Universidade Federal de Goiás – UFG
Escola de Engenharia Elétrica, Mecânica e de Computação – EMC
Graduação em Engenharia de Computação

Aplicação IoT para medição e acompanhamento em tempo real do pH no rúmen de bovinos

Danilo Rodrigues Torchio

Brasil
2023

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

DECLARAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC nº 1204/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG):

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): Danilo Rodrigues Torchio

Título do trabalho: Aplicação IoT para medição e acompanhamento em tempo real do pH no rúmen de bovinos

2. Informações de acesso ao documento:

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

Independente da concordância com a disponibilização eletrônica, é imprescindível o envio do(s) arquivo(s) em formato digital PDF do TCCG.

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;

- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor

Documento assinado eletronicamente por **José Wilson Lima Nerys**, **Professor do Magistério Superior**, em 27/02/2023, às 11:53, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Danilo Rodrigues Torchio**, **Discente**, em 27/02/2023, às 11:56, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3552326** e o código CRC **A298182C**.

Referência: Processo nº 23070.060073/2022-46 SEI nº 3552326

Danilo Rodrigues Torchio

Aplicação IoT para medição e acompanhamento em tempo real do pH no rúmen de bovinos

Trabalho de Conclusão de Curso apresentado a Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás - EMC/UFG como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Dr. José Wilson Lima Nerys

Brasil

2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Torchio, Danilo Rodrigues
Aplicação IoT para medição e acompanhamento em tempo real do pH no rúmen de bovinos [manuscrito] / Danilo Rodrigues Torchio. - 2023.
LXIV, 64 f.: il.

Orientador: Prof. Dr. José Wilson Lima Nerys.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), , Goiânia, 2023.
Bibliografia.
Inclui lista de figuras, lista de tabelas.

1. Sensor de pH. 2. Sensoriamento. 3. IoT. 4. ESP32. 5. LoRa. I. Nerys, José Wilson Lima, orient. II. Título.

CDU 62+004+005

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

DECLARAÇÃO

ATA DE AVALIAÇÃO DE PROJETO FINAL

Curso

( ) Eng Elétrica	( ) Eng Mecânica	(x) Eng Computação PFC 1 ( ) PFC 2 (x)

Título do Trabalho

Aplicação IoT para medição e acompanhamento em tempo real do pH no rúmen de bovinos

Banca Avaliadora

Membro 1	Prof. José Wilson Lima Nerys
Membro 2	Prof. Álisson Assis Cardoso
Membro 3	TAE. Gustavo Souto de Sá e Souza

Discente

Matrícula	Nome
201400371	Danilo Rodrigues Torchio

NOTAS										
	Membro 1			Membro 2			Membro 3			
Matrícula	NPT	NTE	NAA	NPT	NTE	NAA	NPT	NTE	NAA	Média*
201400371	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0	10,0

NPT – Nota plano de trabalho;
NTE – Nota do trabalho escrito;
NAA – Nota de apresentação e arguição
Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA
Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

* A APROVAÇÃO DO(S) ALUNO(S) ESTÁ CONDICIONADA À APRESENTAÇÃO DO TRABALHO FINAL AO ORIENTADOR COM TODAS AS CORREÇÕES SUGERIDAS PELA BANCA.

OBSERVAÇÕES:

Preencher com modificações solicitadas, caso existam. Em caso de reprovação, informar a justificativa.

_Houve mudança do Título do trabalho, sugerida pelo próprio aluno, e aprovada pela banca._____

_Foram sugeridas pequenas alterações no texto_____

_____

_____

_____

Documento assinado eletronicamente por **Gustavo Souto De Sá E Souza**, **Técnico de Laboratório**, em 23/02/2023, às 10:35, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **José Wilson Lima Nerys**, **Professor do Magistério Superior**, em 23/02/2023, às 10:35, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 23/02/2023, às 10:35, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3543430** e o código CRC **5A70ADAA**.

Referência: Processo nº 23070.060073/2022-46 SEI nº 3543430

A minha esposa, familiares e amigos que me apoiaram durante todo o meu processo de formação.

Agradecimentos

Primeiramente agradeço a Deus por me dar saúde e condições para cursar uma Graduação e superar todos os obstáculos.

Agradeço a minha esposa, por me suportar e apoiar nos períodos de estresse e sempre me incentivar a seguir em frente.

À meu professor e orientador, Dr. José Wilson Lima Nerys, por me dar a oportunidade de desenvolver este trabalho e pela orientação por todo o caminho.

"Cada um de nós é, sob uma perspectiva cósmica, precioso.
Se um humano discorda de você, deixe-o viver. Em cem bilhões
de galáxias, você não irá achar outro como ele."
Carl Sagan

Resumo

O controle do nível de pH no rúmen de bovinos é uma importante variável em projetos de nutrição bovina. Estudos de melhoramento genético das pastagens e avanços na suplementação alimentar (aditivos introduzidos na ração) agregam maior produtividade e qualidade da carne produzida. Pesquisadores do Instituto de Ciências Biológicas (ICB-UFG) da Universidade Federal de Goiás (UFG) realizam coletas periódicas do líquido do rúmen para medição do pH em laboratório. Os pesquisadores utilizaram, por um período, um dispositivo para medição do pH dentro do rúmen e armazenamento em cartão SD. Esse projeto foi resultado de um TCC da Escola de Engenharia Elétrica, Mecânica e de Computação (EMC) da UFG. A proposta atual é uma melhoria do projeto anterior nos seguintes aspectos: leitura do pH e envio para uma base remota, conectada à internet. O resultado é a possibilidade de acompanhamento em tempo real dos valores de pH e o aumento da autonomia da bateria.

Palavras-chave: Sensoriamento, Sensor de pH, ESP32, IoT.

Abstract

Controlling the pH level in the bovine rumen is an important variable in bovine nutrition projects. Studies of genetic improvement of pastures and advances in food supplementation (additives introduced in the feed) aggregate greater productivity and quality of the meat produced. Researchers from the Institute of Biological Sciences (ICB-UFG) at the Federal University of Goiás (UFG) carry out periodic collections of rumen fluid to measure the pH in the laboratory. The researchers used, for a period, a device to measure the pH inside the rumen and store it on an SD card. This project was the result of a TCC at the School of Electrical, Mechanical and Computer Engineering (EMC) at UFG. The current proposal is an improvement on the previous project in the following aspects: pH reading and sending to a remote base connected to the internet. The result is the possibility of real-time monitoring of pH values and increased battery life.

Keywords: Sensing, pH sensor, ESP32, IoT.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADC	*Analog-Digital Converter*
API	*Application Programming Interface*
BLE	*Bluetooth Low Energy*
CPU	*Central Processing Unit*
EMC	Escola de Engenharia Elétrica, Mecânica e de Computação
GPIO	*General-Purpose Input/Output*
GHz	Gigahertz
I2C	*Inter-Integrated Circuit*
ICB	Instituto de Ciências Biológicas
IoT	*Internet of Things*
LoRa	*Long Range*
MCU	*Microcontroller Unit*
MHz	Megahertz
pH	Potencial Hidrogeniônico
PWM	*Pulse Width Modulation*
RAM	*Random-Access Memory*
RF	Radiofrequência
ROM	*Read Only Memory*
RTC	*Real Time Clock*
SD	*Secure Digital*
SPI	*Serial Peripheral Interface*
TSMC	*Taiwan Semiconductor Manufacturing Company*
UART	*Universal Asynchronous Receiver / Transmitter*
UFG	Universidade Federal de Goiás
Wi-Fi	*Wireless Fidelity*

Sumário

1 INTRODUÇÃO

Em projetos de nutrição bovina, o controle do pH intra-ruminal é uma variável importante na pesquisa e desenvolvimento de aditivos que serão colocados na ração do animal, cujo objetivo é aumentar e melhorar a qualidade da carne produzida. Compreender os fatores que influenciam o pH é essencial para projetar e otimizar processos químicos.

Pesquisadores do Instituto de Ciências Biológicas (ICB) da Universidade Federal de Goiás (UFG) utilizam uma sonda, inserida no rúmen do animal (via cânula), para medir e acompanhar os níveis de pH durante as pesquisas. Esta sonda realiza a medição do pH, em um intervalo programável, e armazena as leituras em um cartão de memória (SD). Os pesquisadores precisam, então, remover periodicamente a sonda para coleta dos dados e recarregar a bateria do dispositivo. Este procedimento invasivo é realizado pelo menos uma vez ao dia, durante todo o período da pesquisa.

Este trabalho apresenta uma proposta e desenvolvimento de uma melhoria neste dispositivo, utilizando técnicas modernas de IoT, a fim de eliminar a necessidade da remoção da sonda para coleta dos dados (utilizando o meio sem fio), aumentar a capacidade da bateria (utilizando componentes de baixa potência) e prover uma interface para visualização em tempo real das medidas realizadas. Além de agilizar o processo de análise de dados, esta melhoria também visa diminuir o estresse do animal, que poderá, ainda, ter um efeito sobre as pesquisas.

Os capítulos deste trabalho foram organizados de forma cronológica, seguindo a ordem da pesquisa e produção dos protótipos.

1.1 Objetivos Gerais

Este projeto tem como objetivo realizar a leitura e acompanhamento em tempo real do pH no rúmen de bovinos através da utilização de IoT.

1.2 Objetivos específicos

- Apresentar um conceito básico de pH e o método de medição com eletrodo eletrônico;
- Apresentar os conceitos e características do microcontrolador ESP32;
- Apresentar os conceitos dos meios de transmissão sem fio WiFi, Bluetooth e LoRa;

- Apresentar os conceitos e tecnologias utilizadas no desenvolvimento de sistemas para web;
- Construção de um protótipo em protoboard da sonda e estação de recebimento;
- Desenvolvimento do sistema embarcado da sonda de pH e da estação de recebimento;
- Desenvolvimento de uma plataforma web para acompanhamento em tempo real das medidas de pH.

Parte I

REFERENCIAL TEÓRICO

2 Conceito de pH

O potencial hidrogeniônico, ou pH, é uma escala desenvolvida pelo dinamarquês Soren P. L. Sorensen, no ano de 1909 (NOVAIS, 2023) . É uma medida de acidez ou basicidade de uma solução e é definida como o logaritmo negativo da concentração de cátions hidrônio ($H+$ ou H_3O+) e íons hidróxido (OH^-) presentes no meio e indica se esse meio, ou mistura, é ácido, básico ou neutro (DIAS, 2023).

A faixa de variação do pH vai de 0 a 14, sendo 0 uma solução totalmente ácida e 14 uma solução totalmente básica. Podemos classificar uma solução por meio do valor do pH da seguinte maneira:

- **Solução neutra**, pH = 7;
- **Solução ácida**, pH < 7;
- **Solução básica**, pH > 7.

Para medir o valor do pH de uma solução utiliza-se o princípio da **potenciometria**, ou **titulação potenciométrica**. Este é um método de medida de potencial entre dois eletrodos (referência e indicador). Essa diferença de potencial (d.d.p) é função de pH (SABADIN, 2003). O químico alemão Hermann Walther Nernst (1864-1941) definiu essa função como

$$E = E_0 + 2,3\frac{RT}{F} \cdot log([H^+]) \tag{2.1}$$

Onde:

- $E0$ = valor do potencial padrão / $[H^+] = 1mol/l$;
- F = Constante de Faraday = $96.485,3sA/mol$;
- R = Constante Universal dos Gases = $8.3145J.K^-1 \cdot mol^-1$;
- T = Temperatura Absoluta, em graus Kelvin;
- $-log[H^+]$ = pH da solução.

Perceba que a temperatura tem uma influência forte na medição de pH. Um bom equipamento de medição deverá levar em consideração a temperatura da solução para a correta medição.

Neste trabalho não iremos nos aprofundar na teoria aqui apresentada pois não é nosso objetivo a construção de um novo sensor. A Figura 1 mostra um exemplo de sensor de pH com eletrodo de vidro.

Figura 1 – Sensor de pH com eletrodo de vidro

Fonte: Google[1]

[1] Disponível em: <https://tinyurl.com/2r6fy92c>. Acessado em 22 fev 2023.

3 IoT

IoT é uma sigla que significa "Internet das Coisas"(*Internet of Things*). É um termo usado para descrever uma rede de dispositivos físicos, conectados à internet, que vão desde dispositivos simples como relógios, câmeras e equipamentos de áudio até dispositivos complexos como televisões, máquinas de lavar roupas, carros e equipamentos industriais. Estes dispositivos podem se comunicar entre si e com outros dispositivos e sistemas.

Embora pareça que todo objeto capaz de se conectar à internet irá cair na categoria "Coisas", essa notação é usada para abranger um conjunto mais genérico de entidades, sendo qualquer objeto que esteja ciente de seu contexto e seja capaz de se comunicar com outras entidades, tornando acessível a qualquer hora, em qualquer lugar (BUYYA; DASTJERDI, 2016).

Alguns exemplos de dispositivos de IoT são:

- ***Smartwatches***: conectado ao smartphone, os smartwatches (Figura 2) agregam valor e estendem as funções do celular.
- **Carros inteligentes**: veículos autônomos já são uma realidade e só existem graças às aplicações de IoT. Centenas de sensores e atuadores conectados comunicando-se entre si para entregar uma experiência única, como alerta de colisão e frenagem automática, manobras de estacionamento automáticas, entre outras.
- **Indústria 4.0**: sensores implantados nas máquinas e processos de produção industrial tornam a fábrica mais inteligente, otimizando a produção e predizendo falhas, a fim de tratá-las com maior agilidade.

Não é à toa que esses dispositivos estão se tornando cada vez mais populares. Com a IoT, nós não apenas podemos controlar equipamentos, como lâmpadas e motores, mas também se informar sobre seu estado atual, o que transmite uma maior segurança, como por exemplo, saber se uma porta ou janela está aberta ou o a pressão atual em um sensor de uma máquina industrial.

3.1 Microcontroladores

Segundo (MIYADAIRA, 2009): “Os microcontroladores são pequenos dispositivos dotados de "inteligência", constituídos de CPU, memória e periféricos.”. Ou seja, um microcontrolador é um dispositivo que contém, dentro do mesmo revestimento, um micro-

Figura 2 – Smartwatch Xiaomi Redmi 3

Fonte: Google[1]

processador (CPU) e outros dispositivos como memória ROM, memória RAM, conversores A/D e etc.

A alta demanda por sistemas embarcados é um importante fator no desenvolvimento e evolução deste tipo de componente eletrônico.

