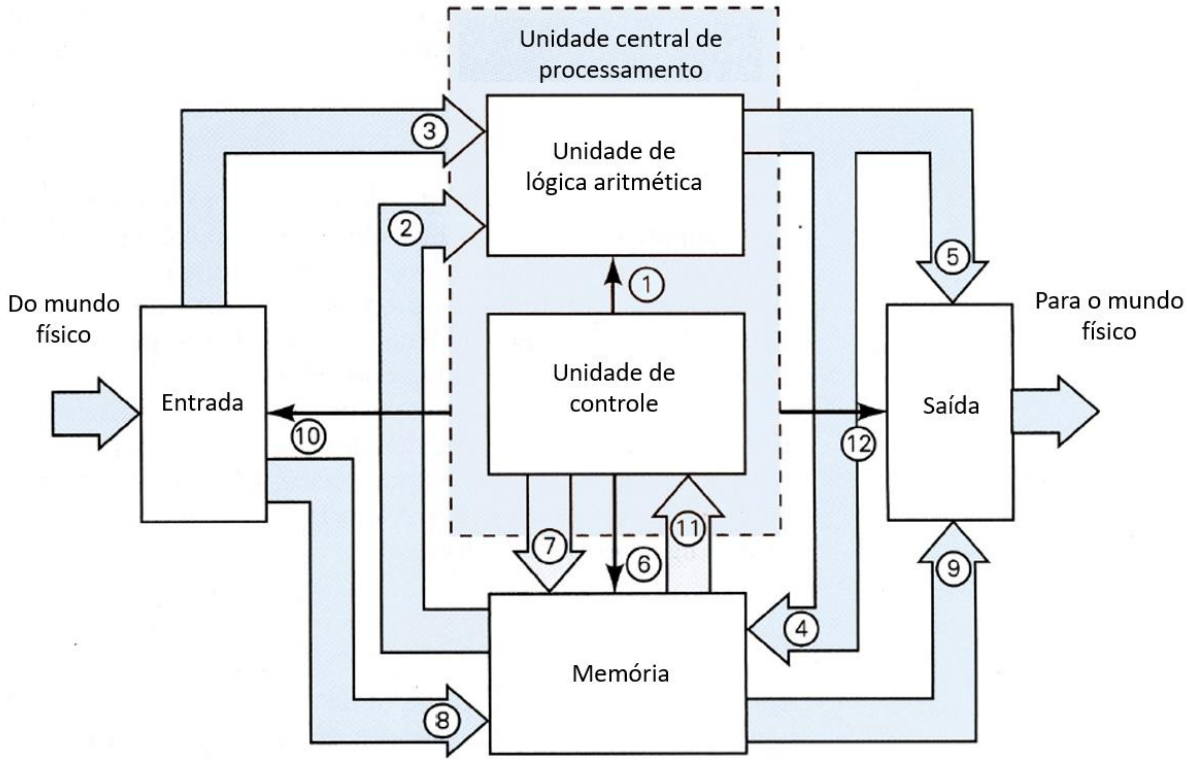
possuem processamento e comunicação própria ao realizar a comunicação direta com as redes de comunicação com ou sem fio (FACCIONI FILHO, 2016).

## 2.2 Microcontroladores

Os microcontroladores são fabricados com uma ampla variedade de recursos inseridos em um *chip* de circuito integrado único a fim de atender aos objetivos de cada aplicação ao executar um programa com um conjunto de operações programadas em sequência e armazenadas na memória interna para guiar a funcionamento do dispositivo de acordo com a utilização destinada (TOCCI, 2011).

De forma geral, cada microcontrolador é composto por uma unidade de lógica aritmética, memória, controle, entrada e saída que podem se conectarem entre si, conforme Figura 3.

Figura 3 – Organização do microcontrolador




Fonte: Adaptado de Tocci (2011)

De acordo com a Figura 3, a unidade de lógica aritmética processa o comando enviado pela unidade de controle (seta 1) e utiliza os dados provenientes da memória

(seta 2) e da entrada (seta 3). Após o processamento, os dados podem ser armazenados na memória (seta 4) ou serem utilizados na saída (seta 5) (TOCCI, 2011).

Pode-se observar na Figura 3 que a unidade de controle sinaliza para a memória quando se deve ler ou gravar os dados (seta 6). A memória pode fornecer a informação alocada de acordo com a posição solicitada (seta 7) e pode transferir as instruções armazenadas no local para a unidade de controle comandar as unidades a fim de executar as operações presentes nas instruções transferidas (seta 11). As informações podem ser gravadas diretamente da unidade de entrada para a memória (seta 8) sob o comando da unidade de controle que sinaliza onde o dado deve ser enviado (seta 10). Os dados armazenados na memória podem ser lidos pela unidade de saída (seta 9) de acordo com a sinalização direcionada da unidade de controle (seta 12) (TOCCI, 2011).

Entre as vantagens de empregar os microcontroladores em um projeto está o baixo custo de aquisição, a dimensão física compacta e o baixo consumo de energia. As vantagens tornaram possível a utilização dos microcontroladores nos circuitos eletrônicos na maioria dos sistemas embarcados, permitindo a implementação em eletrodomésticos, brinquedos, controles de acesso com reconhecimento digital, relógios e outros produtos (MELLO, 2022a).

### 2.2.1 ESP32

O microcontrolador ESP32 foi lançado pela empresa Espressif Systems com a proposta de integrar *Wi-Fi* de frequência 2.4 GHz e *Bluetooth* em um único *chip* com preço acessível e baixo consumo de energia. A Espressif Systems desenvolveu o dispositivo com alto desempenho em potência e radiofrequência para permitir um sistema versátil, robusto e confiável para aplicar em diversos perfis de potência (ESPRESSIF SYSTEMS, 2023).

O ESP32 possui um processador dual core executando instruções da Xtensa LX6, um processador secundário Ultra Low Power de 8 MHz, Wi-Fi de 2,4 GHz, memória *FLASH* de 4 MB, RAM de 520 KB e ROM de 448 KB. Trabalha com tensão de até 3,3 V e suporta protocolos de rede do tipo SSL (*Secure Sockets Layer*), TCP (*Transmission Control Protocol*) e MQTT (*Message Queuing Telemetry Transport*), conforme especificações técnicas do Quadro 1 (OLIVEIRA, J., 2018).

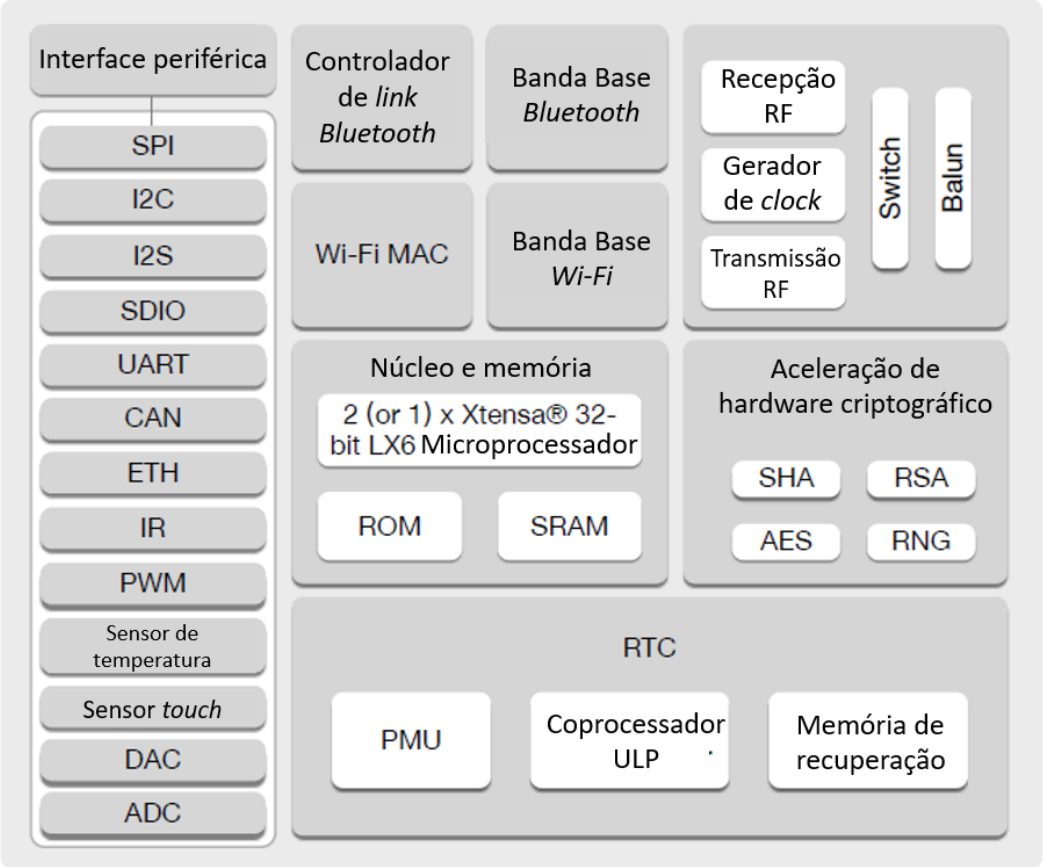
Quadro 1 – Especificações técnicas do ESP32

| **Parâmetros** | **Valores** |
|---|---|
| Alimentação | 2,2 a 3,3 V |
| Corrente de consumo média | 80 mA |
| Temperatura de operação | -40°C a 85°C |
| Processador | Xtensa Dual-Core 32-bit LX6 |
| Processador secundário | ULP 8 MHz |
| Frequência de operação | 80 ~ 240 MHz |
| Memória FLASH | 4 MB |
| Memória RAM | 520 KB |
| Memória ROM | 448 KB |
| GPIOs (Saidas e Entradas) | 34 GPIOs de 3,3 V e 12 mA |
| Wi-Fi | 2,4 GHz |
| Bluetooth | Bluetooth Low Energy v4.2 |
| Protocolos de rede | IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT |

Fonte: (OLIVEIRA, J., 2018)

No diagrama de blocos da Figura 4 pode-se observar o processador dual-core Xtensa LX6 e a memória na parte central. À esquerda, situam-se os blocos de interfaces periféricas contendo conversores analógicos-digitais e digitais-analógicos, sensores de temperatura e de toque, controlador infravermelho programável para transmissão, o protocolo de comunicação *Controller Area Network* (CAN), o controle PWM (*Pulse Width Modulation*) para sinais analógicos, *Ethernet* e outros periféricos. À direita, localizam-se os modelos do acelerador de hardware de criptografia do tipo AES (*Advanced Encryption Standard*), SHA (*Secure Hash Algorithm*), RNG (*Random Number Generator*) e RSA (*Rivest-Shamir-Adleman*) para implementar as operações matemáticas nos algoritmos, aumentar a velocidade de operação e diminuir a complexidade do *software*. No topo encontram-se os blocos de *Wi-Fi* para comunicação com outros dispositivos *Wi-Fi*, *Bluetooth* para interação com dispositivos *Bluetooth* compatíveis e transceptor de radiofrequência. Na parte inferior do diagrama está o relógio de tempo real para manter datas atualizadas, o coprocessador de ultrabaixo consumo e a memória de recuperação (IBRAHIM, 2019).

