UNIVERSIDADE FEDERAL DE SANTA CATARINACENTRO TECNOLÓGICO
CENTRO TENCOLÓGICO - CTC
DEPARTAMENTO DE ENGENHARIA QUÍMICA E DE ALIMENTOS - EQA
CURSO DE ENGENHARIA DE ALIMENTOS

Jackelyne de Souza Carvalho

INTERNET OF THINGS EM BIOTECNOLOGIA: ARDUINO NO CULTIVO DE MICROALGAS

Florianópolis – SC
2022

Jackelyne de Souza Carvalho

INTERNET OF THINGS EM BIOTECNOLOGIA: ARDUINO NO CULTIVO DE MICROALGAS

Trabalho de conclusão de curso de Graduação em Engenharia de Alimentos do Centro Tecnológico da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Bacharel em Engenharia de Alimentos.

Orientador: Prof. Dr. Cristiano José de Andrade

Florianópolis – SC

2022

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

de Souza Carvalho, Jackelyne
Internet of things em biotecnologia: Arduino no cultivo de microalgas / Jackelyne de Souza Carvalho ; orientador, Cristiano José de Andrade, 2022.
75 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, , Graduação em Engenharia de Alimentos, Florianópolis, 2022.

Inclui referências.

1. Engenharia de Alimentos. 2. Internet das coisas (IoT). 3. Sensores de baixo custo. 4. Microcontrolador. 5. Chlorella vulgaris. I. José de Andrade, Cristiano. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Alimentos. III. Título.

Jackelyne de Souza Carvalho

INTERNET OF THINGS EM BIOTECNOLOGIA: ARDUINO NO CULTIVO DE MICROALGAS

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Bacharel em Engenharia de Alimentos e aprovado em sua forma final pelo Curso de Graduação de Engenharia de Alimentos.

Florianópolis, 22 de março de 2022.

Documento assinado digitalmente
Joao Borges Laurindo
Data: 09/06/2022 15:38:00-0300
CPF: 052.167.828-50
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Dr. João Borges Laurindo
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Cristiano Jose de Andrade
Data: 02/06/2022 18:41:18-0300
CPF: 311.875.168-13
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Dr. Cristiano José de Andrade
Orientador

Documento assinado digitalmente
Marco Di Luccio
Data: 02/06/2022 19:17:00-0300
CPF: 016.865.237-42
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Dr. Marco Di Luccio
Avaliador

Documento assinado digitalmente
Agenor Furigo Junior
Data: 08/06/2022 17:10:20-0300
CPF: 867.789.438-15
Verifique as assinaturas em https://v.ufsc.br

_____

Prof. Dr. Agenor Furigo Junior
Avaliador

Este trabalho é dedicado ao meu companheiro e à minha família, pelo apoio em todos os momentos da minha vida.

AGRADECIMENTOS

Em primeiro lugar à minha família, que sempre me apoiou e me incentivou durante o período da graduação.

Ao meu companheiro, pelo apoio incondicional, por me introduzir ao mundo da programação e me ensinar tanto sobre o assunto.

Aos meus amigos da faculdade, que sempre estiveram ao meu lado e trilharam esse caminho junto a mim.

Ao meu orientador Prof. Dr. Cristiano José de Andrade, por ter me ensinado tanto nos últimos quatro anos e ter desempenhado a função de orientador com dedicação e confiança.

À Vanessa, mestranda que me acompanhou nos últimos 2 anos, que me ensinou muito e tornou a minha vivência no laboratório mais leve.

À Universidade Federal de Santa Catarina, por propiciar a minha evolução pessoal e profissional e me permitir vivenciar a rotina de laboratório, que eu tanto aprecio.

Aos membros da banca examinadora pelas contribuições.

Aos professores do Departamento de Engenharia Química e de Alimentos, pelos ensinamentos e conselhos que enriqueceram o meu aprendizado.

E a todos que de certa forma fizeram parte da minha formação, o meu muito obrigada.