Os dispositivos de IoT dependem fortemente de microcontroladores e estes têm evoluído muito nos últimos anos. O ESP32 é um microcontrolador muito popular usado em aplicações de IoT devido seus poderosos recursos e conectividade Wi-Fi e Bluetooth integradas. Com o ESP32, os desenvolvedores podem criar dispositivos de baixo custo e com baixo consumo de energia, capazes de coletar e transmitir dados para outros dispositivos e para a internet.

Figura 3 – Microcontrolador PIC16F1826 produzido pela MICROCHIP

Fonte: Google[2]

[1] Disponível em: <https://tinyurl.com/3yuzmhnd>. Acessado em 18 fev 2023.
[2] Disponível em: <https://tinyurl.com/2rmw83en>. Acessado em 18 fev 2023.

3.1.1 ESP32

O ESP32 é um MCU (microcontrolador) desenvolvido pela Espressif (2023)[1], rico em recursos, sendo apropriado para uma grande gama de aplicações.

Em um único chip, o ESP32 conta com Wi-Fi e Bluetooth de 2,4 GHz integrados, fabricado com a tecnologia TSMC de baixa potência de 40nm. Ele foi projetado para obter o melhor desempenho de potência e RF, mostrando robustez, versatilidade e confiabilidade em uma ampla variedade de aplicações, tornando-o uma escolha certa para projetos de IoT (ESPRESSIF, 2023).

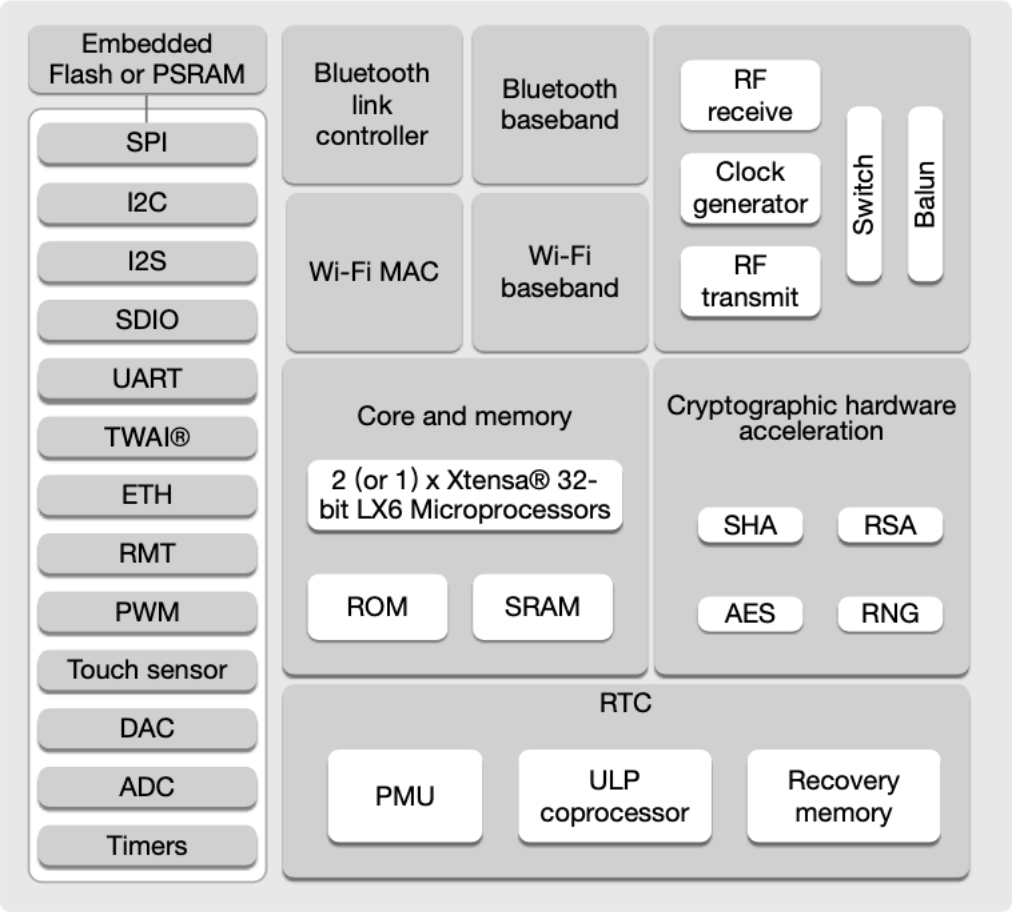
Principais especificações:

- Microprocessador Xtensa single-/dual-core 32-bit LX6;
- Wi-Fi 802.11 b/g/n 2.4 GHz
- Bluetooth v4.2 e BLE
- Frequência típica de 240 MHz
- 520 KB SRAM;
- 448 KB ROM;
- Memória flash embutida de 4 MB
- 2 conversores ADC de 12-bits com 18 canais
- 34 pinos programáveis (GPIO);
- 4 SPI;
- 3 UART;
- 2 I2C;
- 2 grupos de timers com 2 x 64-bit timers e 1 watchdog em cada grupo;
- 1 RTC timer;
- LED PWM com 16 canais;
- Processador de baixa potência ULP (Ultra-Low Power);
- Modo deep-sleep (consumo de $100\mu A$).

[1] *Espressif Systems* é uma empresa multinacional de semicondutores, estabelecida em 2008 na China e focada no desenvolvimento de soluções de comunicação sem fio e IoT de baixa potência.

No diagrama da Figura 4, conseguimos ver em detalhes a organização dos componentes do ESP32.

Figura 4 – Diagram de blocos funcional do ESP32



Fonte: Espressif

Como podemos observar, o ESP32 é um microcontrolador poderoso, recheado de recursos e uma boa capacidade computacional. Uma de suas principais características, e que o torna uma boa opção para aplicações IoT é o seu recurso de gerenciamento de energia. Você pode alternar entre os diferentes modos de acordo com a necessidade da aplicação, poupando energia quando não é necessário. Confira abaixo os 5 modos de gerenciamento de energia:

- **Ativo**: todos os recursos ativos;
- **Modem-sleep**: A CPU permanece ativa e os clocks configuráveis. Os rádios Wi-Fi e Bluetooth estão desativados;
- **Light-sleep**: A CPU fica pausada. A memória do RTC e seus periféricos, além do Ultra-Low Power co-processador continuam ativos. Qualquer evento de wake-up irá despertar o chip;

- **Deep-sleep**: Somente a memória e os periféricos do RTC estão ligados. Os dados de conexão do Wi-Fi e Bluetooth são guardados na memória. O co-processador ULP permanece funcional.
- **Hibernação**: O clock interno de 8 MHz e o co-processador ULP são desativados. A memória do RTC é desligada. Somente o timer RTC e algumas portas RTC GPIOs permanecem ativas. Somente o timer RTC ou as portas GPIO do RTC podem despertar o chip do modo de hibernação.

O recurso de gerenciamento de energia avançado que o ESP32 possui será uma valiosa propriedade para o projeto da Sonda de medição de pH, tendo em vista que um dos requisitos é o aumento da capacidade da bateria e o controlador pode permanecer no modo deep-sleep ou em hibernação entre uma leitura e outra.

A Tabela 1 mostra um comparativo do consumo de energia do chip nos seus diferentes modos de operação:

Tabela 1 – Consumo de energia vs modos de operação

Modo de Operação	**Consumo de Energia**
Ativo	$95 \sim 240mA$
Modem-sleep	$20 \sim 68mA$
Light-sleep	$800\mu A$
Deep-sleep	$10 \sim 150\mu A$
Hibernação	$5\mu A$

Fonte: Espressif (2023)

Mais detalhes sobre os modos de operação podem ser consultados no *datasheet* do componente na página do fabricante.

3.1.2 Devkits

Para rápida prototipagem, a fabricante disponibiliza placas de desenvolvimento baseadas no ESP32, com todos os recursos prontos para utilização. Estas placas já vem com todos os componentes devidamente soldados, além de todos os circuitos de proteção e integração necessários para utilização dos recursos que o controlador oferece.

A Figura 5 mostra um exemplo de placa de desenvolvimento disponibilizada pela própria fabricante.

Além dos DevKits disponibilizados pela própria fabricante, diversas empresas e terceiros também produzem essas placas de desenvolvimento com diversos recursos e módulos já integrados. Uma dessas empresas é a Heltec Automation, que produz uma placa de desenvolvimento baseada no ESP32 e também possui um display de OLED e conexão LoRa.

Figura 5 – ESP32 DevKitC

Fonte: Espressif (2023)

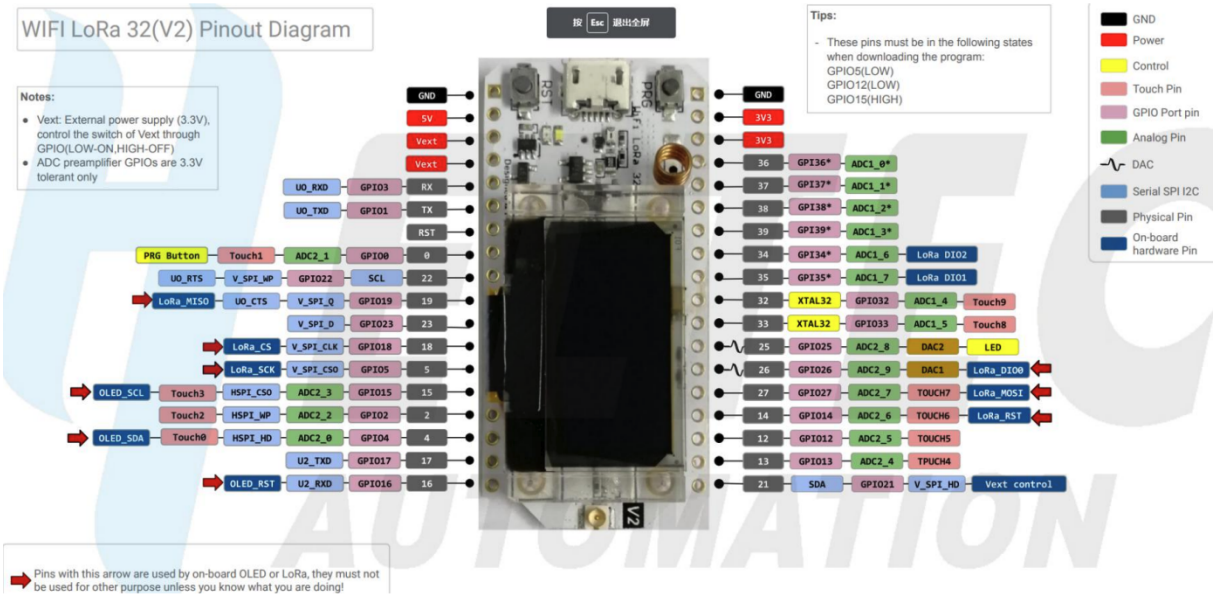
A placa de desenvolvimento da Heltec Automation, chamada Heltec WiFi LoRa 32 possui também as seguintes características:

- Microprocessador: ESP32 (dual-core 32-bit MCU);
- LoRa chip: SX1276/SX1278;
- Interface SH1.25-2: sistema integrado de gerenciamento de baterias de lithium;
- Display OLED: 128x64;
- USB CP2102: para download do programa e debugging;
- Suporta o ambiente de desenvolvimento do Arduino;
- SDKs de desenvolvimento: ESP32 + LoRaWAN protocolos.

Para este projeto, foi escolhida a placa de desenvolvimento da empresa Heltec Automation, pois a mesma já possui integração com o sistema de comunicação LoRa, facilitando a prototipagem do projeto.

A Figura 6 mostra a pinagem da placa de desenvolvimento WiFi LoRa 32 (v2).

Figura 6 – Pinagem do módulo WiFi LoRa 32 (v2)



Fonte: Heltec Automation (2023)

3.2 Sensores

Em aplicações IoT, diversos sensores e atuadores são usados para compor a solução desejada. Neste projeto, iremos utilizar dois módulos, sendo eles o PH4502C, que é um módulo de leitura do sensor de pH e o DS18B20 que é um sensor de temperatura à prova d 'água.

3.2.1 PH4502C

O módulo PH4502C (Figura 7), produzido pela empresa Diy More, é um sensor de detecção de valor de pH, desenvolvido para medir e monitorar os níveis de pH em soluções e com integração com plataformas de desenvolvimento como o Arduíno e o ESP32 DevKit.

Possui as seguintes características:

- Corrente de trabalho: $5 \sim 10mA$;
- Faixa de detecção de pH: $0 \sim 14$;
- Tempo de resposta: $<= 5s$;
- Tempo de estabilização <= 60s;
- Consumo de energia: $<= 5W$;
- Saída: tensão analógica do valor de pH lido.

Figura 7 – Módulo PH4502C da empresa Diy More

Fonte: Diy More (2023)[1]

3.2.2 DS18B20

O módulo DS18B20 (Figura 8) é um sensor de temperatura à prova d'água que fornece medições utilizando apenas um fio. É bastante preciso e proporciona leituras de temperatura de até 12-bits.

Especificações técnicas:

- Tensão de operação: $3 \sim 5,5V$;
- Faixa de medição: $-55 \sim 125°C$
- Precisão: $\pm 0,5C$ entre -10 e $85°C$;
- Ponta de aço inoxidável.

Figura 8 – Módulo DS18B20

Fonte: Google[2]

[1] Disponível em: <https://tinyurl.com/4uhptx45>. Acessado em 20 fev 2023.
[2] Disponível em: <https://tinyurl.com/yckwsua9>. Acessado em 20 fev 2023.

Parte II

DESENVOLVIMENTO

4 Hardware

Este capítulo descreve todos os componentes e placas eletrônicas utilizados na prototipação deste projeto.

4.1 Visão Geral

Este projeto foi dividido em três componentes distintos, sendo eles a parte física (hardware), que é composta pelos projetos da Sonda e da Estação, o aplicativo mobile, utilizado para fazer a configuração do hardware e o sistema web que é a interface de visualização em tempo real das medições realizadas.

O diagrama da Figura 9 mostra uma visão geral do projeto, com todos os componentes conectados compondo a solução geral.

A Tabela 2 mostra os componentes adiquiridos para prototipação da parte física deste projeto.