Figura 4 – Diagrama de blocos do ESP32

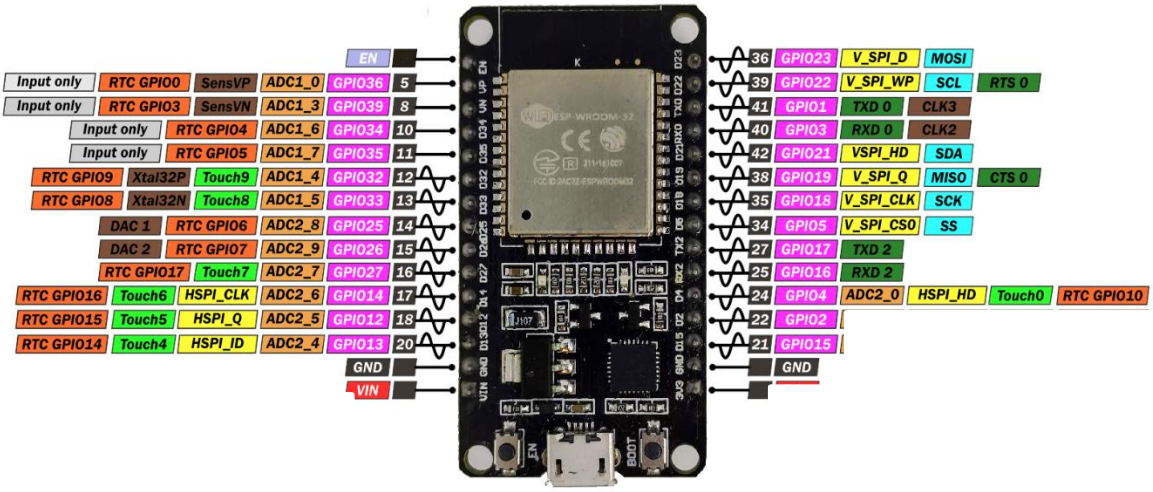



Fonte: Adaptado de ESPRESSIF SYSTEMS (2023)

### 2.2.2 DOIT ESP32 DevKit V1

A fim de facilitar a prototipagem de projetos com o ESP32, foram produzidas diversas placas de desenvolvimento. Uma das placas de desenvolvimento baseadas no ESP32 mais populares no mercado é a DOIT ESP32 DevKit V1 com aproximadamente 5,5 cm de comprimento e 2,8 cm de largura. Conforme Figura 5, a placa possui uma porta mini USB capaz de energizar e realizar a conexão com o computador. Por padrão, a placa recebe +5 V e converte em +3,3 V para chegar na tensão de operação do ESP32. O botão EN reinicia a placa e o botão BOOT prepara a placa para fazer o download do código. O módulo ESP32-WROOM-32 é o módulo padrão inserido na placa por ter o ESP32 no núcleo (MOHANAN, 2022).

Figura 5 – Funções dos pinos da placa DOIT ESP32 DevKit V1




Fonte: Adaptado de Mischianti (2021)

Em cada lado da placa existe um conector com 15 pinos de múltiplas funções destinados para GPIO (*General Purpose Input/Output*), frequência de operação e energização, conforme Figura 5. Os pinos VIN e 3,3V são respectivamente a entrada e a saída do regulador de tensão. O pino GND se refere ao aterramento e o pino ENABLE reinicia o ESP32 quando está conectado com o pino GPD. Os pinos GPIO são identificados no editor de código pelos respectivos números de 0 a 39. Os pinos GPIO 20, 24, 28, 29, 30 e 31 não podem ser acessados. Entre os 34 pinos GPIO presentes no chip, os pinos GPIO 34, 35, 36 e 39 funcionam apenas como entrada, os pinos GPIO 6, 7, 8, 9, 10 e 11 correspondem à interface da memória *flash*, os pinos GPIO 1 e 3 estão relacionados com a porta serial para programar e enviar mensagens de depuração e os pinos de *Strapping* 0, 2, 5, 12 e 15 mudam de comportamento para atuar no processo de inicialização e retornam para a função padrão após finalizar a etapa (MOHANAN, 2022).

A placa possui 10 sensores capacitivos de toque e sustenta até 16 canais PWM independentes com 16 bits de precisão. Os pinos com a possibilidade de operar como saída também funcionam como pinos PWM ao especificar a frequência do sinal, o pino GPIO para o envio do sinal, o ciclo de trabalho e o canal PWM no código. Há 16 canais de entrada de 12 bits disponíveis dos 18 canais existentes para realizar a

conversão de sinais analógicos para digitais e 2 canais de 8 bits para converter sinais digitais em analógicos, conforme Figura 5 (MOHANAN, 2022).

## 2.3 Sensores de Monitoramento da Temperatura e Oxigênio Dissolvido

Nesta seção será apresentado o conjunto de sensores utilizados no desenvolvimento do projeto.

### 2.3.1 Sonda Óptica de Oxigênio Dissolvido

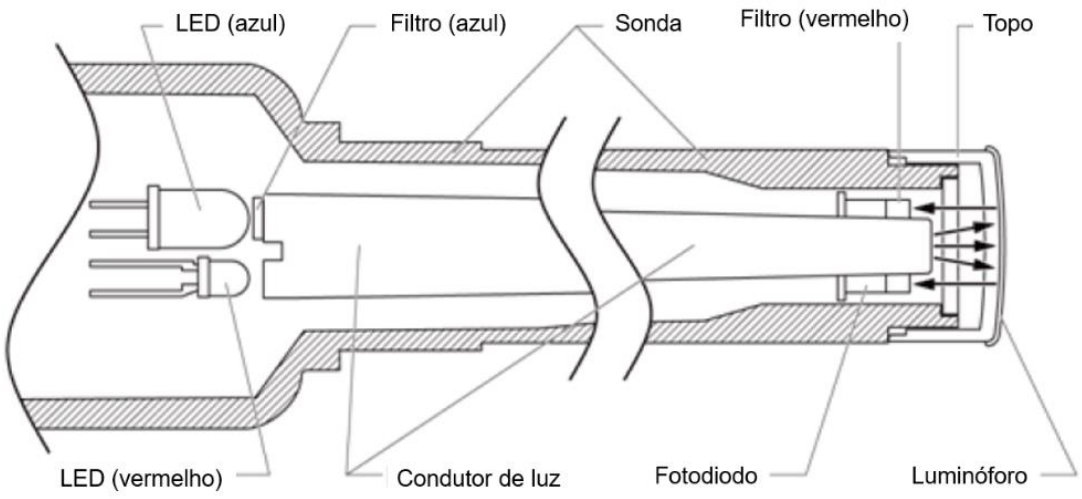
A sonda óptica de oxigênio dissolvido ODO-BTA desenvolvida pela empresa Vernier é um sensor óptico utilizado para medir a concentração do oxigênio dissolvido na água através da luminescência. O equipamento permite selecionar a unidade de medida para mg/L ou porcentagem de saturação (%) utilizando o interruptor presente na Figura 6. A medida absoluta da concentração do oxigênio dissolvido está em mg/L ao quantificar em miligramas o gás oxigênio dissolvido por litro de água. A faixa de medição está entre 0 e 20 mg/L com acurácia de ± 0,2 mg/L para medidas abaixo de 10 mg/L e ± 0,4 mg/L para valores acima de 10 mg/L. A porcentagem de saturação relaciona a concentração de oxigênio com a máxima quantidade de oxigênio que a água consegue conter em pressão e temperatura determinados. Os valores podem ser medidos entre 0 e 300% com acurácia de ± 2% abaixo de 100% e ± 5% acima de 100%. Para ler até 90% do sinal, o sensor pode levar até 40 segundos (VERNIER, 2016).

Figura 6 – Sonda óptica de oxigênio dissolvido ODO-BTA


Fonte: (VERNIER, 2016)

O funcionamento da sonda óptica é baseado no princípio da luminescência reversível ao esfriar um luminóforo com o oxigênio que entra em contato com o topo do dispositivo. Devido ao revestimento de um composto luminescente protegido por uma matriz, a luz azul do LED é transmitida pelo topo para estimular o luminóforo, conforme esquema da Figura 7. Quando a molécula de oxigênio colide com o luminóforo estimulado eletronicamente ocorre a transferência de energia do luminóforo para o oxigênio. Ao transferir a energia, o luminóforo emite uma luz vermelha. O tempo entre a transmissão da luz azul e a emissão da luz vermelha é medido por um fotodiodo para obter a concentração do oxigênio relacionada. Quanto maior for a presença de oxigênio na água, o tempo para a emissão da luz vermelha será menor (VERNIER, 2016).

Figura 7 – Esquema de funcionamento da sonda óptica



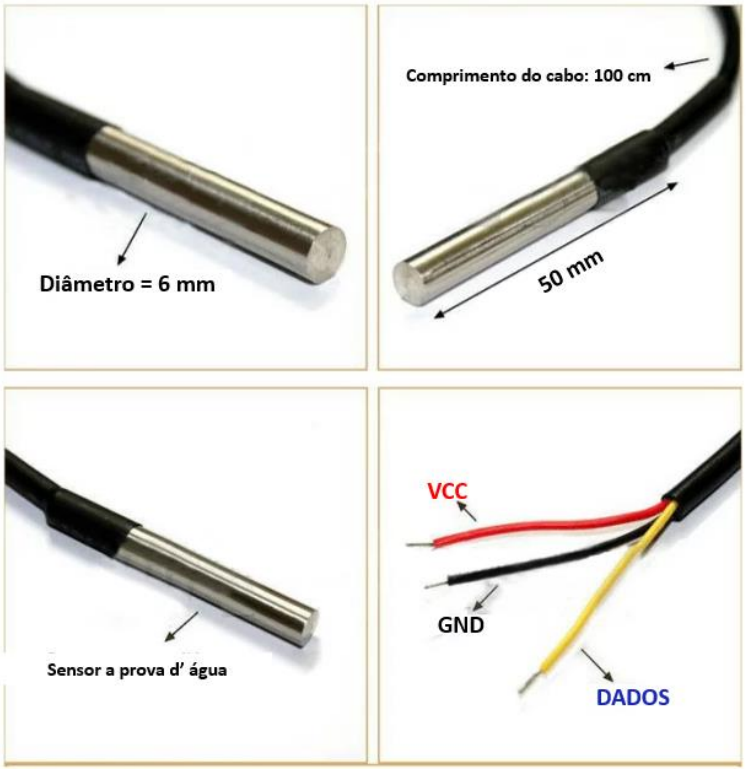

Fonte: Adaptado de Vernier (2016)

Para medir a concentração de oxigênio na água, a parte que abriga o cartão microSD não pode ter contato com a água, a sonda pode permanecer submersa até 30 minutos em ambientes aquáticos de até um metro e o topo é capaz de permanecer submerso por períodos maiores sem danos ao equipamento. Para realizar a compensação de temperatura automática o ponto de metal no topo da sonda precisa estar submerso. É realizada a leitura da pressão por um barômetro interno para compensar a variação da difusão do oxigênio pelo topo e da solubilidade do gás na água (VERNIER, 2016).