RESUMO

A Internet das coisas (IoT) é um aperfeiçoamento do conceito de Internet que visa ampliar as suas vantagens de conectividade, permitindo o acompanhamento contínuo e em tempo real de dados e o compartilhamento destes. O objetivo deste trabalho foi aplicar a IoT no monitoramento em tempo real do cultivo da microalga *Chlorella vulgaris*. Para medir os parâmetros associados ao crescimento microalgal, foi desenvolvido um sistema de baixo custo formado por sensores Arduino de temperatura, pH, turbidez e temperatura com conexão à nuvem e exibição dos dados em aplicação *web*. Os dados de pH e turbidez dos sensores foram validados por meio de pHmetro e espectrofotômetro (equipamentos laboratoriais clássicos), respectivamente. O sistema proposto é consideravelmente de menor custo – cerca de 95% de economia - quando comparado aos equipamentos comerciais. Além disso, foi observada uma boa correlação, especialmente, nos sensores de temperatura, pH e luminosidade, viabilizando o uso em futuros estudos.

Palavras-chave: Internet das coisas (IoT). Sensores de baixo custo. Microcontrolador. *Chlorella vulgaris*.

ABSTRACT

The Internet of Things (IoT) is an improvement of the Internet concept that aims to expand its connectivity advantages, allowing continuous and real-time monitoring of data and their sharing. The purpose of this work is to apply the IoT in real-time monitoring of the microalgae *Chlorella vulgaris* cultivation. To measure the parameters associated with microalgal growth, a low-cost system was developed consisting of Arduino sensors for temperature, pH, turbidity and temperature, connected to the cloud and displaying the data in a web application. The pH and turbidity data from the sensors were validated using a pH meter and spectrophotometer (classical laboratory equipment), respectively. The proposed system is considerably lower cost - around 95% savings - when compared to commercial equipment. In addition, a good correlation was observed, especially in temperature, pH and luminosity sensors, enabling the use in future studies.

Keywords: Internet of Things (IoT). Low-cost sensors. Microcontroller. *Chlorella vulgaris*.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BD	Banco de dados
C. vulgaris	*Chlorella vulgaris*
IDC	*International Data Corporation*
IoT	Internet das coisas
LDR	*Light-Dependent Resistor*
UV	Ultravioleta

SUMÁRIO

1. INTRODUÇÃO

A Internet das coisas (IoT) é uma tecnologia emergente que está cada vez mais presente na nossa sociedade. A IoT interconecta sistemas e dispositivos por meio da Internet, garantindo coleta de informações de diferentes origens e interações entre elas, independente da localização e do momento (PERWEJ *et al.*, 2019; SETHI e SARANGI, 2017).

Essa tecnologia desperta cada vez mais interesse por interconectar o real e o virtual e pela sua alta capacidade de compartilhamento de dados e monitoramento destes em tempo real. Segundo a *International Data Corporation* (IDC, 2022), em 2025 haverá mais de 84 bilhões de dispositivos IoT conectados.

Neste contexto, a IoT é uma promissora ferramenta para o controle e aperfeiçoamento de processos biotecnológicos, como o cultivo microalgal. Supervisionando parâmetros como a luminescência da fonte (e.g LEDs), temperatura, pH, concentração de nutrientes e oxigênio dissolvido (KHOO *et al.*, 2020; RAHMAT *et al.*, 2020).

Em relação aos processos biotecnológicos, os sensores tradicionalmente usados são, por um lado, fundamentais (monitoramento), por outro lado são dispendiosos (aquisição e manutenção). Assim, uma promissora alternativa são os sensores à base de Arduino, um microcontrolador de baixo custo que pode ser facilmente aplicado a basicamente todos processos biotecnológicos (MONK, 2013).

Portanto, de forma inédita, o objetivo deste trabalho foi monitorar o cultivo da microalga *Chlorella vulgaris* a partir de sensores à base de Arduino, particularmente os sensores de pH, temperatura, luminosidade e turbidez. Além disso, foi elaborada uma página na *web* para monitoramento em tempo real.