Tabela 2 – Orçamento dos componentes utilizados (2023)

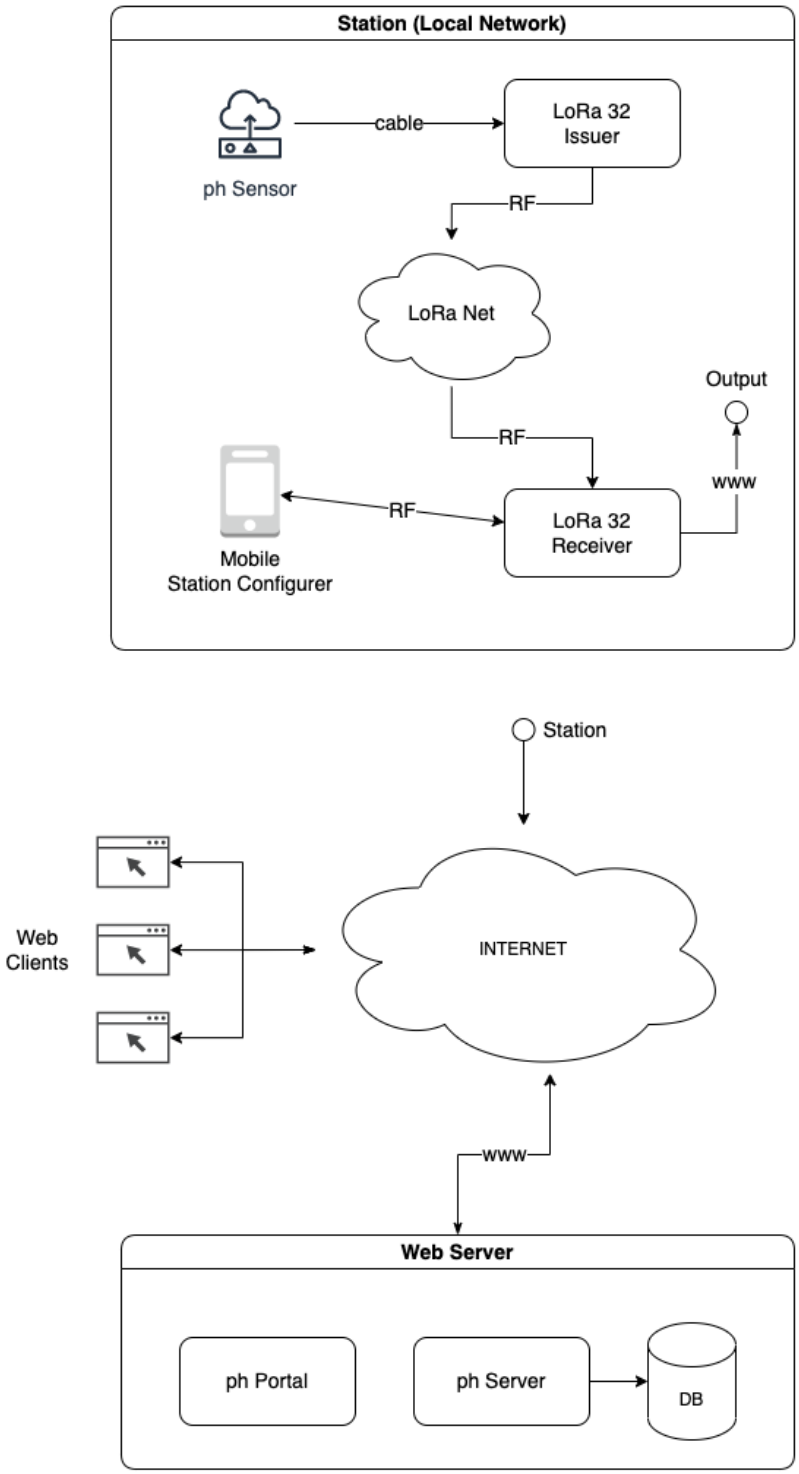
Componente	**Quantidade**	**Valor Unitário**	**Total**
Placa Heltec WiFi LoRa 32 (V2)	2	R$ 185,00	R$ 370,00
Sensor PH4502C com Eletrodo BNC	1	R$ 139,00	R$ 139,00
Sensor DS18B20 a prova d'água	1	R$ 21,95	R$ 21,95
Kit Cabo Jumper 40 fios Macho-Macho	1	R$ 16,99	R$ 16,99
Kit 10 Sachês Solução pH 4.01 - 6.86	1	R$ 39,00	R$ 39,00
Água Deionizada 1 Litro	1	R$ 19,90	R$ 19,90
Total			R$ 606,84

Fonte: Elaboração própria.

Conforme podemos observar, as placas de desenvolvimento de prototipação rápida e o sensor de pH com o eletrodo do tipo BNC são os componentes que encareceram o projeto. Isto é aceitável para prototipação mas é inviável para produção. Apesar da praticidade das placas com LoRa, uma pesquisa rápida pelos componentes individuais (microcontrolador, memórias e componentes passivos) e a confecção da PCB, mostram que os custos de produção poderiam ser reduzidos de forma considerável.

A seguir serão descritos em detalhes cada componente do projeto.

Figura 9 – Diagrama Geral



Fonte: Elaboração própria.

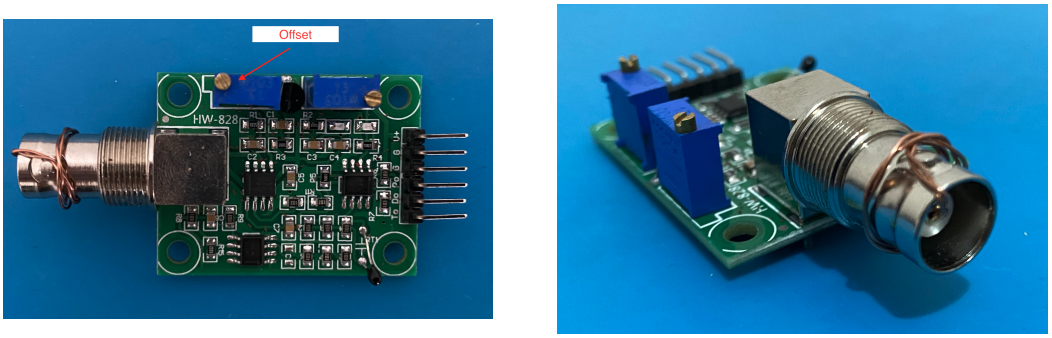
4.2 Sonda

A sonda é o componente responsável por fazer a leitura do valor de pH, que será enviado para a estação via comunicação LoRa. A leitura é realizada pelo sensor PH4502C. Este sensor é composto por duas partes, sendo o eletrodo de leitura, com conexão do tipo BNC, e a placa eletrônica responsável por fazer o condicionamento do sinal enviado pelo sensor e colocar na saída um sinal analógico, que será lido pelo microcontrolador.

Devido às características de construção do sensor, conforme apresentado no capítulo X, a diferença de potencial lida poderá ser um valor negativo. Por este motivo se faz necessário realizar o condicionamento do sinal antes do envio para o microcontrolador. Também será necessário fazer a calibração da sonda. O ponto médio deste sensor é 0V para pH 7, logo, devemos deslocar a saída para os níveis adequados.

O procedimento de calibração da sonda é bastante simples e realizado em duas partes. Na primeira parte devemos calibrar o ponto médio (pH 7) para a tensão de 2,5V. Esse procedimento é realizado de forma muito simples, bastando curto-circuitar os terminais do eletrodo, conforme ilustrado na Figura 10, simulando assim uma diferença de potencial igual a zero. Com os terminais curto-circuitados, ajustamos a tensão de saída do sensor variando o trimpot de offset até que a tensão chegue a 2,5V. Deste modo, a tensão de saída do sensor será uma tensão analógica de 0 à 5V, representando, respectivamente, os níveis de pH 14 e pH 0.

Figura 10 – Curto-circuito dos terminais do eletrodo da sonda de pH



Fonte: Elaboração própria.

Na sequência, devemos encontrar a equação da reta característica do dispositivo. Para isso, precisamos de duas soluções com pH conhecidos, para fazermos a leitura e anotarmos os pontos. Depois basta encontrarmos a equação da reta que passa pelos dois pontos.

Foram adquiridas duas soluções, com pH 4,01 e pH 6,82, para esta calibração e as tensões encontradas foram $2,14V$ e $1,8V$, respectivamente. Logo, a função para cálculo do pH lido pode ser definida na forma de $y = mx + b$, donde y será o valor do pH, m é o coeficiente angular, x é a tensão lida e b é o coeficiente linear (OLIVEIRA, 2023).

$$m = \frac{Y_2 - Y_1}{X_2 - X_1} = \frac{6,82 - 4,01}{1,8 - 2,14}$$
$$m = -8,29 \tag{4.1}$$

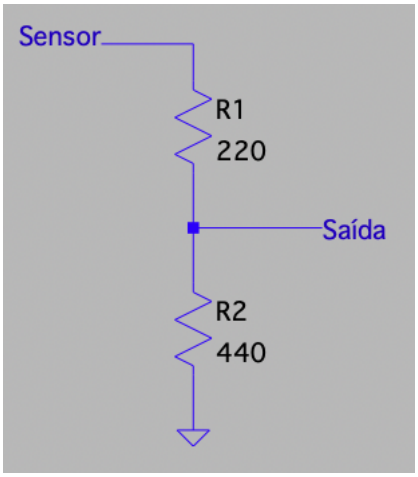
$$b = y - mx = 6,82 + 8,29 \cdot 1,8$$
$$b = 21,74 \tag{4.2}$$

$$y = -8,29x + 21,74 \tag{4.3}$$

O último ponto que devemos ficar atentos é aos níveis de tensão trabalhados pelo microcontrolador. No ESP32, os conversores A/D trabalham com tensões de 0 à 3,3V, e o sinal entregue pela sonda será uma tensão analógica entre 0 à 5V. Logo, será necessário fazer uma conversão para os níveis de tensão adequados ao ESP32.

Para isto foi implementado um simples divisor de tensão, como mostrado na Figura 11.

Figura 11 – Divisor de tensão



Fonte: Elaboração própria.

Note que, pelo princípio da divisão de tensão (ALEXANDER; SADIKU, 2013, pág. 39 e 40), a tensão na saída V_o será

$$V_o = \frac{R_2}{R_1 + R_2} \cdot V_i \tag{4.4}$$

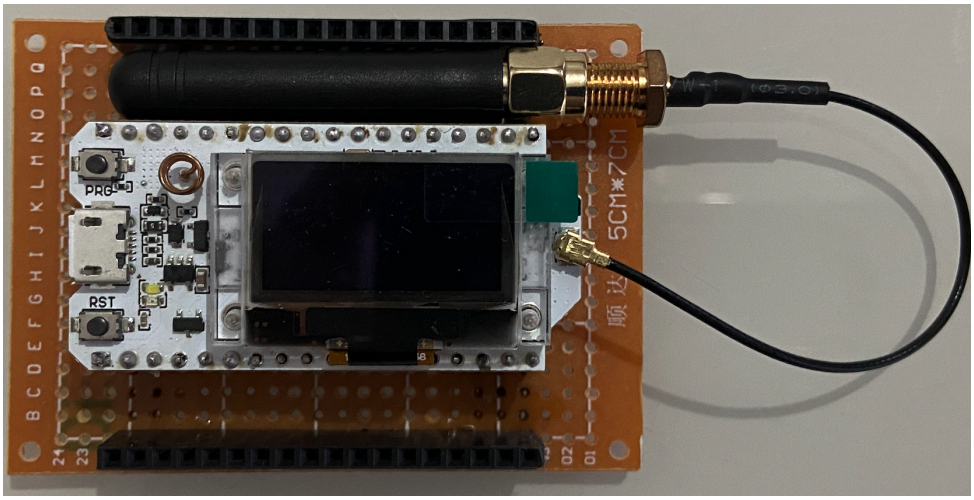
Logo, para uma tensão de $5V$ na saída do sensor teremos uma tensão de $3,3V$ na entrada do microcontrolador.

4.3 Estação

A estação é o componente responsável por recepcionar a leitura encaminhada pela sonda e publicar a informação no servidor web.

Para a estação, o único componente utilizado foi a placa de desenvolvimento da Heltec (Figura 12). Nenhum outro componente é necessário pois esta placa já contém todos os dispositivos necessários para realizar todo o processamento e comunicações.

Figura 12 – Placa Heltec WiFi LoRa (V2)



Fonte: Elaboração própria.

Apesar da simplicidade, este componente é o mais complexo do projeto, pois será necessário implementar a comunicação com a sonda via LoRa, a comunicação com a aplicação mobile via *Bluetooth* e a comunicação com o servidor web via *Wi-Fi*, através do protocolo *HTTP*.

O detalhamento do programa da estação será apresentado no capítulo XPTO, juntamente com a apresentação do software dos demais componentes.

5 Software

Neste capítulo serão detalhados os softwares desenvolvidos para atender todos os componentes deste projeto.

5.1 Ambiente de desenvolvimento

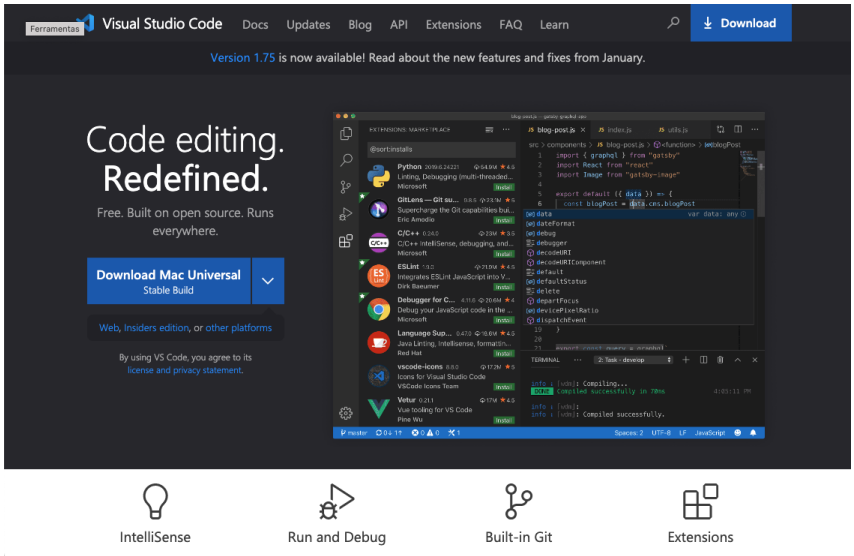
Nesta seção será detalhado as ferramentas utilizadas para o desenvolvimento dos softwares deste projeto.

5.1.1 Visual Studio Code (VSCode)

O ambiente de desenvolvimento integrado (*IDE*) Visual Studio Code, popularmente conhecido como *VSCode*, é uma ferramenta gratuita rica em recursos para apoio ao desenvolvimento de software.