### 2.3.2 Sensor de Temperatura

O sensor digital DS18B20 é capaz de realizar a leitura da temperatura de ambientes rígidos e aquáticos. O dispositivo também pode ser empregado nos sistemas de controle de termostato, climatização, termômetros e produtos sensíveis ao calor. O equipamento envolto por um material à prova d'água possui a ponta de 50 mm de comprimento envolvida por aço inoxidável com diâmetro de 6 mm, conforme Figura 8 (LOUSADA, 2020).

Figura 8 – Características físicas e pinagem do DS18B20




Fonte: Adaptado de Lousada (2020)

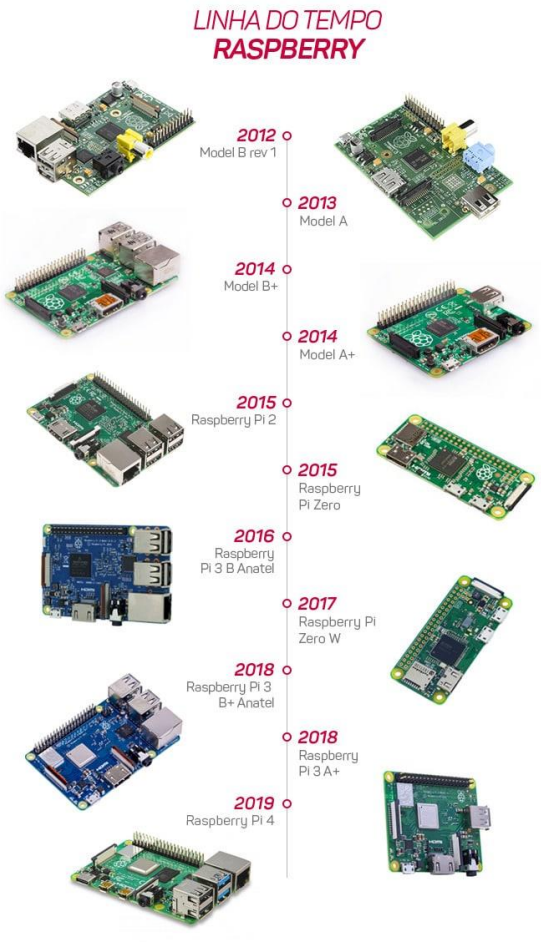
O sensor mede a temperatura entre -55°C e 125°C com precisão de ±0,5°C e resolução configurável de 9 a 12 bits em um período de atualização de até 750 ms. Para possibilitar a ligação de vários sensores DS18B20 ao microcontrolador utilizando apenas uma porta, o sensor comunica-se através de um protocolo de barramento de fio único e permite a identificação de cada sensor com ID único de 64 bits. A fim de eliminar a necessidade de alimentação externa, o dispositivo possui a função de energia parasita para alimentar o sensor por meio de um barramento de dados (OLIVEIRA, E., 2018).

## 2.4 Raspberry Pi

O Raspberry Pi é um computador de única placa que integra os componentes em uma multiplataforma e permite a inserção do monitor, mouse, teclado e outros periféricos através de conectores. A primeira versão do dispositivo foi lançada no Reino Unido em 2012 pela Fundação Raspberry Pi com o intuito de estimular o estudo da programação por estudantes das escolas e universidades europeias ao fornecer um dispositivo de qualidade a um preço acessível (MELLO, 2022b).

A primeira versão do Raspberry Pi contou com um circuito integrado Broadcom BCM2835, um processador de 700 MHz e uma memória RAM de 256 MB. Ao longo dos anos foram desenvolvidos diversos modelos de Raspberry Pi para serem utilizados em várias aplicações de programação e controle, conforme Figura 9 (GUSE, 2020).

Figura 9 – Evolução do Raspberry Pi




Fonte: (GUSE, 2020)

A entrada para cartão micro SD com a gravação do sistema operacional é uma das características comuns de todos os modelos de Raspberry Pi. As versões comercializadas atualmente contam com conector Ethernet, entrada USB para periféricos, entrada HDMI para monitor e saída P2 para áudio (GUSE, 2020).

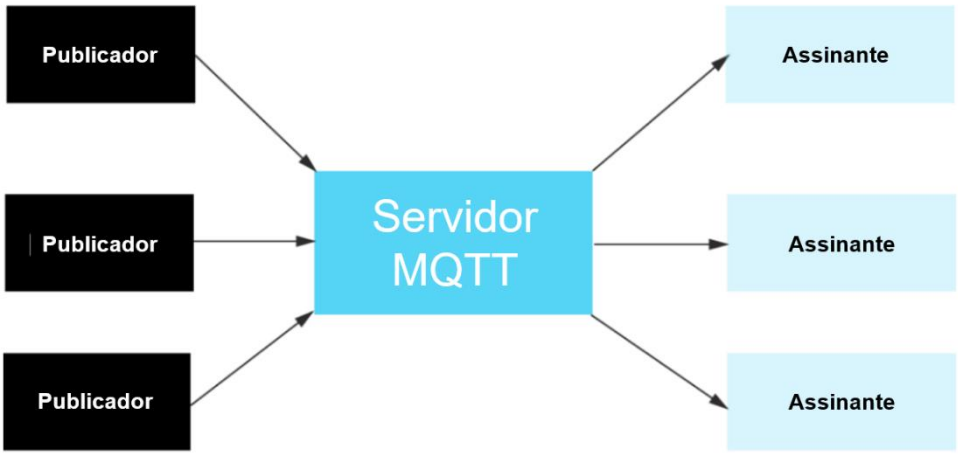
O modelo Raspberry Pi 3b+ foi lançado em 2018 com um circuito integrado Broadcom BCM2837B0 de 4 processadores ARM Cortex-A53 de 64 bits e 1,4 GHz, memória RAM de 1 GB, GPU VideoCore IV 3D, Bluetooth 4.2 e Wi-Fi Dual Band 2,4 GHz e 5 GHz. O dispositivo também possui um conector Gigabit Ethernet até 300 Mbps, 4 portas USB 2.0 para conectar periféricos e 40 pinos do tipo GPIO (NATH, 2020).

## 2.5 Protocolo MQTT

O protocolo de comunicação MQTT foi desenvolvido pelas empresas IBM e Eurotech com base no protocolo *Transmission Control Protocol/Internet Protocol* (TCP/IP) para troca de mensagens assíncronas através da comunicação entre cliente e servidor ao permitir a comunicação remota entre dispositivos com tamanho de mensagem limitado em rede com alta latência e largura de banda restrita (BASÍLIO, 2018).

Os clientes podem enviar e receber as mensagens roteadas pelo servidor broker através de mecanismos de publicação e subscrição de mensagens. O servidor broker gerencia as mensagens recebidas ao enfileirar e disparar os dados recebidos dos *publishers* (publicadores) para os *subscribers* (assinantes). Uma comunicação TCP/IP ou TLS (*Transport Layer Security*) criptografada conecta o cliente ao broker para assinar tópicos específicos de armazenamento de dados para cada estrutura de publicação (*publishers*) com o propósito de fazer o cliente ler os dados do tópico (*subscribers*) com a possibilidade de configurar o limite de interação dos clientes com a mensagem dos tópicos, conforme Figura 10 (BASÍLIO, 2018).

Figura 10 – Princípio de funcionamento do protocolo MQTT




Fonte: Adaptado de Basílio (2018)

Devido à versatilidade e compatibilidade em ter bibliotecas aceitas em diversas linguagens de programação e segurança ao escalar em ambientes de redes instáveis, o MQTT é comumente implementado em sistemas embarcados para realizar a comunicação entre dispositivos da Internet das Coisas (BASÍLIO, 2018).

## 2.6 *Softwares* e Serviços

Nesta sessão serão levantadas informações acerca dos softwares e serviços utilizados para aplicação do projeto.

### 2.6.1 Visual Studio Code

O editor de código Visual Studio Code foi desenvolvido em código aberto pela empresa Microsoft para editar código em diferentes linguagens de programação com terminal de comandos agregado. É possível customizar a aparência e alterar o comportamento das funcionalidades na aba de configurações do editor. Ao realizar o login no editor, pode-se salvar as configurações alteradas na nuvem para sincronizar em diferentes computadores (HANASHIRO, 2021).

Uma das principais vantagens do Visual Studio Code é a possibilidade de instalar extensões para ampliar as funcionalidades do editor. A extensão PlatformIO é comumente utilizada para sistemas embarcados e projetos de IoT devido às ferramentas para desenvolvimento em C/C++, à disponibilidade de diversas

bibliotecas, à compatibilidade com mais de 778 tipos de placas e 20 *frameworks*, entre os quais pode-se citar o Arduino e o ESP-IDF (BAUMEISTER, 2020).

A possibilidade de desenvolver em JavaScript ou TypeScript e publicar novas extensões proporciona um crescente aumento das funcionalidades do editor. A simplicidade de trabalhar com uma ferramenta de código aberto e estrutura planejada com customização alternativa torna o Visual Studio Code um dos *softwares* de edição de código mais utilizados atualmente (HANASHIRO, 2021).

### 2.6.2 Node-RED

Node-RED é uma plataforma de programação visual para simplificar os códigos em uma sequência de blocos de tarefas. Foi desenvolvida pela empresa IBM em 2013 para facilitar a comunicação dos *hardwares* com os *softwares* e serviços através da realização de tarefas em blocos. Cada bloco de código predefinido forma um nó para executar uma tarefa. Podem existir nós de entrada, de processamento e de saída e ao serem conectados entre si formam um fluxo (LEA, 2016).

Inicialmente, a ferramenta Node-RED foi desenvolvida como um projeto de código aberto pelos pesquisadores Nick O'Leary e Dave Conway-da empresa IBM Emerging para atender as próprias necessidades de conectar os sensores com os sistemas de maneira simplificada. Com o rápido aumento da base de usuários da ferramenta e a crescente comunidade de desenvolvedores ativa, foi possível criar nós novos e contribuir para a utilização do Node-RED em várias áreas de aplicação (LEA, 2016).