1.1. OBJETIVOS

1.1.1. Objetivo Geral

Monitoramento remoto de baixo custo à base de Arduino aplicado ao cultivo de microalgas.

1.1.2. Objetivos Específicos

- Desenvolver um sistema com sensores de pH, temperatura, turbidez e luminosidade à base de Arduino aplicados ao cultivo da microalga *C. vulgaris*;
- Elaborar uma página na *web* para monitoramento em tempo real do cultivo da microalga *C. vulgaris*;
- Comparar os resultados obtidos com sensores tradicionais.

2. REVISÃO BIBLIOGRÁFICA

A pesquisa bibliométrica no banco de dados Scopus (2022) utilizando os termos "Arduino" e "*Chlorella vulgaris*", resultou em apenas 2 documentos:

- "An Arduino UNO based biosensor for water pollution monitoring immobilized algae *Chlorella vulgaris*"
    - Ano: 2017;
    - Autores: Lazuardi Umar, Rahmondia Nanda Setiadi, Yanuar Hamzah, Tetty Marta Linda;
    - Objetivo do estudo: analisar a quantidade de oxigênio dissolvido no cultivo da *Chlorella vulgaris*.
- "A guide to Open-JP, a low cost open-source chlorophyll fluorometer"
    - Ano: 2019;
    - Autores: Harvey Bates, Alonso Zavafer, Milán Szabó, Peter J. Ralph;
    - Objetivo do estudo: utilizar um fluorímetro de Arduino para medir a fluorescência da clorofila *a* da *Chlorella vulgaris*.

Além disso, quando as palavras utilizadas foram "Internet of things" e "*Chlorella vulgaris*", a pesquisa resultou em apenas 1 documento:

- "Characterization of biomass pellets from *Chlorella vulgaris* microalgal production using industrial wastewater"
    - Ano: 2017;
    - Autores: Arys Carrasquilla-Batista, Alfonso Chacón-Rodríguez, Francinie Murillo-Vega, Kattia Núñez-Montero, Olman Gómez-Espinoza, Maritza Guerrero-Barrantes;
    - Objetivo do estudo: cultivar *Chlorella vulgaris* em laboratório, com acompanhamento de dispositivo eletrônico para envio de informações de luminosidade, temperatura, pH, condutividade e oxigênio dissolvido para a nuvem.

Portanto é notória a escassez de dados relacionando IoT e *C. vulgaris*.

2.1. MICROALGA

2.1.1. *Chlorella vulgaris*

Chlorella spp. são microalgas fotossintetizantes protistas amplamente investigadas (e.g. cultivo, composição, etc), que destacam-se pela intensa coloração verde. Normalmente são encontradas em águas doces ricas em nutrientes ou no solo (BOROWITZKA, 2018).

Atualmente existem 97 espécies de *Chlorella*, porém apenas 43 são reconhecidas e referendadas taxonomicamente (GUIRY e GUIRY, 2022).

A *C. vulgaris* é representada por células imóveis e esféricas, com diâmetro entre 2 e 10 µm. As células de *C. vulgaris* podem estar dipostas individualmente ou em colônias de até 64 células (SAFI *et al.*, 2014). A sua reprodução ocorre assexuadamente por mitose. Assim, a célula-parental gera de 2 a 32 células-filhas no seu interior, que quando amadurecidas, são liberadas pelo rompimento da parede celular da célula-mãe. Os restos da célula-parental são consumidos pelas células-filhas recém-formadas durante a auto-espuração (YAMAMOTO *et al.*, 2003; YAMAMOTO *et al.*, 2004; SAFI *et al.*, 2014).

Em relação ao metabolismo energético central, a *C. vulgaris* é extremamente versátil, visto que pode ser autotrófico, heterotrófico ou mixotrófico. É válido mencionar que o cultivo de microalgas mixotróficas reduz significativamente o tempo de geração celular, ou seja, aumenta a produtividade da biomassa microalgal (VIDOTTI *et al.*, 2020).