O VSCode foi desenvolvido pela *Microsoft*, lançado no ano de 2015, como uma alternativa ao seu ambiente de desenvolvimento pago, o *Visual Studio*. Ele possui recursos como suporte a depuração de programas, controle de versionamento de código integrado (*GIT*), *syntax highlighting*[1], complementação inteligente de código (*autocomplete*) e ferramentas para refatoração de código.

Figura 13 – Site do Visual Studio Code



Fonte: Visual Studio Code (2023)

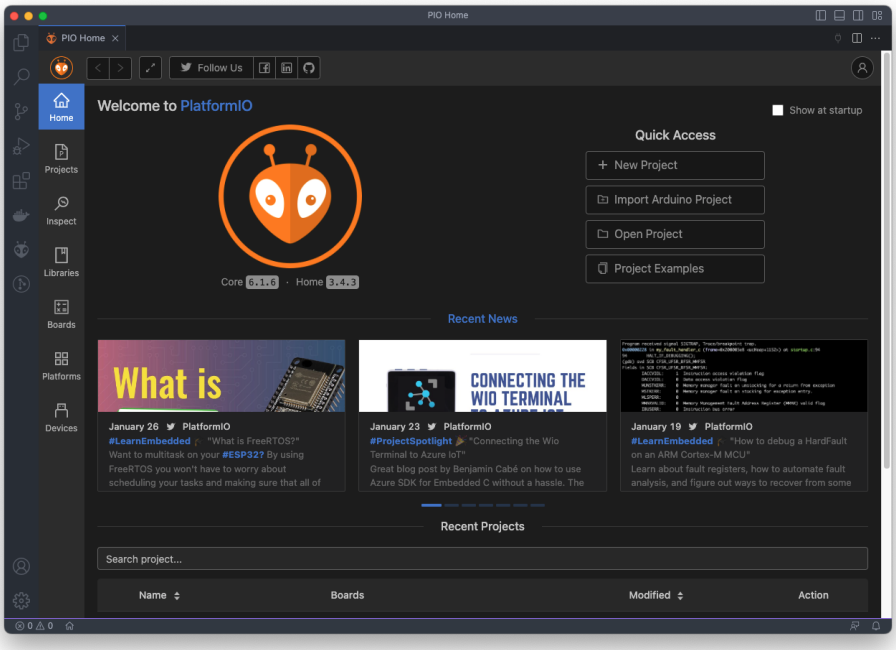
[1] *Syntax Highlighting* é uma funcionalidade disponível em alguns editores de texto que apresenta o texto numa formatação específica para cada categoria de termos (principalmente fonte e coloração).

Além dos recursos disponibilizados de forma nativa pela aplicação, a ferramenta possui um sistema de plugins, aberto ao desenvolvimento de terceiros, que estende as funcionalidades da IDE, tornando-a hoje uma das principais ferramentas de desenvolvimento de software disponíveis no mercado (Figura 13).

Para o desenvolvimento dos programas embarcados, foi utilizado o plugin para VSCode da *PlatformIO*. Segundo a definição da empresa: "PlatformIO é uma plataforma cruzada, arquitetura cruzada, estrutura múltipla, ferramenta profissional para engenheiros de sistemas embarcados e para desenvolvedores de software que escrevem aplicativos para produtos embarcados" (PLATFORMIO, 2023).

A Figura 14 mostra a tela inicial do plugin após a instalação.

Figura 14 – Tela inicial do ambiente de desenvolvimento PlatformIO no VSCode



Fonte: Elaboração própria.

5.1.2 Git

O *Git* é uma ferramenta gratuita, de código aberto, para versionamento de código. É uma ferramenta bastante simples e no entanto extremamente poderosa. É utilizado principalmente no desenvolvimento de software, porém, ela pode ser utilizada para versionamento de qualquer tipo de trabalho.

A cada alteração nos arquivos do projeto que está sendo versionado, o sistema acusa quais arquivos foram alterados e em quais partes. Quando o usuário estiver pronto para "gravar"uma nova versão, o comando *"git commit"* é utilizado. Caso o projeto esteja

publicado em algum sistema de repositórios remoto, o comando *"git push"*poderá ser utilizado para publicar a nova versão dos arquivos no repositório. A partir deste ponto, outros usuários que estejam trabalhando no mesmo projeto poderão então utilizar o comando *"git pull"* para baixar a nova versão do arquivo em suas máquinas locais. Isso torna o gerenciamento de versão do repositório distribuído, destacando o Git de outras ferramentas como o *SVN*, que possui gerenciamento centralizado, dificultando o trabalho coletivo.

Os comandos mencionados são apenas alguns exemplos de utilização da ferramenta. O Git é muito completo e possui uma grande variedade de comandos e cenários para utilização. O comando *"git –help"* mostra uma listagem com os comandos disponíveis, conforme ilustrado na Figura 15.

Figura 15 – Interface do Git

Fonte: Elaboração própria.

Neste projeto, o Git foi utilizado para o controle do código-fonte dos sistemas e publicação na plataforma de hospedagem com Git integrado chamada GitHub.

5.2 Sistemas embarcados

Nesta seção serão detalhados os softwares embarcados desenvolvidos para a sonda e a estação.

5.2.1 FreeRTOS

O ESP32 oferece suporte em seu framework de desenvolvimento a utilização do sistema operacional de tempo real RTOS (*Real Time Operational System*). O *FreeRTOS* é um kernel de RTOS desenvolvido especialmente para sistemas embarcados.

Dentre as principais características do FreeRTOS, podemos citar:

- Código livre desenvolvido sob a licença MIT;
- Provê suporte a Tasks (threads) com suporte a ajuste de prioridade, mutex, controle de semáforo, temporizadores, eventos e filas;
- Alocação de memória estática ou dinâmica para as Tasks;
- Suporta o modo de baixo consumo de energia.

A utilização de um RTOS ao invés da codificação pura no microcontrolador (bare metal) acrescenta complexidade ao projeto mas também trás algumas vantagens, tais como:

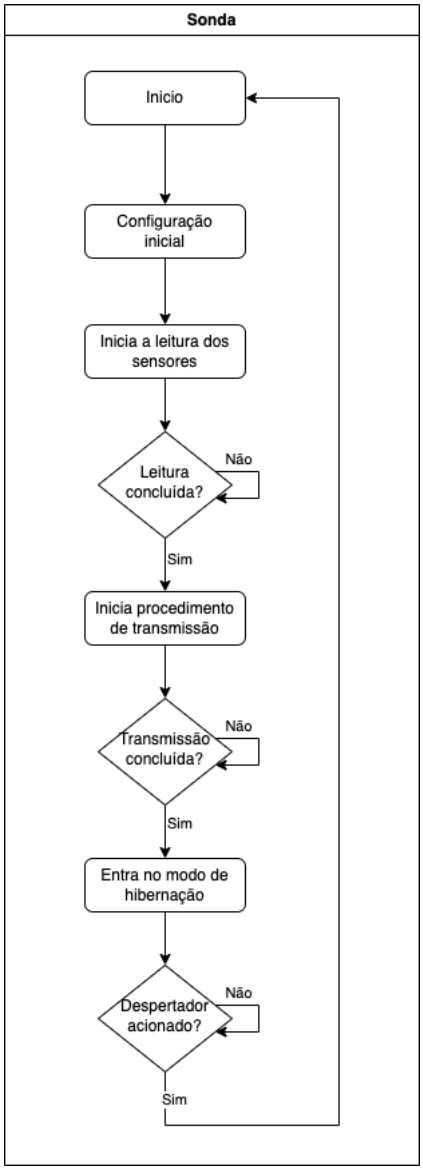
- Abstração do controle de tempo;
- Modularidade do código (tasks);
- Manutenibilidade e reusabilidade de código;
- Gerência automática de potência;
- Gerência de interrupções.

Todos os sistemas embarcados desenvolvidos neste projeto fazem uso do FreeRTOS para facilitar a escrita do código, separar as atividades em tarefas torna o entendimento do código mais fácil, além de ser mais natural o processo de programação.

5.2.2 Sistema da sonda

O sistema da sonda possui a função específica de realizar as leituras de pH e temperatura e transmiti-las, via LoRa, para a estação. O diagrama da Figura 16 mostra o fluxo do sistema.

Figura 16 – Fluxograma do programa da Sonda



Fonte: Elaboração própria.

No início da programação, é feita a importação das bibliotecas necessárias para se trabalhar com o FreeRTOS, além das bibliotecas para manipular os periféricos do ESP32. Estamos utilizando o ambiente de desenvolvimento do Arduino e a empresa Heltec também disponibiliza uma biblioteca própria para facilitar o uso dos periféricos como o display de OLED e o RF LoRa. Esta biblioteca já importa todas as outras necessárias para o desenvolvimento (Arduino e ESP32). O ambiente do PlatformIO se encarrega de compilar

o código e fazer o deploy para o microcontrolador (Figura 17).

Figura 17 – Código da Sonda - Importação das bibliotecas

```
#include <heltec.h>

#include "driver/adc.h"
#include "nvs_flash.h"
```

Fonte: Elaboração própria.

Todo sistema baseado no ambiente de desenvolvimento do Arduino possui duas funções básicas que precisam obrigatoriamente estar presentes no código, mesmo que vazias. A função setup é chamada no momento da inicialização do sistema no microcontrolador, já a função loop será chamada após a finalização do setup.

Na função setup, fazemos a inicialização de todos os periféricos que serão utilizados. Aqui estamos utilizando as comunicações Serial e RF LoRa. Além disso, criamos a tarefa que será responsável por fazer a leitura do sensor de pH, conectado ao conversor A/D 2, no canal 0 (Figura 18).

Figura 18 – Código da Sonda - Função Setup

```
// Método chamado na inicialização do programa no microcontrolador
void setup()
{
    // Display = false |  LoRa = true | Serial = true
    Heltec.begin(false, true, true, true, LORA_BAND);
    delay(20);

    // Escreve a fonte do despertador
    print_wakeup_reason();

    // Configura o timer como fonte do despertador
    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
    delay(20);

    // Inicializa o conversor A/D no canal 0
    adc2_config_channel_atten(ADC2_CHANNEL_0, ADC_ATTEN_DB_11);
    // setup_storage();
    setup_loRa();

    // Initialize control variables
    jobDone = false;
    phValue = 0.0;

    // Initial tasks
    delay(20);
    xTaskCreate(task_read_probe, "task_read_probe", 2048, NULL, 3, NULL);
}
```

Fonte: Elaboração própria.

A função loop fica encarregada de verificar se a leitura do sensor de pH já foi finalizada e transmitida. Neste ponto a sonda já terminou seu trabalho e está pronta para entrar em estado de hibernação, para poupar bateria. Os periféricos são então desligados e a placa entra no modo de hibernação (Figura 19).

Figura 19 – Código da Sonda - Função Loop

```
// Método chamado após a inicialização do programa no microcontrolador
void loop()
{
    while (1)
    {
        Serial.printf("Job done: %d \n", jobDone);

        if (jobDone)
        {
            Serial.println("Entering on sleep mode...");

            // End LoRa transmission and put on sleep mode
            LoRa.end();
            LoRa.sleep();

            // Enter on deep sleep mode (<= 800uA)
            esp_deep_sleep_start();
        }

        vTaskDelay(500 / portTICK_PERIOD_MS);
    }
}
```

Fonte: Elaboração própria.

A tarefa de leitura do sensor implementa a função definida na Equação 4.3. São obtidas 20 amostras de leitura, com o intuito de diminuir as variações do sensor, e aplicada a função de pH. Após a leitura, a tarefa de envio de dados é criada. Ao final do envio, a flag que indica que o trabalho foi concluída é marcada e na próxima execução do loop o sistema entrará no modo de hibernação (Figura 20 e Figura 21).

Figura 20 – Código da Sonda - Tarefa de leitura

```
/*** Implementação das tarefas do FreeRTOS ***/
void task_read_probe(void *pvParams)
{
    (void)pvParams;

    int value = 0;

    float avg = 0.0;
    float voltage = 0.0;

    for (int i = 0; i < 20; i++)
    {
        adc2_get_raw(ADC2_CHANNEL_0, ADC_WIDTH_12Bit, &value);

        avg += value;
        value = 0;

        vTaskDelay(50 / portTICK_PERIOD_MS);
    }

    voltage = (avg / 10) * (3.3 / 4095);
    phValue = (-8.29 * voltage) + 21.16;

    vTaskDelay(5000 / portTICK_PERIOD_MS);
    xTaskCreate(task_send_data, "task_send_data", 4096, NULL, 2, NULL);

    vTaskDelay(10 / portTICK_PERIOD_MS);
    vTaskDelete(NULL);
}
```

Fonte: Elaboração própria.

Figura 21 – Código da Sonda - Tarefa de envio

```
164  void task_send_data(void *pvParams)
165  {
166      char payload[10];
167      sprintf(payload, "%.2f;%.1f", phValue, 25.0);
168
169      Serial.printf("Sending pH value: %s\n", payload);
170
171      LoRa.beginPacket();
172      LoRa.print(payload);
173      LoRa.endPacket();
174
175      Serial.println("Sending done!");
176
177      jobDone = true;
178      vTaskDelete(NULL);
179  }
```

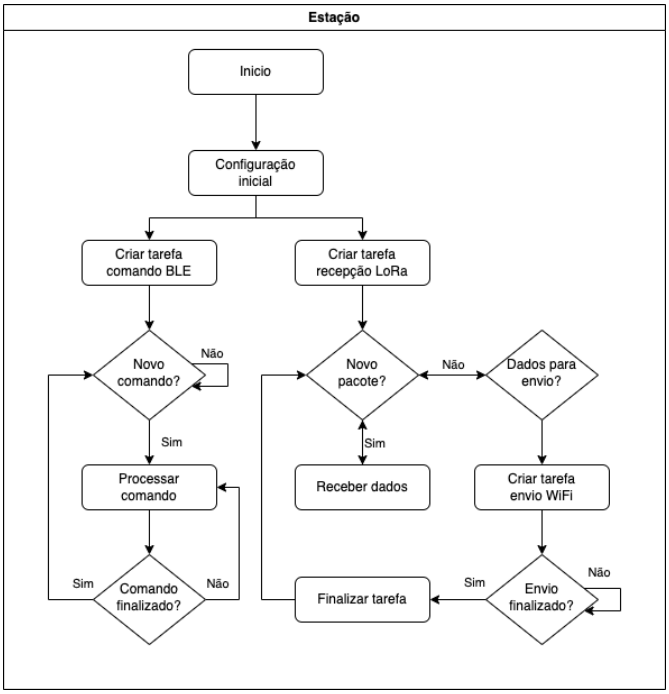
Fonte: Elaboração própria.