Por ser um recurso de programação em fluxo, o Node-RED pode ser aplicado na maioria das aplicações IoT ao trabalhar com eventos empacotados em mensagens no decorrer da sequência de nós para uma entrada do mundo real ser processada no mundo digital e resultar em uma saída com a complexidade da interação inclusa nos blocos sem precisar se atentar para os detalhes da programação envolvida no processo (LEA, 2016).

### 2.6.3 Banco de Dados InfluxDB

O InfluxDB é um banco de dados focado em otimizar desempenho e modelagem para trabalhar com séries temporais. Todos os dados são registrados no

InfluxDB acompanhados de data e hora específicas do registro gravados em tipo timestamp na coluna time (CRUZ, 2017).

As colunas no banco de dados podem ser estruturadas em *Fields* ou *Tags*. As colunas *Fields* possuem o elemento *Field Key* para armazenar o nome da coluna e o *Field Value* para armazenar os valores da medição seguindo o modelo chave-valor. As colunas *Tags* armazenam dados convertidos em *string* de acordo com o formato chave-valor e geralmente são utilizadas como critério de busca nas consultas do banco devido à indexação da estrutura para aumentar a rapidez do processo de busca. As colunas *Fields* e *Tags* são agrupadas em bases conhecidas como *Measurements*, equivalentes as tabelas no modelo de entidade relacionamento. O conjunto de tabelas *Measurements* são agrupadas em um *Database*. Ao realizar uma consulta em tabelas *Measurements* configurando o filtro de acordo com a parametrização das *Tags*, o conjunto de dados resultado corresponde a uma série temporal (CRUZ, 2017).

O InfluxDB é equipado com um conjunto de ferramentas para auxiliar o trabalho com dados de séries temporais ao facilitar a escrita das consultas através da linguagem específica Flux. Os recursos de visualização integrados ao banco e a integração compatível com outras ferramentas de visualização, como o Grafana, tornam o InfluxDB um banco de dados versátil para armazenar séries temporais de alto volume com o mínimo impacto do sistema operacional (MAHLER, 2022).

### 2.6.4 Grafana

A plataforma Grafana é uma ferramenta de código aberto desenvolvida para consulta e análise de dados de modo organizado e informativo. A ferramenta Grafana é capaz de suportar diversas fontes de dados e realizar consultas com um editor exclusivo. Por ser um processo que permanece em constante execução no computador, a plataforma Grafana pode ser acessada a qualquer momento através de um navegador web (LEPPÄNEN, 2021).

Com a plataforma Grafana, é possível montar uma interface gráfica flexível para o usuário final monitorar as métricas e logs dos dados no painel. As métricas podem ser utilizadas para comparar diferentes fontes de dados e consultas em intervalos de tempo distintos no mesmo painel através das visualizações fragmentadas. Os logs podem ser consultados rapidamente e transmitidos ao vivo. Pode ser realizada a

alternância entre métricas e logs sem alterar os filtros de rótulo selecionados anteriormente (KHANH, 2021).

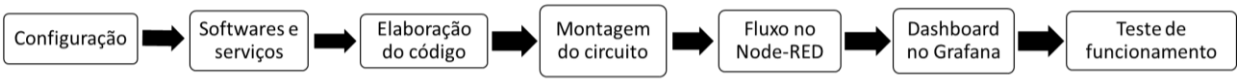
A ferramenta permite a reutilização de dashboards dinâmicos em múltiplos painéis através das variáveis de modelo, a presença de diversas fontes de dados por gráfico, o registro e rastreamento de um gráfico de eventos para montar um metadados completo, a utilização de filtros *Ad-hoc* para aplicar um novo valor-chave em todas as consultas da mesma fonte de dados automaticamente e a configuração de alertas para enviar notificações de acordo com os valores das métricas e logs (KHANH, 2021).

É possível instalar ferramentas externas (*plugins*) de fonte de dados, painéis e aplicativos para ampliar a área de aplicação da plataforma. Os *plugins* de fonte de dados fazem a interação com as fontes de dados externas e devolvem os dados em formato compatível para que seja viabilizada a comunicação com a ferramenta Grafana. Os *plugins* de painéis adicionam novos tipos de visualizações para enriquecer a análise de dados das consultas, possibilitam a navegação entre os dashboards desenvolvidos e permitem o controle de sistemas externos. Os *plugins* de aplicativos conciliam as fontes de dados e os painéis para proporcionar uma experiência de monitoramento personalizada ao usuário (KHANH, 2021).

## 3 MATERIAIS E MÉTODOS

Conforme Figura 11, a construção do protótipo iniciou-se com a etapa de configuração do sistema envolvendo o Raspberry Pi e os periféricos seguida pela etapa de instalação do *software* Mosquitto, do banco de dados InfluxDB, da ferramenta Node-RED e da aplicação Grafana no sistema. Em seguida, elaborou-se o código de programação dos microcontroladores para captação e envio dos dados e montou-se o circuito de ligação entre os microcontroladores e os sensores. Após o carregamento do código nos ESP32 e a conexão física dos dispositivos, foi gerado um fluxo no Node-RED para que os dados enviados para o Raspberry Pi fossem armazenados no banco de dados InfluxDB. A partir dos dados armazenados no banco, foi possível criar um dashboard na aplicação Grafana para acompanhar a variação da temperatura e oxigênio dissolvido da água utilizada no teste.

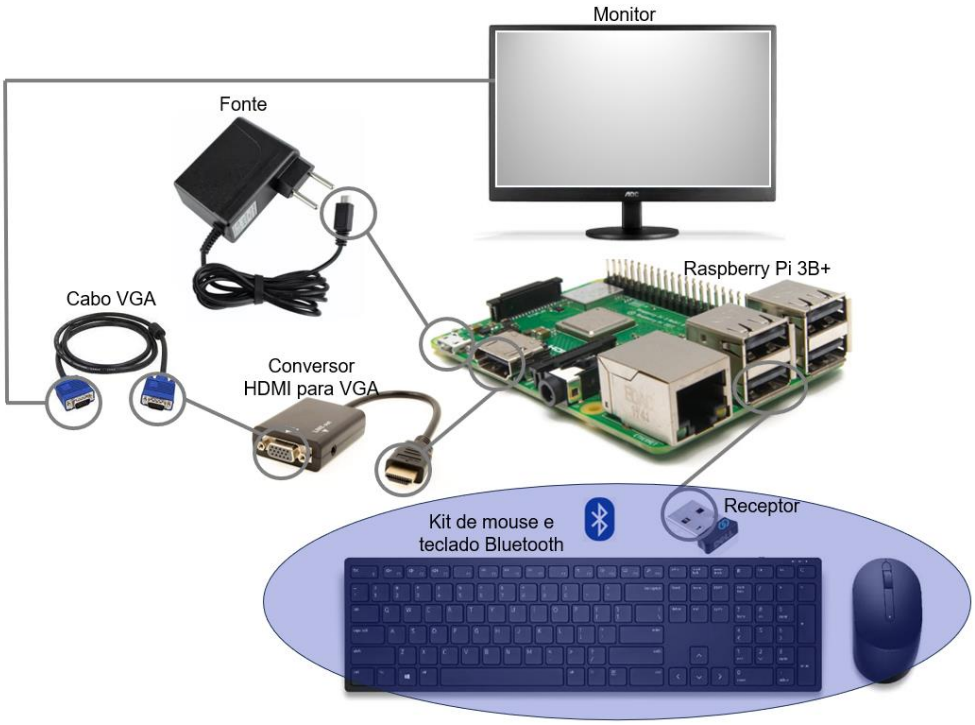
Figura 11 – Etapas de construção do protótipo




Fonte: Próprio Autor

Para conectar os periféricos ao Raspberry Pi 3B+, utilizou-se a uma fonte de 5V para energização, um cabo VGA e um cabo adaptador conversor HDMI para VGA para conectar o monitor na porta HDMI do dispositivo e conectou-se o receptor Bluetooth do kit de mouse e teclado em uma das portas USB, conforme Figura 12.

Figura 12 – Circuito esquemático do sistema com Raspberry Pi e periféricos




Fonte: Próprio Autor

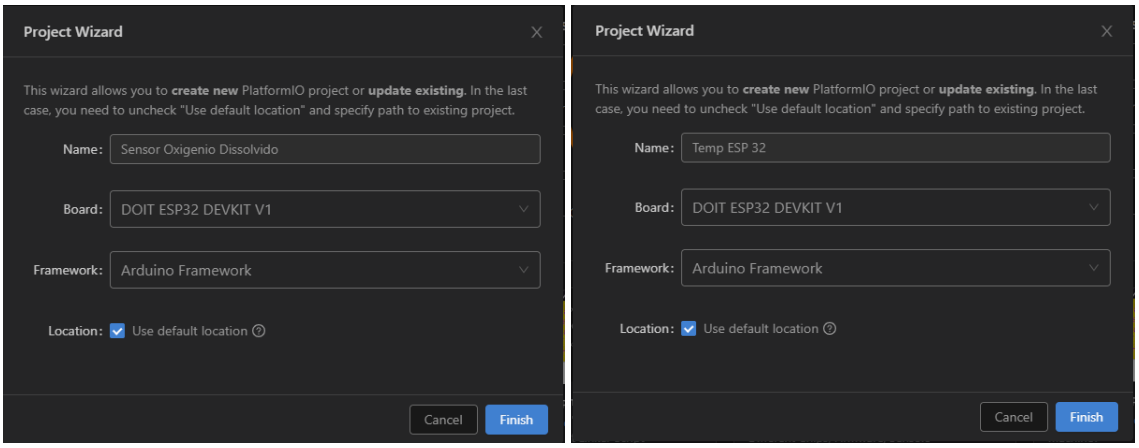
Inicialmente configurou-se o Raspberry Pi com a instalação prévia do sistema operacional Raspbian para trabalhar como servidor broker MQTT ao instalar e configurar o cliente e message broker de *software* livre Mosquitto. Para proteger o servidor broker, foi desabilitado o acesso anônimo e criado um usuário e senha de administrador.

Posteriormente, realizou-se a instalação e configuração do banco de dados influxDB. Foi criado um usuário e senha de administrador para controlar o acesso ao banco através da autenticação. Para armazenar os dados dos sensores ao longo do tempo, foi criado o banco de dados vazio chamado sensorsTempOD para ser automaticamente preenchido sem necessitar de tabelas previamente criadas. Através do sistema de controle, foi possível habilitar e configurar a inicialização automática do InfluxDB com a ativação do Raspberry Pi.

Para intermediar a relação entre os dados enviados via MQTT e o banco de dados InfluxDB, foi instalada a ferramenta Node-RED. A fim de apresentar as informações atualizadas de forma gráfica, foi realizada a instalação da ferramenta de aplicação web de código aberto Grafana. Ambas as ferramentas foram habilitadas para iniciar o serviço automaticamente com o início do Raspberry Pi.