Após o tempo configurado de hibernação se esgotar, a plataforma é novamente inicializada, refazendo novamente todo o processo descrito acima.

5.2.3 Sistema da estação

O sistema da estação possui a função específica de recepcionar as leituras enviadas pela sonda, publicar a leitura no servidor web e comunicar com o dispositivo móvel para configuração das credenciais de acesso à conta do usuário na aplicação web e credenciais de acesso à rede WiFi. O diagrama da Figura 22 mostra o fluxo do sistema.

Figura 22 – Fluxograma do programa da Estação



Fonte: Elaboração própria.

Assim como na sonda, a primeira parte do programa realiza a importação das bibliotecas necessárias no projeto (Figura 23).

Figura 23 – Código da Estação - Importação das bibliotecas

```
#include <heltec.h>

#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino_JSON.h>

#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

#include "string.h"
#include "config.h"
```

Fonte: Elaboração própria.

A função setup inicializa todos os periféricos e cria as tarefas que serão responsáveis por monitorar as transmissões de entrada via RF LoRa e Bluetooth (Figura 24).

Figura 24 – Código da Estação - Função setup

```
void setup()
{
    Heltec.begin(true, true, true, true, LORA_BAND);
    delay(100);

    Heltec.display->setFont(ArialMT_Plain_10);
    Heltec.display->setTextAlignment(TEXT_ALIGN_LEFT);

    // Setups
    init_peripherals();

    // Tasks
    xTaskCreatePinnedToCore(task_bt_execute_cmd, "task_bt_execute_cmd", 2048, NULL, 3, NULL, 1);
    xTaskCreate(task_lora_receive_data, "task_lora_receive_data", 2048, NULL, 2, NULL);

    // Turn off the display after 10 seconds
    xTaskCreate(task_turn_display_onoff, "task_turn_display_onoff", 1024, 0, 1, NULL);
}
```

Fonte: Elaboração própria.

As configurações do projeto ficam armazenadas na memória flash interna do ESP32. Nestas configurações estão as credenciais de acesso a rede WiFi e também as credenciais de acesso à conta do usuário no servidor web. Durante a inicialização dos periféricos, o sistema verifica se existe configuração armazenada e, caso exista, prossegue com a inicialização dos mesmos. Caso contrário, uma mensagem é mostrada no display indicando que é necessário realizar a configuração da estação (Figura 25).

A seguir temos as duas tarefas responsáveis por recepcionar as leituras enviadas pela sonda e marcá-las para envio, via Wi-Fi (Figura 26 e Figura 27).

Aqui foram mostradas somente as principais funções do programa da estação. O código completo pode ser encontrado no Anexo 2.

Figura 25 – Código da Estação - Inicialização dos periféricos

```
133  void init_peripherals()
134  {
135      Heltec.display->clear();
136
137      setup_storage();
138      setup_ble();
139
140      if (config.loaded)
141      {
142          Serial.println(config.to_json_string());
143
144          setup_wifi();
145          setup_rtc();
146          setup_lora();
147
148          Heltec.display->drawString(0, 50, "Estação pronta para uso!");
149      }
150      else
151      {
152          Heltec.display->drawString(0, 20, "Realize a configuração");
153          Heltec.display->drawString(0, 30, "inicial pelo celular.");
154      }
155
156      Heltec.display->display();
157  }
```

Fonte: Elaboração própria.

Figura 26 – Código da Estação - Recepção de dados via LoRa

```
363  void task_lora_receive_data(void *params)
364  {
365      while (1)
366      {
367          int packet_size = LoRa.parsePacket();
368
369          if (packet_size > 0)
370          {
371              char packet[10] = "";
372
373              while (LoRa.available())
374              {
375                  char ch = (char)LoRa.read();
376                  strncat(packet, &ch, 1);
377              }
378              Serial.printf("LORA: Received %d bytes: %s\n", packet_size, packet);
379
380              while (has_data_to_upload)
381                  vTaskDelay(2000 / portTICK_PERIOD_MS);
382
383              String strPacket = String(packet);
384
385              lora_ph_value.clear();
386              lora_ph_value = strPacket.substring(0, strPacket.indexOf(";"));
387
388              lora_temp_value.clear();
389              lora_temp_value = strPacket.substring(strPacket.indexOf(";") + 1);
390
391              has_data_to_upload = true;
392              xTaskCreate(task_wifi_send_data, "task_wifi_send_data", 2048, NULL, 1, NULL);
393          }
394
395          vTaskDelay(1000 / portTICK_PERIOD_MS);
396      }
397  }
```

Fonte: Elaboração própria.

Figura 27 – Código da Estação - Envio de dados via Wi-Fi

```
399  void task_wifi_send_data(void *params)
400  {
401      while (1)
402      {
403          if (has_data_to_upload && WiFi.status() == WL_CONNECTED)
404          {
405              time_t tt = time(NULL);
406
407              char body[60];
408              sprintf(body, "{\"reading\":%s,\"temp\":%s,\"timestamps\":%d}",
409                  lora_ph_value.c_str(), lora_temp_value.c_str(), int32_t(tt));
410              Serial.println(body);
411
412              http.begin(config.api_url.c_str());
413
414              http.setAuthorization(config.user_email.c_str(), config.user_pass.c_str());
415              http.addHeader("Content-Type", "application/json");
416
417              int responseStatusCode = http.POST(body);
418              http.end();
419
420              has_data_to_upload = false;
421              lora_ph_value.clear();
422
423              Serial.printf("WiFi: Uploaded done with status %d\n", responseStatusCode);
424              vTaskDelete(NULL);
425          }
426
427          vTaskDelay(5000 / portTICK_PERIOD_MS);
428      }
429  }
```

Fonte: Elaboração própria.

5.3 Sistema web

Nesta seção serão detalhados os softwares desenvolvidos para o servidor e portal web.

5.3.1 Backend: Servidor web

O servidor web é o componente responsável por recepcionar os dados enviados pelas estações, armazená-los no banco de dados e prover uma interface web (via navegador de internet) para visualização destes dados. Para realizar esta tarefa foi criada apenas uma API, para recebimento dos dados.

O servidor precisa notificar os clientes conectados (portais abertos) sobre quaisquer novos dados que forem recebidos para aquela conta logada. Uma maneira de resolver este problema seria através da utilização de WebSockets, este é um protocolo de comunicação que permite comunicação *Full-Duplex* em cima da conexão TCP. Teríamos então que lidar com todas as conexões abertas e determinar para quais clientes deverá ser enviada uma determinada atualização.

Apesar de possível, atualmente existem bancos de dados de tempo real que já provêm essa funcionalidade fora da caixa. O Firebase, banco de dados não-relacional da Google, é uma dessas ferramentas. Além do banco de dados em tempo real, o Firebase possui integração com várias plataformas e linguagens, tornando uma escolha agradável

para prototipação e também para sistemas em produção.

Atualmente existem vários modelos de arquitetura de aplicações web disponíveis. Cada linguagem, empresa, emprega práticas e padrões em seus frameworks. Podemos fazer uma separação do código de servidor com as páginas que são entregues aos clientes (navegadores de internet). A parte do servidor é comumente chamada de Backend, já a parte que fica do lado do cliente de Frontend. Uma outra distinção que podemos fazer é observar onde o código está de fato sendo executado. Ao realizar uma requisição de dados em uma API de um servidor web, diversas operações e validações são executadas antes de devolver os dados. Estes dados podem estar em vários formatos, XML ou JSON, para citar alguns, ou eles podem estar embutidos em uma página HTML, que será aberta no navegador de internet. A execução deste código é chamada Server Side, pois a execução é feita do lado do servidor. Servidores web que empregam este padrão a execução do programa é feita toda do lado do servidor, ou seja, o servidor processa as páginas e devolve a página pronta, apenas para ser mostrada no cliente.

Existem programas que rodam do lado do cliente, ou seja, no navegador de internet. Essa execução é chamada de Client Side, pois está sendo executada do lado do cliente. Atualmente existem vários frameworks de aplicações Client Side, por exemplo, React.js, Angular e Solid.js, onde o servidor web irá responder a primeira requisição com o código do programa, geralmente javascript, que será executado do lado do cliente. A partir desse ponto, a aplicação do lado do cliente assume o papel principal e apenas irá requisitar os dados brutos no servidor web para serem mostrados na tela, de acordo com as iterações do usuário na página. Não é mais o servidor que monta a página (HTML) com os dados que serão mostrados, por exemplo, uma tabela de produtos. O servidor agora apenas responde com os dados dos produtos, comumente em formato JSON ou XML, e a aplicação que está rodando no cliente irá se encarregar de montar a página com os dados e mostrar na tela.

Esses frameworks para Frontend se popularizaram na última década. Eles apresentam uma solução sofisticada de computação distribuída. Rodar esse código do lado do cliente significa reduzir a carga computacional, necessária para processar todas essas páginas, no servidor. Aplicações cada vez mais robustas e rápidas estão sendo produzidas utilizando esta arquitetura (Figura 28).

Figura 28 – Javascript Frameworks

Fonte: H. e A. (2023)

Todas essas definições, Backend e Frontend, Server Side e Client Side, são importantes para definirmos a arquitetura da nossa aplicação web e quais componentes serão utilizados durante o desenvolvimento. Para a aplicação backend do sistema web foram utilizadas as seguintes linguagens, tecnologias e bibliotecas:

- Linguagem: Node.js;
- Framework: Express.js;
- Banco de dados: Firestore (Google Firebase);

O framework Express.js é um framework muito popular para criação de API's com Node.js. É um framework minimalista e baseado em middlewares. Middlewares podem ser vistos como uma cadeia de execução, ou pipes, cada middleware fica responsável por uma ação antes repassar a requisição para o próximo middleware da cadeia.

No projeto da API foi utilizado apenas um middleware para identificação do cliente que está realizando a requisição. Toda requisição deverá conter nos headers os dados de autorização (usuário e senha). O sistema então identifica o cliente e obtém as credenciais de acesso deste cliente ao sistema de banco de dados (Firestore). Caso esse procedimento seja bem sucedido, o sistema repassa a requisição para o próximo agente da cadeia, do contrário, será retornada imediatamente uma resposta de erro ao cliente, indicando a falha, neste caso, acesso não autorizado.

A Figura 29 mostra o middleware de autenticação implementado. Já a Figura 30 mostra o próximo agente da cadeia, que irá obter a propriedade da requisição, vinda do middleware de autenticação, e realizar a solicitação de inclusão dos novos dados no banco. A responsabilidade de notificação dos clientes ficará por conta do próprio serviço do Firebase, que possui integração direta com os frameworks de frontend.

O código completo do servidor está disponível no Anexo 3.

Figura 29 – Código do servidor: Middleware de autenticação

```
31  // Auth
32  app.use(async (req, res, next) => {
33    let token = '';
34    const basicToken = (req.headers.authorization || '').trim();
35
36    if (basicToken !== '' && basicToken.includes('Basic')) {
37      const base64Token = basicToken.split(' ')[1];
38
39      const [email, password] = Buffer.from(base64Token, 'base64')
40        .toString()
41        .split(':');
42
43      const url = `${GOOGLE_API_URL}?key=${process.env.FB_API_KEY}`;
44      const body = { email, password, returnSecureToken: true };
45
46      const res = await axios.post(url, body, {
47        headers: new AxiosHeaders().set('Content-Type', 'application/json'),
48      });
49
50      token = res.data.idToken;
51    } else {
52      token = (req.header('X-Token-Api') || '').trim();
53    }
54
55    if (token !== '') {
56      try {
57        const decodedToken = await auth.verifyIdToken(token);
58        req.user = await auth.getUser(decodedToken.uid);
59
60        return next();
61      } catch (_) {}
62    }
63
64    res.status(401).end('Unauthorized');
65  });
```

Fonte: Elaboração própria.

Figura 30 – Código do servidor: API para gravar os dados

```
85  app.post('/api/data', async (req, res) => {
86    const userId = req.user?.uid ?? '';
87
88    if (userId.trim() !== '') {
89      try {
90        const valid = req.body.reading >= 0 && req.body.reading <= 14;
91        await db
92          .collection(`accounts/${userId}/data`)
93          .add({ ...req.body, valid });
94        return res.status(201).end();
95      } catch (error) {
96        console.error(error);
97        return res.status(500).json(error);
98      }
99    }
100
101   res.status(200).end();
102 });
```

Fonte: Elaboração própria.

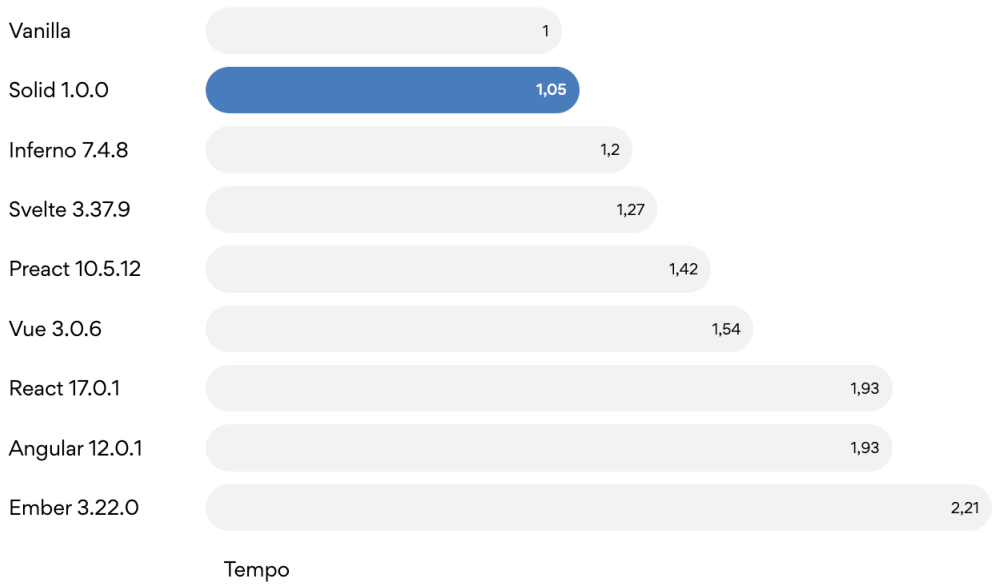
5.3.2 Frontend: Portal do pesquisador

O portal do pesquisador é uma página web onde os pesquisadores poderão acompanhar as leituras da sonda medidora de pH em tempo real.