Para realizar a programação, foi instalado o editor de código Visual Studio Code (VS Code) em ambiente Windows. Para programar a placa DOIT ESP32 DevKit V1, foi necessário instalar a versão 3.10.1 do Python e agregar a extensão PlataformIO ao VS Code. Foi criado um projeto para monitoramento da temperatura e outro projeto para monitoramento do oxigênio dissolvido na água. Escolheu-se elaborar um projeto para cada parâmetro a fim de programar de acordo com as particularidades de cada sensor de forma independente. Ambos os projetos utilizaram o *framework* do Arduino e foram configurados para a placa DOIT ESP32 DevKit V1, conforme Figura 13.

Figura 13 – Criação dos projetos no VS Code




Fonte: Próprio Autor

No arquivo main.cpp na pasta src de cada projeto foi escrito o código conforme Apêndice A e B para medir e armazenar as leituras de temperatura e oxigênio dissolvido, respectivamente.

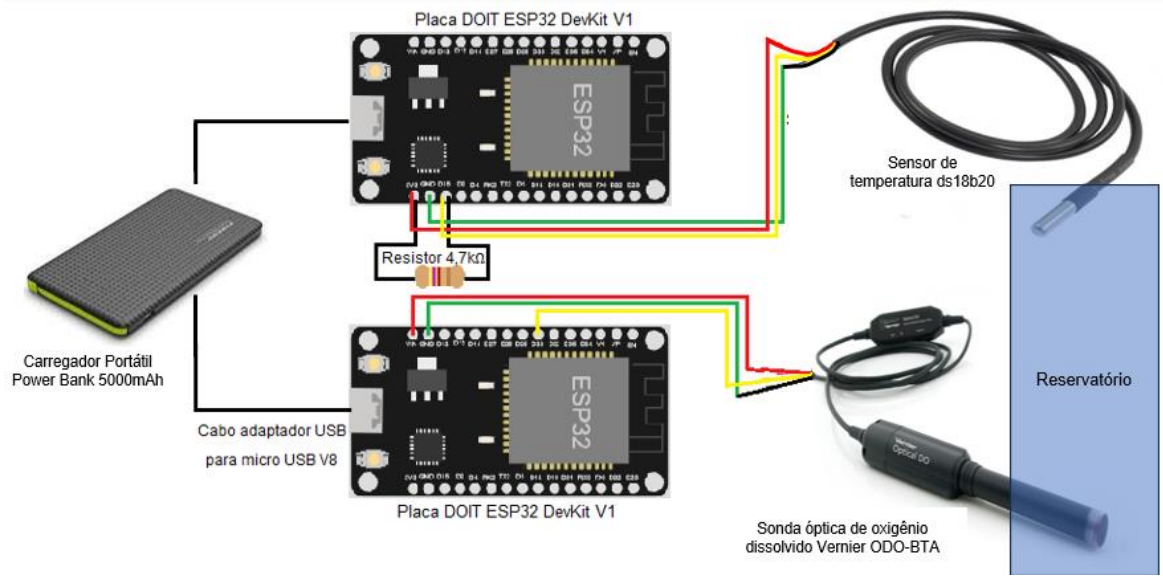
Em ambos os códigos, primeiramente é iniciada a comunicação serial para observar a depuração e o carregamento do código no microcontrolador. Em seguida, inicia-se a conexão do ESP32 à rede Wi-Fi e o status da conexão é verificado em intervalos de 100 ms. O código continuará tentando realizar a conexão em até 120 tentativas antes de reiniciar a placa. Para continuar acompanhando a depuração do código pelas notificações no terminal após conectar à rede Wi-Fi, é necessário chamar a biblioteca RemoteDebug. O Bluetooth foi desabilitado para não ultrapassar a capacidade de memória do ESP32 ao utilizar as bibliotecas Wi-Fi e Bluetooth no mesmo código.

Posteriormente, inicia-se a comunicação pelo protocolo MQTT com o servidor broker instalado no Raspberry Pi pela porta padrão 1883. Conforme Apêndice A, o

sensor de temperatura é iniciado com a indicação do pino conectado e a configuração da instância One Wire a fim de estabelecer comunicação com outros dispositivos One Wire. A temperatura captada pelo sensor a cada 5 segundos é enviada para o tópico TOPIC_H2O_TEMP do servidor broker MQTT. De acordo com Apêndice B, a inicialização do sensor de oxigênio dissolvido conecta o pino ao conversor analógico digital para indicar a entrada da placa utilizada como entrada analógica. O oxigênio dissolvido é obtido através da conversão da tensão média das amostras do período e publicado no tópico TOPIC_H2O_OD do servidor broker MQTT. Os dois códigos em anexo apresentam uma função de verificação da conexão com a rede Wi-Fi e o servidor broker MQTT. Se alguma das conexões não estiver ativa será realizado tentativas de conexão antes de ser necessário reiniciar a placa.

Para realizar a montagem do circuito de captação e envio de dados conforme Figura 14, conectou-se o sensor DS18B20 e o sensor Vernier ODO-BTA nas placas DOIT ESP32 DevKit V1 e utilizou o carregador portátil Power Bank 5000mAh para alimentar ambas as placas. Utilizou-se um ESP32 para comandar cada sensor separadamente a fim de possibilitar a programação individual de acordo com as particularidades de cada sensor e evitar que as falhas em um sensor afetem a coleta de dados do outro sensor. Devido ao protocolo *One Wire* utilizado pelo sensor DS18B20, foi instalado um resistor *pull-up* de 4,7 kΩ entre o pino GPIO 15 e o pino 3V3 a fim de garantir a estabilidade da comunicação entre sensor e microcontrolador ao manter o pino GPIO com nível lógico alto quando não há dados sendo transmitidos para evitar flutuações. Devido à precisão própria do sensor Vernier ODO-BTA, ele foi conectado diretamente no pino GPIO 33 da placa.

Figura 14 – Circuito esquemático de ligação dos microcontroladores e sensores




Fonte: Próprio Autor

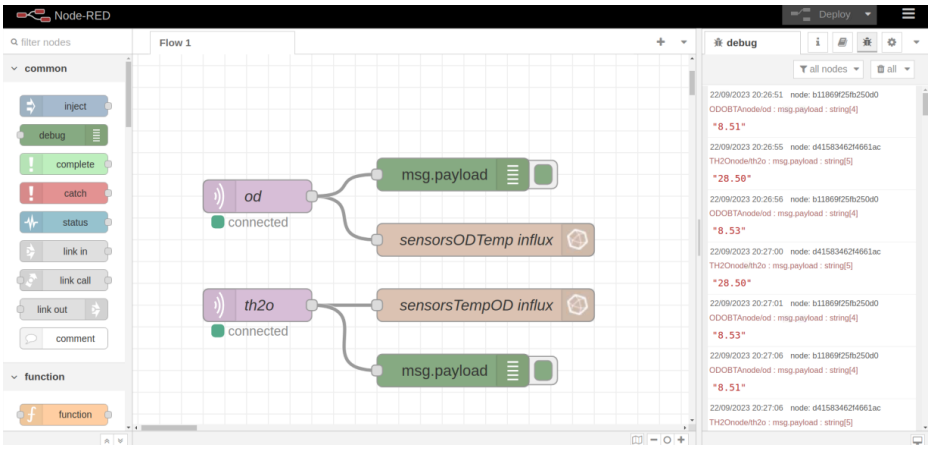
Após carregar o código na placa ESP32 utilizando um cabo adaptador USB para micro USB e finalizar a montagem do circuito das Figuras 11 e 13, foi montado o protótipo conforme Figura 15.

Figura 15 – Protótipo

Fonte: Próprio Autor

Após instalar e configurar as ferramentas no Raspberry Pi e montar o protótipo da Figura 15, acessou-se a plataforma Node-RED pelo navegador web e criou-se o fluxo da Figura 16 para armazenar os dados transferidos via MQTT no banco InfluxDB.

Figura 16 – Fluxo construído no Node-RED




Fonte: Próprio Autor

De acordo com a Figura 16, os nós od e th2o fazem a conexão com o servidor broker MQTT e assinam o tópico ODOBTAnode/od e TH2Onode/th2o, respectivamente. Em cada um dos nós de entrada está conectado um nó msg.payload para visualizar os valores recebidos pelo sensor na aba debug e um nó influx out para enviar a informação transferida via MQTT para o banco de dados sensorsTempOD armazenar com as medidas ODOBTAnode/od e TH2Onode/th2o.

Em seguida, acessou-se a ferramenta Grafana pelo navegador para empregar os dados armazenados no banco sensorsTempOD como fonte de dados de um *dashboard* criado para monitorar a temperatura e oxigênio dissolvido. Utilizou-se gráficos de linha para analisar a variação dos parâmetros em ordem cronológica, *gauges* para destacar as leituras relevantes do período e histogramas para salientar a frequência dos valores durante o período no dashboard.

Para verificar se o protótipo está efetivamente funcionando, inseriu-se 900 mL de água em temperatura ambiente no reservatório e variou-se a temperatura e oxigênio dissolvido da amostra. Para variar a temperatura, adicionou-se de forma intercalada 100 mL de água gelada (em torno de 11°C) e 100 mL de água quente (aproximadamente em 60°C) em períodos de 30, 60 e 90 segundos. Para simular a variação do oxigênio dissolvido, empregou-se o princípio da agitação mecânica ao movimentar manualmente uma espátula nos primeiros 30 segundos de cada período intercalando entre velocidade alta e baixa a fim de criar turbulência na superfície para transferir oxigênio do ar para a água.

# 4 RESULTADOS

Para a realização do projeto foi realizado um investimento total de aproximadamente R$ 2.373,00 para a compra dos materiais utilizados, conforme Tabela 1.