Seguindo a arquitetura distribuída, comentada na seção anterior, foi escolhida a biblioteca Solid.js para desenvolvimento do portal. O Solid.js é uma biblioteca para construção de UI's (*user interface*) ricas e dinâmicas, seguindo, de forma muito parecida, com a famosa biblioteca *React.js*.

Segundo um benchmark disponibilizado no próprio site do Solid.js (Figura 31), a performance desta biblioteca vence todas as outras bibliotecas ou frameworks existentes atualmente, perdendo apenas para o Javascript puro (executado diretamente, sem intermediários).

Figura 31 – Solid.js - Benchmark de performance



Fonte: (SOLID.JS, 2023).

O sistema consiste em uma área para autenticação e registro e uma área para usuários logados com um gráfico que será atualizado automaticamente à medida que novos dados forem coletados pela sonda.

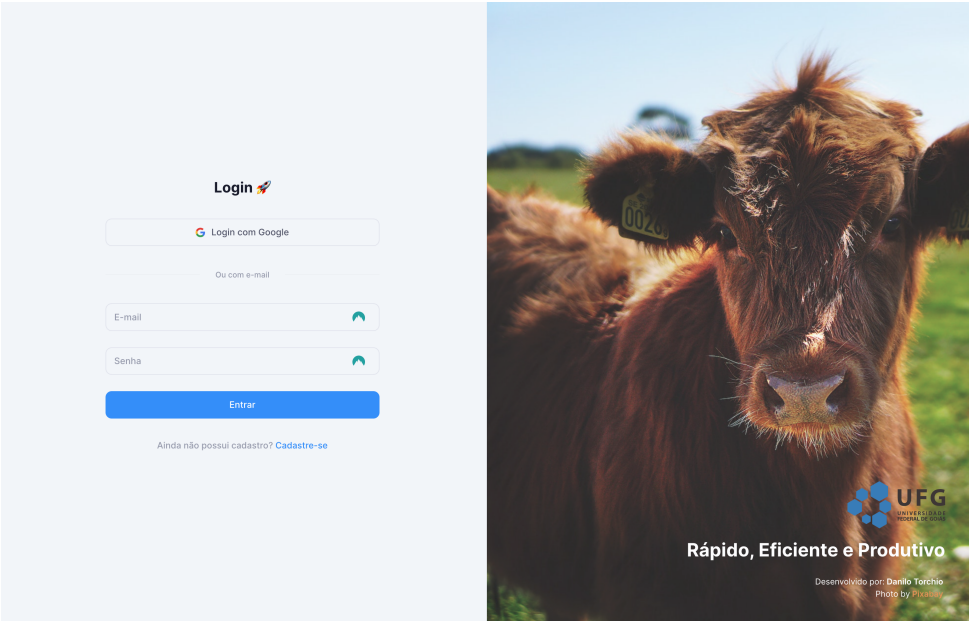
5.3.2.1 Autenticação

A área de autenticação possui duas páginas, sendo a primeira para realizar o login na aplicação e a outra para realizar o registro.

O login conta com um campo de e-mail, senha e um botão para confirmar o login. Também é possível realizar o login diretamente utilizando seu login do Google, clicando

no botão "Login com Google". A página também contém um link para o usuário navegar para a parte de registro, conforme ilustrado na Figura 32.

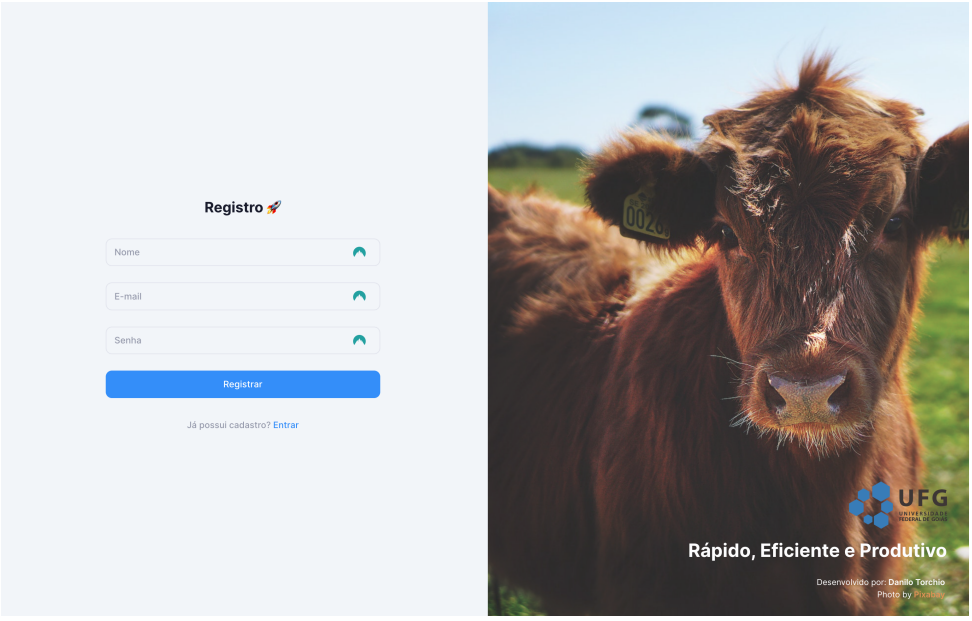
Figura 32 – Portal do pesquisador - Página de Login



Fonte: Elaboração própria.

O registro, conta com os campos de nome, e-mail, senha e botão de confirmação, conforme ilustrado na Figura 33.

Figura 33 – Portal do pesquisador - Página de Registro



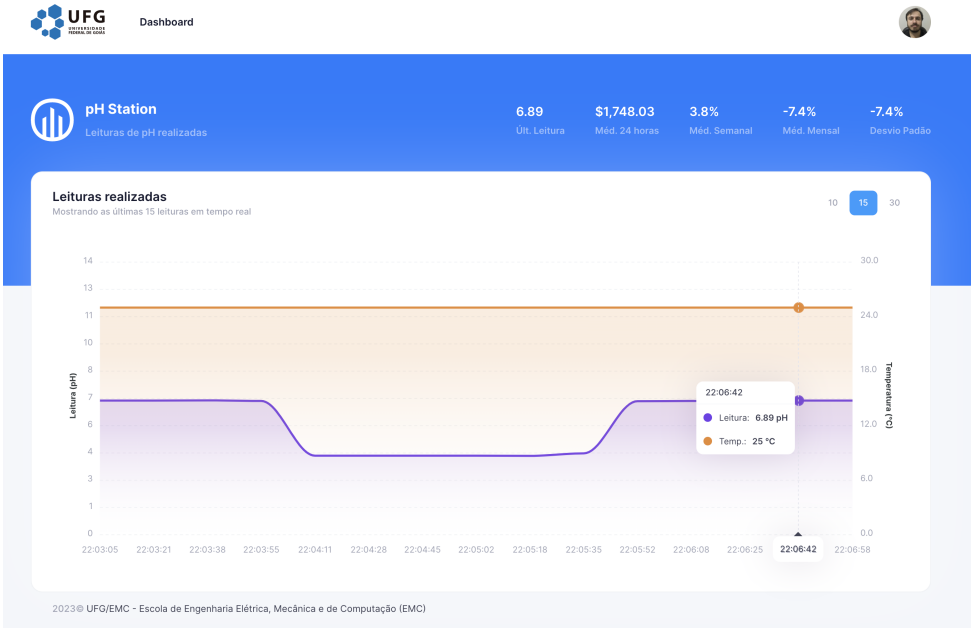
Fonte: Elaboração própria.

5.3.2.2 Dashboard

Após a autenticação ser realizada com sucesso, o usuário será direcionado para a página de dashboard, que contém o gráfico que será atualizado automaticamente sempre que novas leituras forem enviadas pela estação.

A página é demonstrada na Figura 34.

Figura 34 – Portal do pesquisador - Dashboard



Fonte: Elaboração própria.

A atualização automática é realizada por meio da utilização da biblioteca do Firebase para Javascript. Utilizando o método "onSnapshot", importado da biblioteca "firebase/firestore", o Firestore irá notificar o cliente sempre que uma nova atualização estiver disponível nos dados. Através da programação reativa, o sistema "reage"a essas mudanças reconstruindo o gráfico com os novos dados. O método da Figura 35 cria um gatilho para que, a qualquer mudança nos dados devolvidos pela consulta (query) realizada no banco, o sistema irá setar, ou seja, atualizar, os dados do gráfico, através do método "setChartData"destacado.

O código completo das páginas de login e dashboard estão disponíveis nos anexos 4 e 5.

5.4 Sistema para dispositivos móveis (Android e iOS)

O aplicativo para dispositivo móvel tem por objetivo realizar a configuração da estação e sonda. As configurações que podem ser realizadas são:

Figura 35 – Portal do pesquisador - Método de atualização automática dos gráficos

```
createEffect(() => {
  if (!chart.rendered) initChart();
  if (!!unsub) unsub();

  unsub = onSnapshot(cQuery(), (snap) => {
    if (snap.empty) return;
    const dataArray = snap.docs.map((doc) => doc.data()).sort((a, b) => a.timestamps - b.timestamps);

    setChartData(
      dataArray.map((data) => {
        const date = DateTime.fromSeconds(data.timestamps);

        return {
          value: data.reading.toFixed(2),
          temp: data.temp.toFixed(1),
          label: `${date.toLocaleString(DateTime.TIME_WITH_SECONDS)}`,
        };
      })
    );

    setTimeout(() => updateChart());
  });
});
```

Fonte: Elaboração própria.

- Credenciais de acesso a rede Wi-Fi;
- Credenciais de acesso a conta da plataforma web;
- Intervalo de medições na sonda medidora de pH.

5.4.0.1 Flutter

O aplicativo foi desenvolvido utilizando o framework da Google chamado Flutter. Este framework possibilita a construção de aplicações nativas para Android, iOS, Windows, Mac e Linux utilizando o mesmo código fonte, o que proporciona grande ganho de performance no desenvolvimento de aplicações móveis.

A linguagem utilizada para desenvolvimento é o Dart, também desenvolvida pela Google e é a mesma utilizada para a construção do framework Flutter.

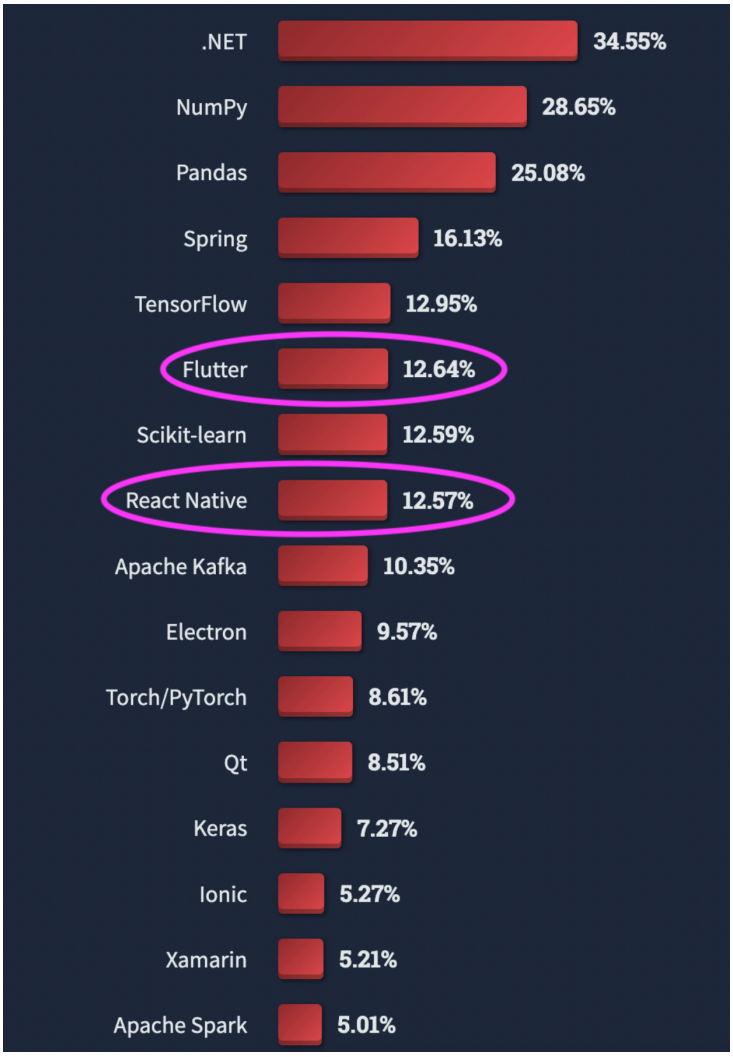
O projeto do Flutter foi iniciado em 2014, na época sob o codinome "sky"e o objetivo central era encontrar uma maneira eficiente de se construir interfaces para aplicações mobile. A versão 1.0 estável do framework foi lançada oficialmente em dezembro de 2018 e vem em crescente aceitação desde então. Atualmente, na versão 3.7, o framework se destaca na qualidade, velocidade e recursos com relação ao seu concorrente direto, o React Native.

Segundo Nowak (2023), do portal *nomtek*, em 2023 o Flutter ultrapassou seu principal concorrente nos índices de popularidade, como mostra a Figura 36.

5.4.0.2 Aplicativo Mobile

O aplicativo desenvolvido possui as seguintes páginas:

Figura 36 – Flutter - Pesquisa anual de desenvolvedores

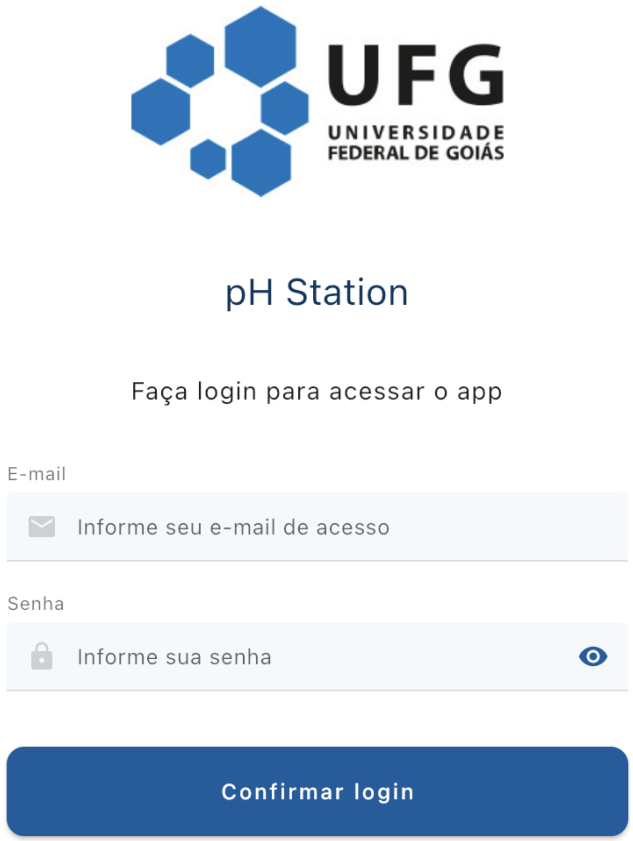


Fonte: Elaboração própria.