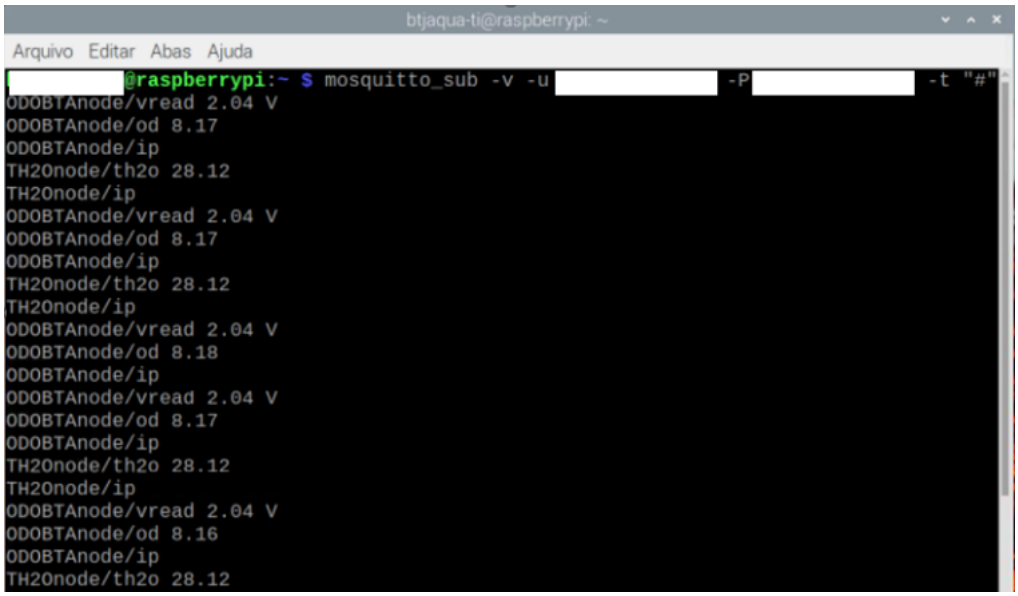
Tabela 1 – Custo do protótipo

| Material | Quantidade | Valor Unitário (R$) |
|---|---|---|
| Placa DOIT ESP32 DevKit V1 | 2 | 42,9 |
| Sensor DS18B20 | 1 | 21,99 |
| Sonda Vernier ODO-BTA | 1 | 990,62 |
| Carregador Portátil 5000mAh | 1 | 46,66 |
| Cabo USB para micro USB V8 | 2 | 14,99 |
| Fonte 5V + Raspberry PI 3B + | 1 | 797 |
| Monitor 15 polegadas + Cabo VGA | 1 | 299,68 |
| Cabo HDMI para VGA | 1 | 22,49 |
| Kit mouse e teclado sem fio | 1 | 78,99 |

Fonte: Elaboração do próprio autor.

Após carregar o código na placa ESP32 e montar o protótipo da Figura 15, foi realizado um teste de comunicação entre os sensores e o servidor broker ao utilizar o cliente Mosquitto para assinar os tópicos e visualizar as publicações enviadas para o servidor broker no terminal do Raspberry Pi, conforme Figura 17.

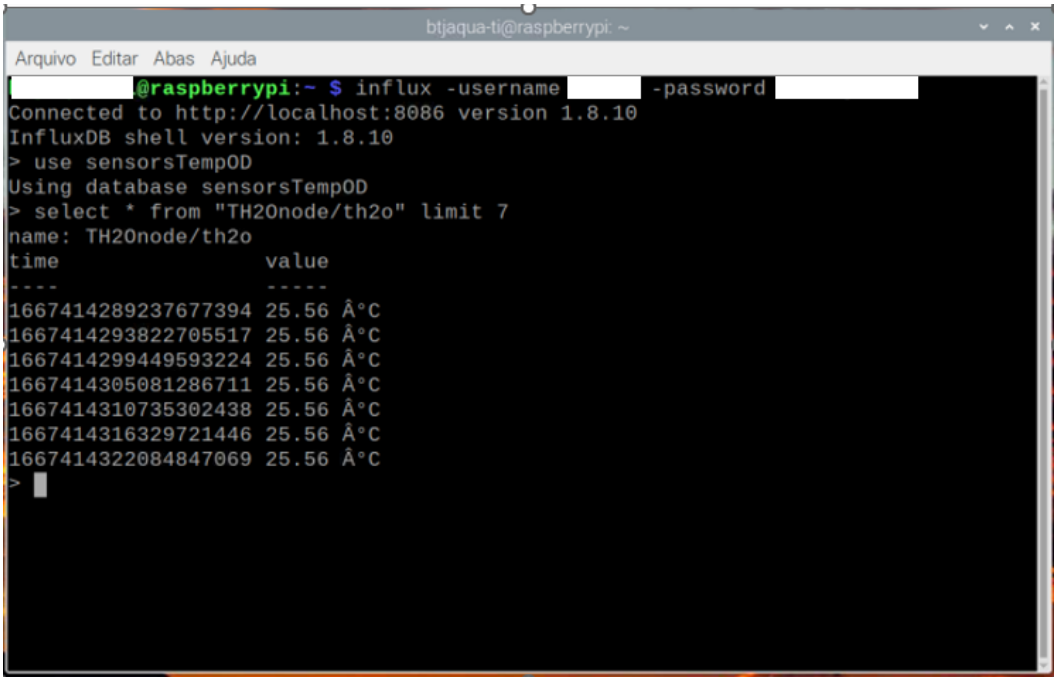
Figura 17 – Teste de comunicação via MQTT entre sensor e servidor broker




Fonte: Próprio Autor

Depois de montar o fluxo da Figura 16 utilizando a ferramenta Node-RED como uma ponte para que os dados recebidos via protocolo MQTT no servidor *broker* fossem armazenados no banco sensorsTempOD, verificou-se que os dados foram armazenados corretamente no banco de dados InfluxDB ao fazer uma consulta nas medidas ODOBTAnode/od e TH2Onode/th2o pelo terminal do Raspberry Pi, conforme Figura 18.

Figura 18 – Verificação dos dados armazenados no banco InfluxDB

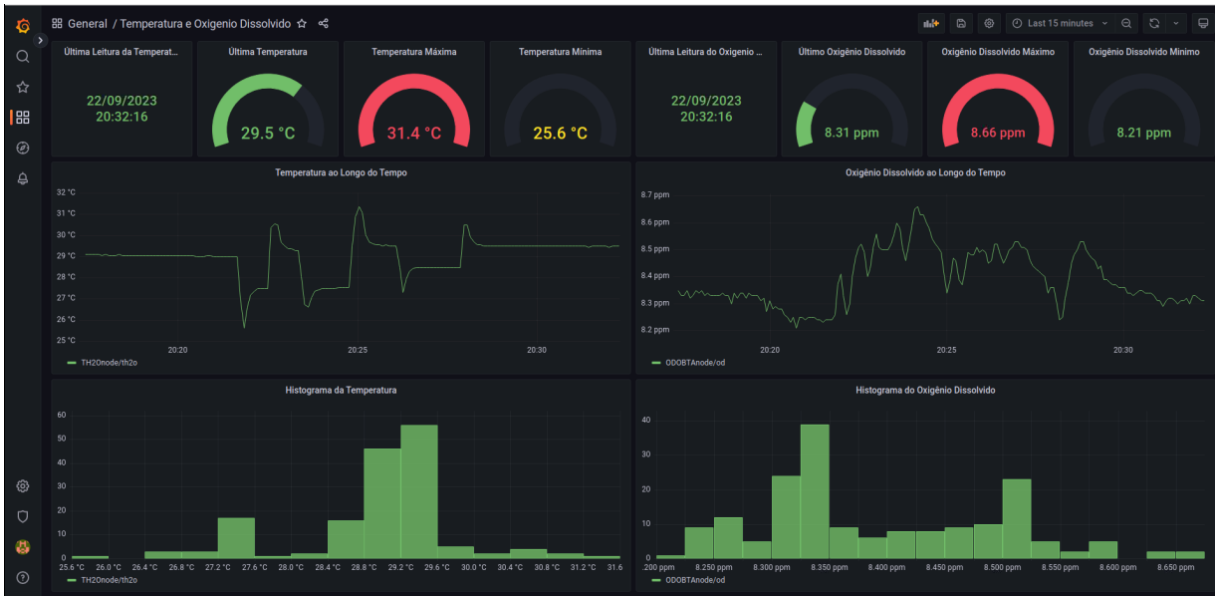



Fonte: Próprio Autor

Com o fluxo na ferramenta Node-RED montado conforme Figura 16 e a verificação dos dados armazenados no banco InfluxDB realizada de acordo com a Figura 18, foi acessada a ferramenta Grafana pelo navegador para salvar a conexão com o banco sensorsTempOD como fonte de dados InfluxDB para ser utilizada nos dashboards. Foi elaborado um dashboard com atualização automática para monitoramento das medições de temperatura e oxigênio dissolvido captadas pelos sensores nos últimos 15 minutos. Para testar a eficiência do protótipo acompanhando a variação no dashboard desenvolvido no Grafana foi feito um teste de funcionamento ao variar a temperatura e oxigênio dissolvido adicionando uma amostra de água gelada intercalada de outra amostra de água quente em períodos de 30, 60 e 90 segundos enquanto gerava-se turbulências de alta e baixa velocidade com uma

espátula nos primeiros 30 segundos de cada período. Os resultados do teste de funcionamento podem ser observados na Figura 19.

Figura 19 – Dashboard de monitoramento dos parâmetros




Fonte: Próprio Autor

A partir do resultado do teste visualizado na Figura 19, foi possível perceber nos gráficos de linha que os intervalos de temperatura constante entre os picos de oscilação aumentaram conforme o aumento do período de teste e que o oxigênio dissolvido do reservatório começou a oscilar de forma crescente no início do teste e com o aumento dos períodos o oxigênio oscilou de modo decrescente. Destacou-se nos *gauges* que a temperatura da amostra ficou entre 25,6°C e 31,4°C e o oxigênio dissolvido variou na faixa entre 8,21mg/L e 8,66 mg/L ao considerar que 1 ppm equivale a 1mg/L. Observou-se nos histogramas que a faixa de temperatura lida na maior parte do teste foi de 29,2°C a 29,6°C e o oxigênio dissolvido teve a maior frequência na faixa de 8,3 mg/L a 8,35 mg/L.

Entre as dificuldades enfrentadas durante o desenvolvimento do projeto, pode-se destacar o gerenciamento de tempo ao realizar as medições de bancadas em um período curto de teste.

A partir da construção do protótipo foi possível constatar que o projeto pode ser expandido para o monitoramento de qualidade da água dos tanques de produção a fim de auxiliar na tomada de decisão para evitar prejuízos com a mortalidade dos peixes devido aos fatores ambientais.

## 5 CONCLUSÃO

Com a montagem e execução do protótipo apresentado, verificou-se que é possível realizar o monitoramento dos principais parâmetros de qualidade da água ao enviar as leituras de temperatura e oxigênio dissolvido captadas pelos sensores controlados pelos módulos ESP32 para um banco de dados InfluxDB com o auxílio da ferramenta Node-RED através do protocolo de comunicação MQTT que permitiu a análise gráfica dos dados em um dashboard na plataforma Grafana atualizado automaticamente.

No teste de funcionamento acompanhado no dashboard na plataforma Grafana concluiu-se que o protótipo captou de forma eficiente a temperatura e o oxigênio dissolvido do reservatório ao observar no gráfico de linhas que os intervalos de temperatura constante entre os picos de oscilação aumentaram obedecendo o aumento do período de teste de 30, 60 e 90 segundos. Observou-se durante o teste que a temperatura do reservatório variou entre 25,6°C e 31,4°C e o oxigênio dissolvido permaneceu na faixa entre 8,21mg/L e 8,66 mg/L comprovando que o valor aumentou de forma crescente a partir do início do teste, mas com o aumento do período de teste oxigênio dissolvido oscilou de forma decrescente.