- Autenticação;
- Tela de busca de estações (via Bluetooth);
- Tela de configuração da estação (via Bluetooth).

A tela de autenticação segue o mesmo princípio da página web, devendo ser informado o e-mail e a senha da conta criada no portal web, conforme ilustrado na Figura 37.

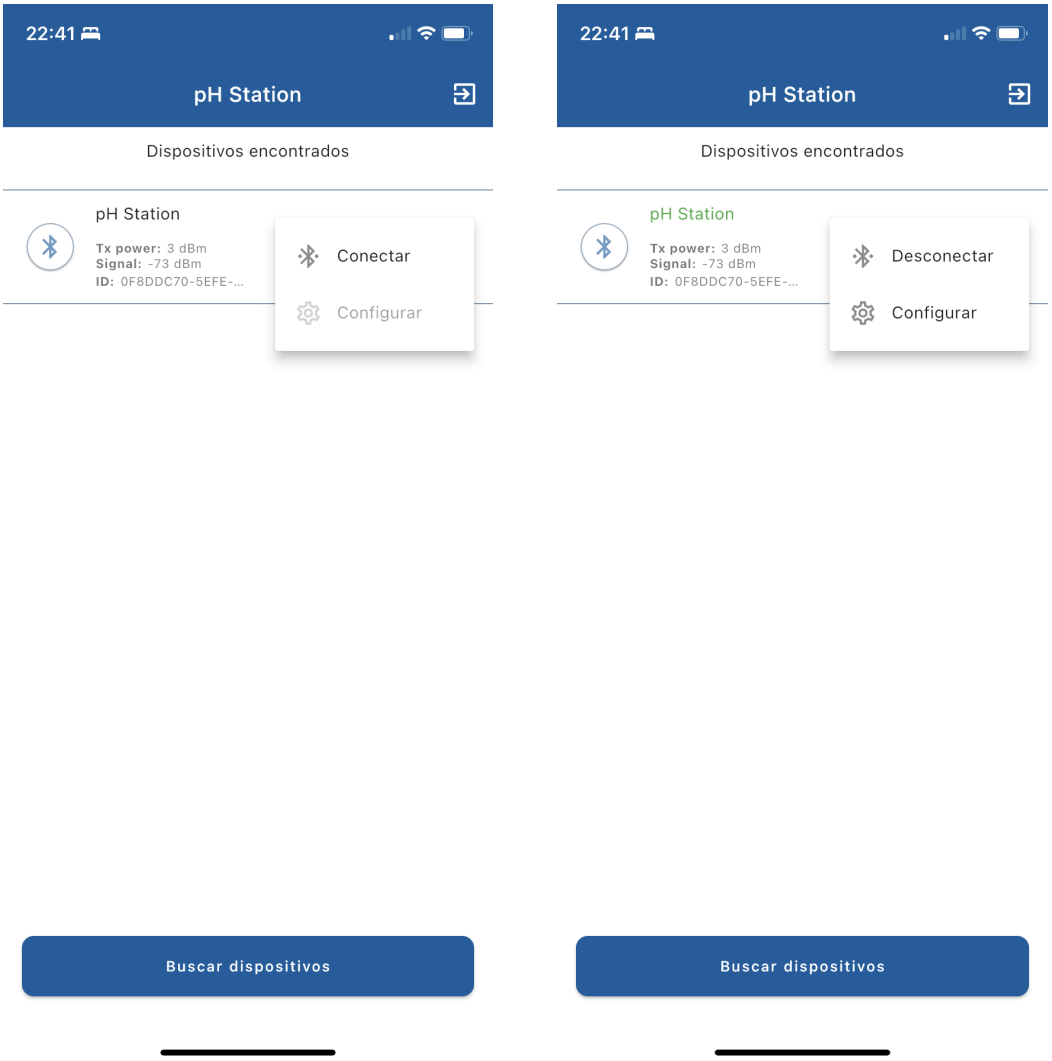
Figura 37 – Aplicativo: Tela de Login



Fonte: Elaboração própria.

Após a autenticação, o usuário será direcionado para tela principal, onde irão aparecer as estações próximas. Clicando no botão conectar, a tela de configuração ficará acessível, conforme ilustrado na Figura 38.

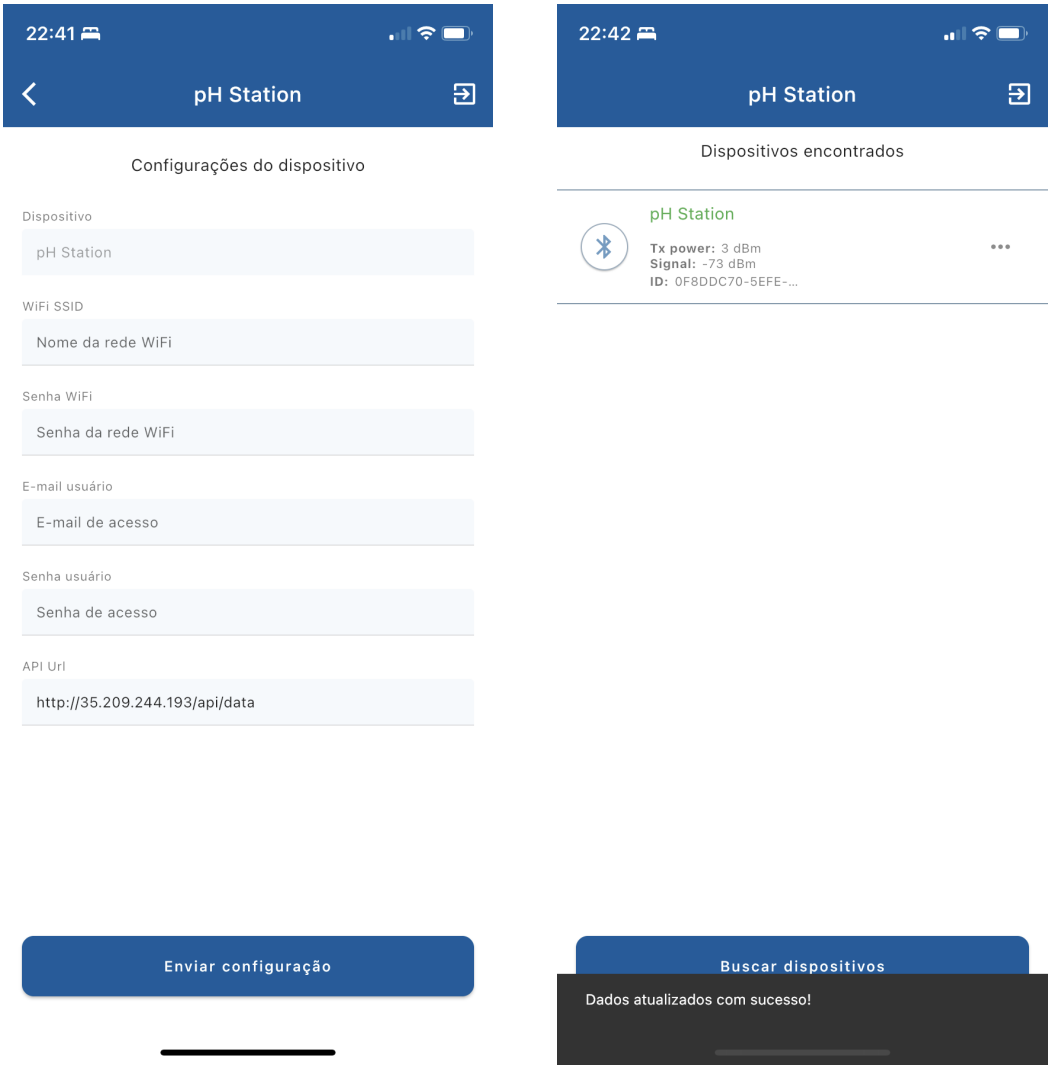
Figura 38 – Aplicativo: Home Page



Fonte: Elaboração própria.

A tela de configuração possui todos os campos necessários para atualizar as credenciais de acesso da rede WiFi e as credenciais da conta, além do campo para informar o intervalo de leituras da sonda medidora de pH, conforme Figura 39.

Figura 39 – Aplicativo: Configurações



Fonte: Elaboração própria.

Ao clicar em "Enviar configuração", o aplicativo envia para a estação as novas configurações. Caso as configurações sejam salvas com sucesso, o aplicativo mostra uma mensagem para o usuário indicando que deu tudo certo e navega automaticamente para a tela inicial.

O código deste aplicativo está disponível no Anexo 5.

Parte III

TESTES E RESULTADOS

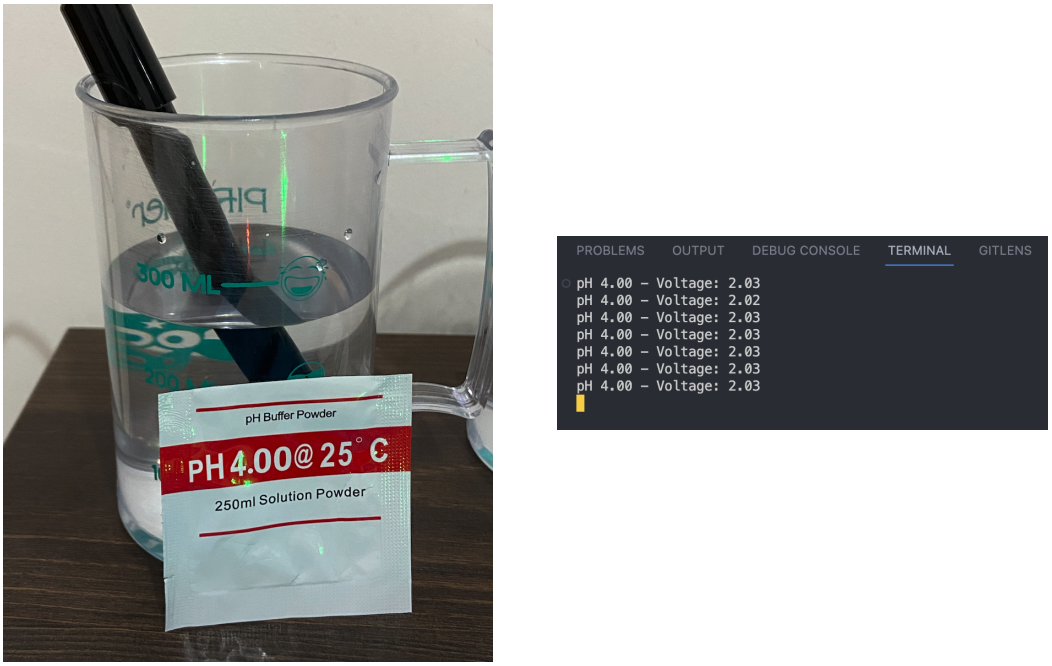
6 Testes

Neste capítulo serão mostrados os testes de calibração da sonda, configuração da estação e a visualização no em tempo real no portal do pesquisador das medições realizadas.

6.1 Calibração da sonda

Conforme explicado no capítulo 4, seção 4.2, é necessário fazer a calibração da sonda. Começamos medindo a tensão de saída para a solução 1 (pH 4), conforme Figura 40.

Figura 40 – Testes: Calibração da Sonda 1



Fonte: Elaboração própria.

O mesmo procedimento foi realizado para a solução 2 (pH 6.86), conforme Figura 41.

Figura 41 – Testes: Calibração da Sonda 2

Fonte: Elaboração própria.

Resultados:

- pH $4,00 = 2,03V$
- pH $6,86 = 1,70V$

Aplicando a Figura 4.2 e Equação 4.2 encontramos a função da reta reduzida para cálculo do pH,

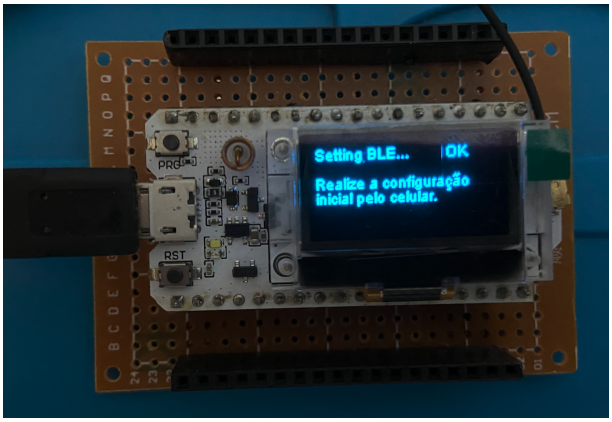
$$y = -8,66x + 21,58 \tag{6.1}$$

Observação: A função acima é diferente da função encontrada durante as pesquisas no capítulo 4. Isso ocorreu provavelmente devido a impurezas na solução, tendo em vista que a mesma foi reutilizada durante vários dias. Para os testes, foram produzidas novas soluções.

6.2 Configuração da estação

Sem a configuração inicial da estação (memória limpa), ao ligar a estação, a mensagem demonstrada na Figura 42 é apresentada, conforme previsto.

Figura 42 – Testes: Estação desconfigurada



Fonte: Elaboração própria

Após realizar as configurações, a sonda foi inicializada corretamente, conforme mostrado na Figura 43.