O sucesso do protótipo mostrou que é possível realizar o monitoramento da qualidade da água em tanques de produção com a finalidade de alertar as variações dos parâmetros em tempo hábil para realizar uma tomada de decisão capaz de diminuir os prejuízos com a mortalidade dos peixes.

Para trabalhos futuros, pode-se abranger outros parâmetros de qualidade da água ao incluir no protótipo um sistema de medição do potencial hidrogeniônico (pH), da transparência e da amônia na amostra testada.

## REFERÊNCIAS


ASHTON, Kevin. **That 'Internet of Things' Thing**. 2009. RFID Journal. Disponível em: https://www.itrco.jp/libraries/RFIDjournal-That%20Internet%20of%20Things%20Thing.pdf. Acesso em: 27 nov. 2022.

AYROZA, Luiz Marques da Silva. **Piscicultura**. Campinas: Cati, 2011. 246 p.

BASÍLIO, Shirley. **Conhecendo o protocolo MQTT**. 2018. Disponível em: https://blogmasterwalkershop.com.br/outros/conhecendo-o-protocolo-mqtt. Acesso em: 20 fev. 2023.

BAUMEISTER, Giovanni. **Como Programar ESP32 com VS Code e PlatformIO**. 2020. Disponível em: https://www.makerhero.com/blog/como-programar-esp32-com-vs-code-e-platformio/. Acesso em: 28 mar. 2023.

CRUZ, Guilherme. **Séries temporais e o InfluxDB**. 2017. Disponível em: https://gcruz.com.br/blog/series-temporais-e-influxdb/. Acesso em: 23 mar. 2023.

ESPRESSIF SYSTEMS. **ESP32 Datasheet**. 2023. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 25 jan. 2023

FACCIONI FILHO, Mauro. **Internet das Coisas**. 2016. Disponível em: https://www.researchgate.net/profile/Mauro-Fazion-Filho/publication/319881659_Internet_das_Coisas_Internet_of_Things/links/59c038d5458515e9cfd54ff9/Internet-das-Coisas-Internet-of-Things.pdf. Acesso em: 04 jan. 2023.

GUSE, Rosana. **O que é Raspberry Pi?** 2020. Disponível em: https://www.makerhero.com/blog/o-que-e-raspberry-pi/. Acesso em: 26 fev. 2023.

HANASHIRO, Akira. **VS Code - O que é e por que você deve usar?** 2021. Disponível em: https://www.treinaweb.com.br/blog/vs-code-o-que-e-e-por-que-voce-deve-usar. Acesso em: 28 mar. 2023.

IBRAHIM, Dogan. **The Complete ESP32 Projects Guide**: 59 experiments with arduino ide and python. 59 Experiments with Arduino IDE and Python. 2019. Disponível em: https://www.elektor.com/the-complete-esp32-projects-guide. Acesso em: 28 jan. 2023.

KHANH, Do Bao. **DATA VISUALIZATION WITH INTEGRATION OF GRAFANA AND QUANTA PLATFORM**. 2021. Disponível em: https://www.theseus.fi/bitstream/handle/10024/497937/e1700699_DBK_Thesis_Final.pdf?sequence=2&isAllowed=y. Acesso em: 23 mar. 2023.

LEA, Rodger. **Node-RED: Lecture 1**: a brief introduction to node-red. A brief introduction to Node-RED. 2016. Disponível em: http://noderedguide.com/nr-lecture-1/. Acesso em: 21 mar. 2023.

LEPPÄNEN, Tiia. **Data visualization and monitoring with Grafana and Prometheus**. 2021. Disponível em: https://www.theseus.fi/bitstream/handle/10024/512860/Turkuamk_Bachelors_Thesis_Leppanen_Tiia.pdf?sequence=2&isAllowed=y. Acesso em: 23 mar. 2023.

LOPEZ RESEARCH LLC (San Francisco). **Uma introdução à Internet das Coisas (IoT)**. 2013. Disponível em: https://www.cisco.com/c/dam/global/pt_br/assets/brand/iot/iot/pdfs/lopez_research_an_introduction_to_iot_102413_final_portuguese.pdf. Acesso em: 27/11/2022.

LOUSADA, Ricardo. **Guia Completo do Sensor DS18B20 a Prova D'água**. 2020. Disponível em: https://blog.eletrogate.com/guia-completo-sobre-sensor-de-temperatura-ds18b20-a-prova-dagua/. Acesso em: 16 fev. 2023.

MAGRANI, Eduardo. **A internet das coisas**. Rio de Janeiro: Fgv Editora, 2018. 192 p. Disponível em: https://bibliotecadigital.fgv.br/dspace/bitstream/handle/10438/23898/A%20internet%20das%20coisas.pdf. Acesso em: 30 dez. 2022.

MAHLER., Charles. **How to Work with Time Series Data Using InfluxDB, Telegraf, and AWS**. 2021. Disponível em: https://www.odbms.org/2021/12/how-to-work-with-time-series-data-using-influxdb-telegraf-and-aws/. Acesso em: 23 mar. 2023.

MELLO, Marcio. **O que é um Microcontrolador, para que serve e principais usos**. 2022a. Disponível em: https://victorvision.com.br/blog/o-que-e-um-microcontolador/. Acesso em: 24 jan. 2023.

MELLO, Marcio. **O que é Raspberry Pi, para que serve e principais modelos no Brasil**. 2022b. Disponível em: https://victorvision.com.br/blog/o-que-e-raspberry-pi/. Acesso em: 26 fev. 2023.

MISCHIANTI, Renzo. **DOIT ESP32 DEV KIT v1 high resolution pinout and specs**. 2021. Disponível em: https://www.mischianti.org/2021/02/17/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/. Acesso em: 06 jun. 2023.

MOHANAN, Vishnu. **DOIT ESP32 DevKit V1 Wi-Fi Development Board**: pinout diagram & arduino reference. Pinout Diagram & Arduino Reference. 2022. Disponível em: https://www.circuitstate.com/wp-content/cache/all/pinouts/doit-esp32-devkit-v1-wifi-development-board-pinout-diagram-and-reference/index.html. Acesso em: 29 mar. 2023.

NATH, Omkar. REVIEW ON RASPBERRY Pi 3b+ AND ITS SCOPE. **International Journal Of Engineering Applied Sciences And Technology.** Madhya Pradesh, p. 157-159. jan. 2020. Disponível em: https://www.ijeast.com/papers/157-159,Tesma409,IJEAST.pdf. Acesso em: 31 mar. 2023.

OLIVEIRA, Euler. **Conhecendo o NodeMCU-32S ESP32**. 2017. Disponível em: https://blogmasterwalkershop.com.br/embarcados/esp32/conhecendo-o-nodemcu-32s-esp32. Acesso em: 07 dez. 2022.

OLIVEIRA, Jailson. **ESP32 – Especificação Técnica**. 2018. Disponível em: https://xprojetos.net/esp32-especificacao-tecnica/. Acesso em: 26 jan. 2023.

OLIVEIRA, Euler. Como usar com Arduino – Sensor de Temperatura DS18B20 Prova D'água do Tipo Sonda. 2018. Disponível em: https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-de-temperatura-ds18b20-prova-dagua-do-tipo-sonda. Acesso em: 16 fev. 2023.

PEIXE BR (São Paulo). **Anuário Brasileiro da Piscicultura PEIXE BR 2022**. 2022. Disponível em: https://www.peixebr.com.br/anuario2022/. Acesso em: 06 dez. 2022.

SANTOS, Bruno P.. **Internet das Coisas: da Teoria à Prática.** 2016. Disponível em: https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf. Acesso em: 30 dez. 2022.

SANTOS, Osmar R. dos. **Internet das Coisas na Indústria**: 5 exemplos de aplicação. 5 exemplos de aplicação. 2020. Disponível em: https://i3csolucoes.com.br/internet-das-coisas-na-industria-5-exemplos-de-aplicacao/. Acesso em: 30 out. 2023.

TAVARES, Camila P.s.. **Sistemas de alimentação inteligentes na piscicultura**. 2021. Disponível em: https://gia.org.br/portal/sistemas-de-alimentacao-inteligentes-na-piscicultura/#:~:text=O%20m%C3%A9todo%20%C3%A9%20realizado%20usando,determinar%20quando%20parar%20de%20alimentar.. Acesso em: 30 out. 2023.

TOLEDO, Victor. **FreeStyle Libre: sensor de glicemia ajuda a controlar a diabetes**. 2019. Disponível em: https://www.techtudo.com.br/noticias/2019/12/freestyle-libre-conheca-o-sensor-de-glicemia-que-ajuda-a-controlar-a-diabetes.ghtml. Acesso em: 30 out. 2023.

VERNIER. **Vernier Optical DO Probe**. 2016. Disponível em: https://www.vernier.com/files/manuals/odo-bta/odo-bta.pdf. Acesso em: 01 fev. 2023.

## Apêndice A – Script de monitoramento da temperatura

```
//importação das bibliotecas
#include <Arduino.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <DallasTemperature.h>
#include <DNSServer.h>
#include <ESPmDNS.h>
#include "RemoteDebug.h"
#include <ESP32AnalogRead.h>
#include <esp_bt.h>
#include <esp_sleep.h>
#include <esp32/sha.h>

//definição de variáveis

#define USE_MDNS true
#define SSID "usuarioWifi"
#define PASS "senhaWifi"
#define HOST_NAME "TH2Onode"
#define MQTT_ID HOST_NAME
#define MQTT_USER "mqtt_username"
#define MQTT_PASS "mqtt_password"
#define MQTT_BROKER "ip.do.raspberry"
#define MQTT_PORT 1883
#define TOPIC_SET_PERIOD "TH2Onode/period"
#define TOPIC_H2O_TEMP "TH2Onode/th2o"
#define TOPIC_IP "TH2Onode/ip"
#define NSAMPLES 100
#define WIFI_CONNECT_TIMEOUT 120
#define MQTT_CONNECT_TIMEOUT 60
#define TEMP_OFFSET_DEG 0
#define TH2O_PIN 15

//variáveis estáticas
WiFiClient espClient;
PubSubClient MQTT(espClient);
OneWire oneWire(TH2O_PIN);
DallasTemperature sensors(&oneWire);
unsigned long int sleep_period;
RemoteDebug Debug;

//funções
void initSerial(void)
{
  Serial.begin(115200);
  Serial.println("Comunicação Serial: Sucesso!");
}
```

```
void initWiFi(void)
{
  delay(10);
  Serial.println("Iniciando Wifi");
  reconnectWiFi();
  Serial.println("Sucesso!");
}

void reconnectWiFi(void)
{
  String myip;
  int timeout_counter = 0;
  if (WiFi.status() == WL_CONNECTED)
    return;
  WiFi.mode(WIFI_STA);
  WiFi.begin(SSID, PASS);
  Serial.println("Conectando ao WiFi ..");
  while ((WiFi.status() != WL_CONNECTED) & (timeout_counter <
WIFI_CONNECT_TIMEOUT))
  {
    //delay(200);
    Serial.print(".");
    timeout_counter++;
    delay(1000);
  }
  if (timeout_counter >= WIFI_CONNECT_TIMEOUT)
  {
    Serial.println("Tempo limite expirado! Reiniciando");
    ESP.restart();
  }

  Serial.println("Sucesso!");
  Serial.print("SSID: ");
  Serial.println(SSID);
  Serial.print(" IP: ");
  Serial.println( WiFi.localIP());
  myip = WiFi.localIP().toString();
  if (myip.equals("0.0.0.0"))
  {
    Serial.println("Erro! Reiniciando");
    ESP.restart();
  }

  if (!MDNS.begin(HOST_NAME))
  {
    Serial.println("Erro ao iniciar mDNS");
    return;
  }
```

```
  else
  {
    MDNS.addService("telnet", "tcp", 23);
  }
}

void initRemoteDebug(void)
  Debug.begin(HOST_NAME);
  Debug.setResetCmdEnabled(true);
  Debug.showProfiler(false);
  Debug.showColors(true);
  Debug.setSerialEnabled(true);
}

void disableBluetooth()
{
    btStop();
    esp_bt_controller_disable();
    delay(1000);
    Serial2.println("BT STOP");
}

void initMQTT(void)
{
  Serial.print("Iniciando MQTT");
  debugD("Iniciando MQTT");
  MQTT.setServer(MQTT_BROKER, MQTT_PORT);
  MQTT.setCallback(mqtt_callback);
}

void reconnectMQTT(void)
{
  int timeout_counter = 0;
  while (!MQTT.connected() & (timeout_counter < MQTT_CONNECT_TIMEOUT))
  {
    Serial.print("Conectando ao broker MQTT: ");
    Serial.print(MQTT_BROKER);
    debugD("Conectando ao broker MQTT: ");
    debugD(MQTT_BROKER);
    if (MQTT.connect(MQTT_ID, MQTT_USER, MQTT_PASS))
    {
      Serial.println(" Sucesso!");
      debugD(" Sucesso!");
      MQTT.subscribe(TOPIC_SET_PERIOD);
    }
    else
    {
      Serial.println(" ERRO!");
      debugD(" ERRO! Tempo limite expirado: %d", timeout_counter);
```

```
      timeout_counter++;
      delay(2000);
    }
    if (timeout_counter >= MQTT_CONNECT_TIMEOUT)
    {
      Serial.print("Tempo limite expirado! Reiniciando");
      debugD("Tempo limite expirado! Reiniciando");
      ESP.restart();
    }
  }
}

void mqtt_callback(char *topic, byte *payload, unsigned int length)
{
  String msg;
  for (int i = 0; i < length; i++)
  {
    char c = (char)payload[i];
    msg += c;
  }
  Serial.print("Msg MQTT recebida: ");
  Serial.println(msg);
  debugD("Msg MQTT recebida: %s", msg);
  if (strcmp(topic, TOPIC_SET_PERIOD) == 0)
  {
    sleep_period = 1000 * msg.toInt();
  }
}

void initSensors(void)
{
  debugD("Iniciando os sensores");
  sensors.begin();
  debugD("Sucesso!");
}

void verifyConns(void)
{
  if (!MQTT.connected())
    reconnectMQTT();
  reconnectWiFi();
}

void setup()
{
  char period_str[15] = {0};

  initSerial();
```

```
  initWiFi();

  initRemoteDebug();

  disableBluetooth();

  initMQTT();

  initSensors();

  sleep_period = 5000;
  sprintf(period_str, "%d", (int) sleep_period / 1000);
  MQTT.publish(TOPIC_SET_PERIOD, period_str);
}

void loop()
{
  float h2o_temp;
  char h2o_temp_str[10] = {0};
  float vbat;
  char vbat_str[10] = {0};

  verifyConns();

  Serial.print("lendo a temperatura da água:");

  sensors.requestTemperatures();
  h2o_temp = sensors.getTempCByIndex(0);
  h2o_temp += TEMP_OFFSET_DEG;
  sprintf(h2o_temp_str, "%.2f", h2o_temp);
  debugD("H2O temp: %s", h2o_temp_str);
  Serial.print("h2oTemp_str: ");
  Serial.println(h2o_temp_str);
  Serial.print("h2oTemp: ");
  Serial.println(h2o_temp);

  MQTT.publish(TOPIC_H2O_TEMP, h2o_temp_str);

  MQTT.publish(TOPIC_IP, WiFi.localIP().toString().c_str());

  MQTT.loop();

  Debug.handle();


  delay(sleep_period);
}
```

## Apêndice B – Script de monitoramento do oxigênio dissolvido

```
//importação das bibliotecas
#include <Arduino.h>
#include <ESP32AnalogRead.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ESPmDNS.h>
#include "RemoteDebug.h"
#include <esp_bt.h>
#include <esp_sleep.h>

//definição das variáveis
#define USE_MDNS true
#define SSID "usuarioWifi"
#define PASS "senhaWifi"
#define HOST_NAME "ODOBTAnode"
#define MQTT_ID HOST_NAME
#define MQTT_USER "mqtt_username"
#define MQTT_PASS "mqtt_password"
#define MQTT_BROKER "ip.do.raspberry"
#define MQTT_PORT 1883
#define TOPIC_SET_PERIOD "ODOBTAnode/period"
#define TOPIC_H2O_OD "ODOBTAnode/od"
#define TOPIC_VREAD "ODOBTAnode/vread"
#define TOPIC_IP "ODOBTAnode/ip"
#define NSAMPLES 100
#define WIFI_CONNECT_TIMEOUT 120
#define MQTT_CONNECT_TIMEOUT 60
#define TEMP_OFFSET_DEG 0
#define ODO_BTA_PIN 33

//variáveis estáticas
WiFiClient espClient;
PubSubClient MQTT(espClient);
unsigned long int sleep_period_ms;
RemoteDebug Debug;
ESP32AnalogRead adc;

//funções
void initSerial(void)
{
  Serial.begin(115200);
  Serial.println("Comunicação Serial: Sucesso!");
}

void initWiFi(void)
{
```

```
  delay(10);
  Serial.println("Iniciando WiFi ...");
  reconnectWiFi();
  Serial.println("Sucesso!");
}

void reconnectWiFi(void)
{
  String myip;
  int timeout_counter = 0;
  if (WiFi.status() == WL_CONNECTED)
    return;
  WiFi.begin(SSID, PASS);
  while ((WiFi.status() != WL_CONNECTED) & (timeout_counter <
WIFI_CONNECT_TIMEOUT))
  {
    delay(100);
    Serial.print(".");
    timeout_counter++;
  }
  if (timeout_counter >= WIFI_CONNECT_TIMEOUT)
  {
    Serial.println("Tempo Limite Expirado! Reiniciando");
    ESP.restart();
  }

  Serial.println("Sucesso!");
  Serial.print("SSID: ");
  Serial.println(SSID);
  Serial.print(" IP: ");
  Serial.println( WiFi.localIP());
  myip = WiFi.localIP().toString();
  if (myip.equals("0.0.0.0"))
  {
    Serial.println("Erro! Reiniciando");
    ESP.restart();
  }

  if (!MDNS.begin(HOST_NAME))
  {
    Serial.println("Erro ao iniciar mDNS");
    return;
  }
  else
  {
    MDNS.addService("telnet", "tcp", 23);
  }
}
void initRemoteDebug(void)
```

```
{
  Debug.begin(HOST_NAME);
  Debug.setResetCmdEnabled(true);
  Debug.showProfiler(false);
  Debug.showColors(true);
  Debug.setSerialEnabled(true);
}

void disableBluetooth()
{
    btStop();
    esp_bt_controller_disable();
    delay(1000);
    Serial2.println("BT STOP");
}

void initMQTT(void)
{
  debugD("Iniciando MQTT ...");
  MQTT.setServer(MQTT_BROKER, MQTT_PORT);
  MQTT.setCallback(mqtt_callback);
}

void reconnectMQTT(void)
{
  int timeout_counter = 0;
  while (!MQTT.connected() & (timeout_counter < MQTT_CONNECT_TIMEOUT))
  {
    debugD("Conectando ao broker MQTT: ");
    debugD(MQTT_BROKER);
    if (MQTT.connect(MQTT_ID, MQTT_USER, MQTT_PASS))
    {
      debugD(" Sucesso!");
      MQTT.subscribe(TOPIC_SET_PERIOD);
    }
    else
    {
      debugD(" ERRO! Tempo Limite Expirado: %d", timeout_counter);
      timeout_counter++;
      delay(2000);
    }
    if (timeout_counter >= MQTT_CONNECT_TIMEOUT)
    {
      debugD("Tempo Limite Expirado! Reiniciando");
      ESP.restart();
    }
  }
}
void mqtt_callback(char *topic, byte *payload, unsigned int length)
```

```
{
  String msg;
  for (int i = 0; i < length; i++)
  {
    char c = (char)payload[i];
    msg += c;
  }
  debugD("Msg MQTT recebida: %s", msg);
  if (strcmp(topic, TOPIC_SET_PERIOD) == 0)
  {
    sleep_period_ms = 1000 * msg.toInt();
  }
}

float read_sensor_volt(int nsamples)
{
  float vread = 0;

  for (int i = 0; i < nsamples; i++)
  {
    vread += adc.readVoltage();
  }
  vread = vread / nsamples;
  return (vread);
}

void initSensors(void)
{
  debugD("Iniciando os sensores ...");
  adc.attach(ODO_BTA_PIN);
  debugD("Sucesso!");
}

void verifyConns(void)
{
  if (!MQTT.connected())
    reconnectMQTT();
  reconnectWiFi();
}

void setup() {
  char period_str[15] = {0};

  initSerial();

  initWiFi();

  initRemoteDebug();
```

```
  disableBluetooth();

  initMQTT();

  initSensors();

  sleep_period_ms = 5000;
  sprintf(period_str, "%d", (int) sleep_period_ms / 1000);
  MQTT.publish(TOPIC_SET_PERIOD, period_str);
}

void loop()
{
  float vread;
  float h2o_od;
  char h2o_od_str[10] = {0};
  char vread_str[10] = {0};

  verifyConns();

  vread = read_sensor_volt(NSAMPLES);
  sprintf(vread_str, "%1.2f V", vread);
  MQTT.publish(TOPIC_VREAD, vread_str);
  debugD("Vread: %s", vread_str);

  h2o_od = vread*13.2/3.3;
  sprintf(h2o_od_str, "%2.2f", h2o_od);
  MQTT.publish(TOPIC_H2O_OD, h2o_od_str);
  debugD("h2o_od: %s", h2o_od_str);

  MQTT.publish(TOPIC_IP, WiFi.localIP().toString().c_str());

  MQTT.loop();

  Debug.handle();
  delay(sleep_period_ms);

}
```