6.3 Leituras em tempo real

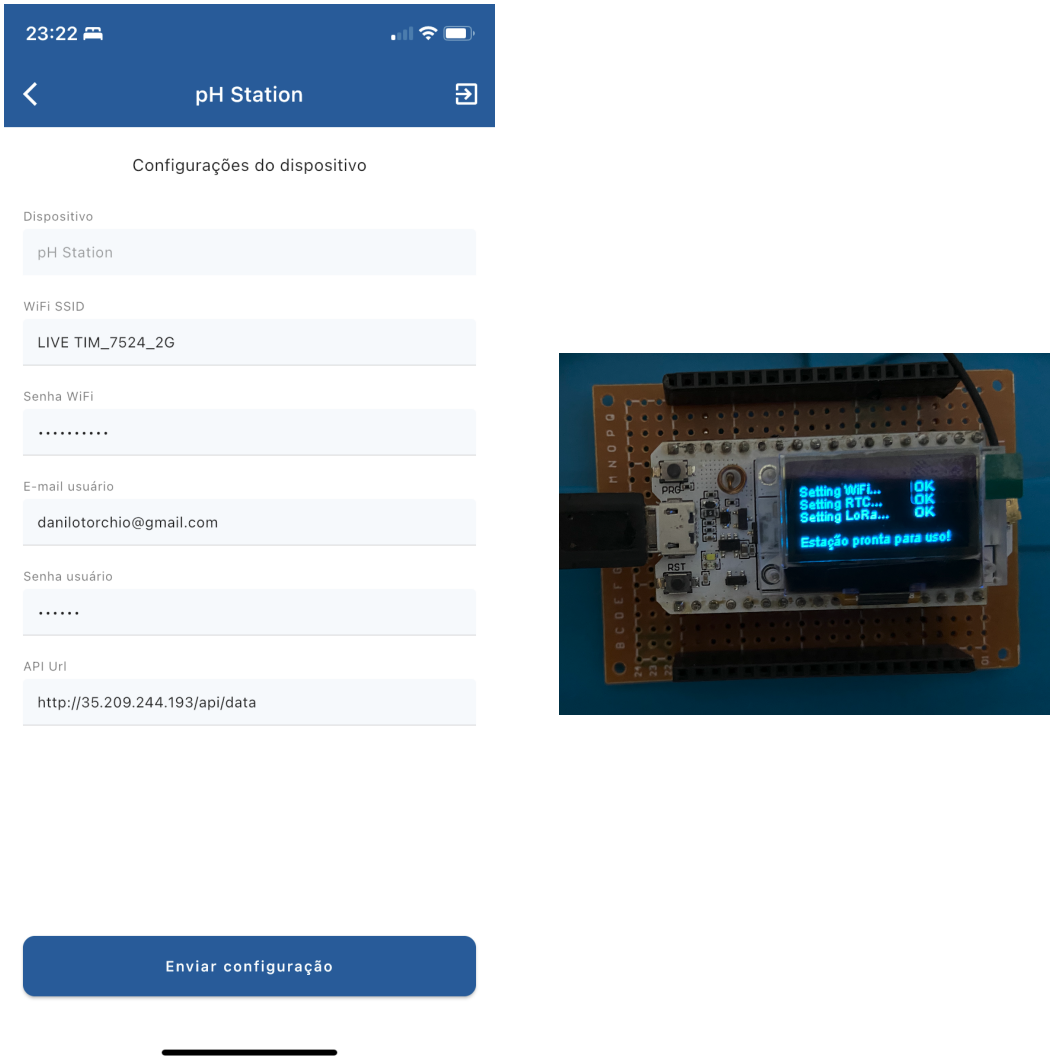
Com a sonda calibrada e a estação configurada, foram testadas as leituras e a apresentação em tempo real.

Iniciando pela solução de pH 6,86, podemos observar que os dados começam a aparecer no portal, conforme Figura 44.

Alterando a solda para a solução de pH 4.00, as novas leituras começam a chegar no portal, conforme a Figura 45.

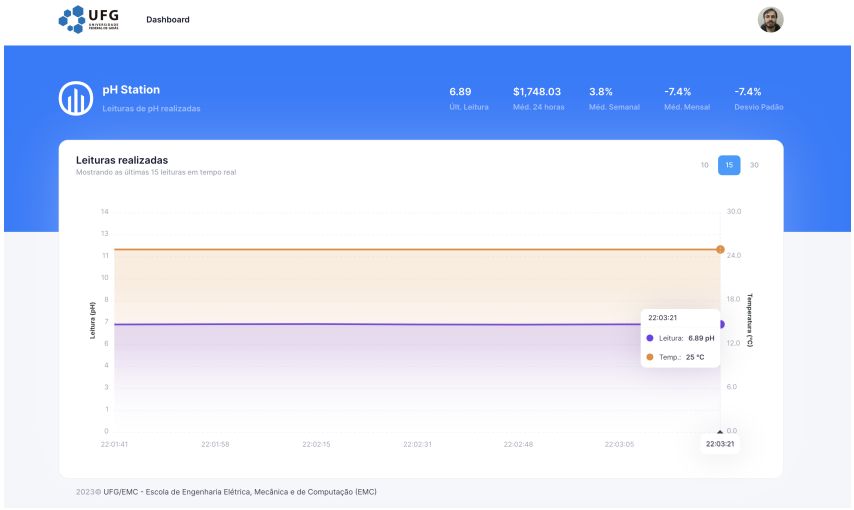
Voltando novamente a sonda para a solução de pH 6,86, os dados são atualizados novamente, conforme Figura 46.

Figura 43 – Testes: Estação desconfigurada



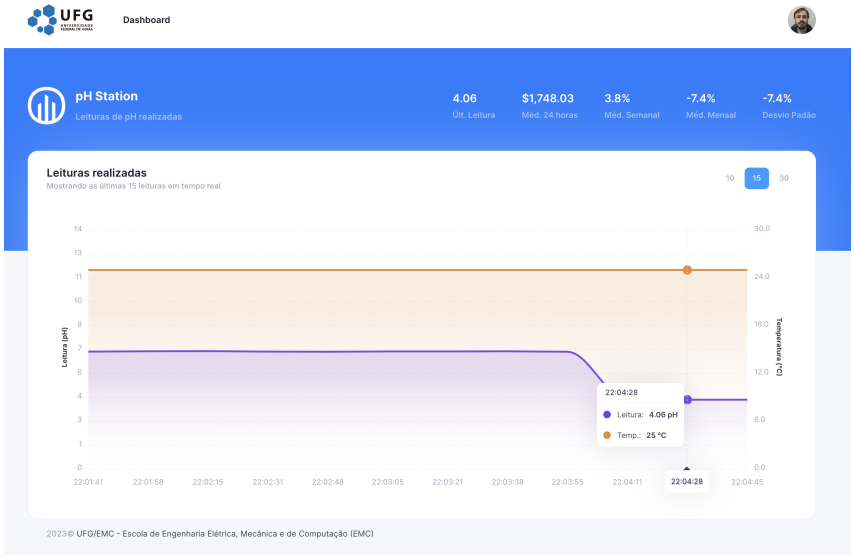
Fonte: Elaboração própria

Figura 44 – Testes: Testando as atualizações em tempo real 1



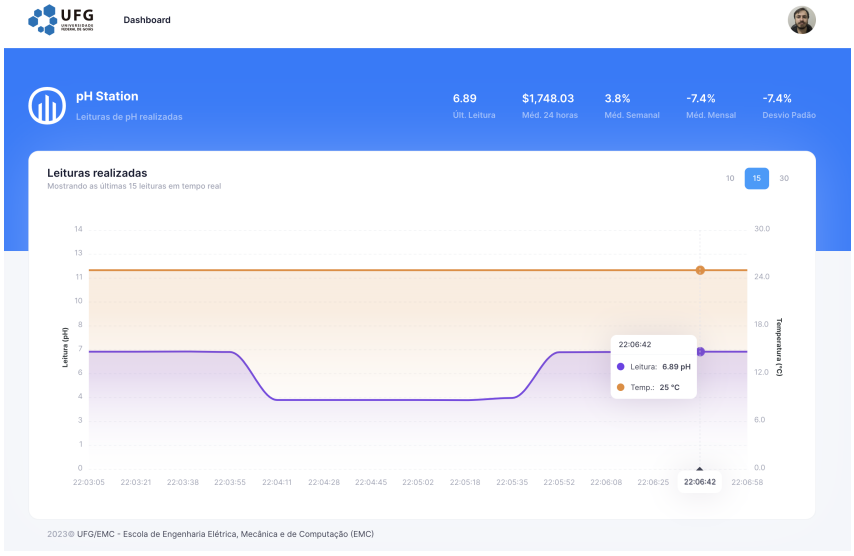
Fonte: Elaboração própria

Figura 45 – Testes: Testando as atualizações em tempo real 2



Fonte: Elaboração própria

Figura 46 – Testes: Testando as atualizações em tempo real 3



Fonte: Elaboração própria

7 Resultados

Todos os testes realizados ocorreram como esperado. A Tabela 3 mostra os resultados obtidos.

Tabela 3 – Resultados dos testes de leitura de pH

Solução	**Valor Lido**	**Erro**
$4,00$	$4,06$	$\frac{\lvert 4,06 - 4,00 \rvert}{4,00} \cdot 100 = 1,5\%$
$6,86$	$6,89$	$\frac{\lvert 6,89 - 6,86 \rvert}{6,86} \cdot 100 = 0,4\%$

Fonte: Espressif (2023)

Conforme podemos observar, as leituras apresentam o erro inferior a 2%, o que é aceitável para este projeto. Os testes de configuração da sonda e estação ocorreram sem erros e o tempo medido para atualização do portal ficou em média 1 segundo.

8 Conclusão

O objetivo desta monografia foi melhorar o dispositivo de medição de pH intra-ruminal existente no Instituto de Ciências Biológicas (ICB-UFG) da Universidade Federal de Goiás (UFG), no sentido de aprimorar a experiência do usuário e, também, as condições do animal.

Para alcançar este objetivo, uma pesquisa foi realizada sobre técnicas e componentes com recursos avançados, específicos para IoT, visando economia de energia, boa capacidade computacional em baixa potência e transmissão de dados via meio sem fio.

A partir dos resultados da pesquisa, foram construídos protótipos da sonda e uma estação de recebimento de dados, utilizando placas de desenvolvimento pré-fabricadas da empresa Heltec, desenvolvida com base no microcontrolador ESP32 e com tecnologia LoRa de transmissão RF.

Os testes mostraram que é possível atender aos requisitos com as tecnologias escolhidas, porém, o custo de aquisição dos componentes ficou elevado. Para prototipação rápida e validação das tecnologias, o valor foi considerado aceitável, mas para produção efetiva no mercado deverão ser estudadas alternativas para desenvolvimento da placa da sonda e estação.

O software desenvolvido para visualização em tempo real das informações atendeu as expectativas. Foram utilizadas somente ferramentas gratuitas, de código livre, e a experiência de usuário final se mostrou promissora para continuar com o trabalho na forma que foi proposto. A utilização do aplicativo mobile para configuração da estação tornou o processo muito mais fácil e humano.

A partir do desenvolvimento dos protótipos, ficou evidente que é possível estender o projeto da aplicação para várias sondas e utilizá-las em outros cenários, não apenas para medição de pH.

No geral o projeto atendeu as expectativas e conseguiu cumprir a proposta de otimização e experiência.

8.1 Trabalhos futuros

Como proposta para o trabalhos futuros, alguns problemas encontrados durante o desenvolvimento deste trabalho podem ser levados em consideração:

- Placas de desenvolvimento pré-fabricadas como Arduino, ESP32 DevKit e Heltec WiFi LoRa são excelentes para prototipação mas na prática existe um enorme desperdício de recursos. A placa da Heltec, por exemplo, possui display OLED integrado, que fica desligado o tempo todo no projeto da sonda. O projeto da placa própria, apenas com os recursos necessários para o projeto, deverá ficar muito mais em conta.
- As dimensões do sensor de pH impedem a diminuição física da sonda. Um estudo mais aprofundado na parte de sensoriamento talvez mostre alternativas melhores para este tipo de medição.
- Além do pH, outros sensores podem ser utilizados para compor uma lista de informações. Um trabalho em conjunto com os pesquisadores de outras áreas poderá abrir novas oportunidades de sensoriamento.
- Utilizar uma rede LoRaWAN para montar uma rede de sensores para o mesmo projeto.

Referências

ALEXANDER, C. K.; SADIKU, M. N. O. *Fundamentos de Circuitos Elétricos*. 5. ed. Porto Alegre: AMGH Editora, 2013. Citado na página 35.

BUYYA, R.; DASTJERDI, A. V. *Internet of Things*: Principles and paradigms. 1. ed. Cambridge, MA: Morgan Kaufmann, 2016. Citado na página 23.

DIAS, D. L. Conceito de ph. *Brasil Escola*, 2023. Disponível em: <https://brasilescola.uol.com.br/quimica/conceito-ph.htm>. Acesso em: 22 fev 2023. Citado na página 21.

ESPRESSIF. *Espressif Systems*. 2023. Disponível em: <https://www.espressif.com>. Acesso em: 18 fev 2023. Citado 4 vezes nas páginas 25, 27, 28 e 65.

H., V.; A., S. *The Best JS Frameworks for Front End*. 2023. Disponível em: <https://rubygarage.org/blog/best-javascript-frameworks-for-front-end>. Acesso em: 21 fev 2023. Citado na página 49.

Heltec Automation. *WiFi LoRa 32 (V2)*: Lora node development kit. [S.l.], 2023. Disponível em: <https://resource.heltec.cn/download/Manual%20Old/WiFi%20Lora32Manual.pdf>. Acesso em: 20 fev 2023. Citado na página 29.

MIYADAIRA, A. N. *Microcontroladores PIC18*: Aprenda e programe em linguagem c. 4. ed. São Paulo: Editora Érica, 2009. Citado na página 23.

NOVAIS, S. A. O que é ph? *Brasil Escola*, 2023. Disponível em: <https://brasilescola.uol.com.br/o-que-e/quimica/o-que-e-ph.htm>. Acesso em: 19 fev 2023. Citado na página 21.

NOWAK, M. 2023. Disponível em: <https://www.nomtek.com/blog/flutter-vs-react-native>. Acesso em: 21 fev 2023. Citado na página 54.

OLIVEIRA, R. R. de. Equação reduzida da reta. *Brasil Escola*, 2023. Disponível em: <https://brasilescola.uol.com.br/matematica/equacao-reduzida-reta.htm>. Acesso em: 20 fev 2023. Citado na página 34.

PLATFORMIO. *What is PlatformIO?* 2023. Disponível em: <https://docs.platformio.org/en/latest/what-is-platformio.html>. Acesso em: 21 fev 2023. Citado na página 38.

SABADIN, V. *Manutenção em eletrodo de pH*: Mecatrônica atual: Automação industrial de processos e manufatura. 1. ed. Cambridge, MA: [s.n.], 2003. Citado na página 21.

SOLID.JS. 2023. Disponível em: <https://www.solidjs.com/>. Acesso em: 21 fev 2023. Citado na página 51.

Visual Studio Code. 2023. Disponível em: <https://code.visualstudio.com/>. Acesso em: 21 fev 2023. Citado na página 37.