2.1.2. Influência dos fatores associados ao crescimento, multiplicação e composição bioquímica da biomassa de *Chlorella vulgaris*

As microalgas são capazes de se adaptar em bruscas variações ambientais, porém a concentração, bem como presença ou ausência de elementos nutricionais pode limitar o crescimento destas. Fatores externos como luminosidade, temperatura, dióxido de carbono e pH, bem como macroelementos como nitrogênio, potássio e enxofre são tradicionamente significativos no crescimento, multiplicação e composição química das microalgas (VÍTOVÁ *et al.*, 2015).

Em relação às fontes de macroelementos, elas podem ser categorizadas em primárias ou secundárias, sendo que as primárias atuam diretamente nas atividades fundamentais das microalgas, como crescimento, fotossíntese e

respiração, por outro lado as secundárias podem auxiliar nos sinais simbióticos e, muitas vezes, aumentam a diversidade química da biomassa microalgal, aumentando assim o potencial de aplicação destas (GUEDES *et al.*, 2011; SANTHOSH *et al.*, 2016).

Neste sentido, a adequada intensidade luminosa favorece a produção de carboidratos e lipídios, em especial amido e triglicerídeos, por outro lado altas intensidades de luz causam fotoinibição enquanto que, a ausência ou baixa intensidade de luz inibe o crescimento microalgal (BREUER *et al.*, 2013; VÍTOVÁ *et al.*, 2015; MINHAS *et al.*, 2016).

É válido notar que a intensidade de luz, bem como o fotoperíodo (claro:escuro) são essenciais ao crescimento microalgal. A *C. vulgaris* necessita de um fotoperíodo de 12:12 para maximizar o seu rendimento, que, além de favorecer os processos bioquímicos envolvendo as células, reduz os custos relacionados ao uso de energia. Contudo, durante o período escuro ocorre a degradação de amido, com perdas de 10 a 20% (BRÁNYIKOVÁ *et al.*, 2011; KHOEYI *et al.*, 2012; RATOMSKI e HAWROT-PAW, 2021).

A temperatura também é fundamentalmente determinante na composição bioquímica da microalga cultivada, visto que tanto a fotossíntese quanto os processos respiratórios microalgais são dependentes da temperatura. Segundo Barghbani *et al.* (2012) a temperatura ótima de cultivo é de 30 °C, com a maior produção de biomassa. Em geral, temperaturas abaixo da ideal são responsáveis pela maior concentração lipídica (BREUER *et al.*, 2013; MORAIS *et al.*, 2015; VÍTOVÁ *et al.*, 2015).

O ajuste do pH de cultivo microalgal é importante, visto que favorece o crescimento celular, aumenta a absorção de nutrientes e a biossíntese de metabólitos produzidos, bem como reduz as chances de contaminação (MORAIS *et al.*, 2015; RU *et al.*, 2020). Segundo Ratomski e Hawrot-Paw (2021), o pH neutro (7) é definido como ótimo para o crescimento de microalgas.

Portanto, a determinação e controle dos fatores ambientais e as fontes de carbono são fundamentais para a produtividade do cultivo de microalgas.

2.2. SISTEMAS DE CONTROLE BIOTECNOLÓGICOS DE BAIXO CUSTO

2.2.1. Internet das coisas (IoT)

Segundo Gillis (2020), IoT é um sistema que interliga dispositivos mecânicos ou digitais, fornecendo um identificador único, a fim de garantir a transferência de dados, sem a necessidade de interações humano-humano ou humano-máquina. O ecossistema IoT apresenta um imenso potencial de aplicação, permitindo a automatização de processos e o maior controle destes. Possibilita acessar informações em tempo real de qualquer lugar e momento através de dispositivos eletrônicos, economizando tempo e dinheiro. Para isso, utilizam-se dispositivos inteligentes, habilitados a se conectar com a rede de Internet, a fim de coletar e enviar dados à nuvem para serem analisados.

A integração entre nuvem e IoT é extremamente vantajosa. A IoT se beneficia dos recursos virtuais ilimitados da nuvem, como armazenamento, processamento e comunicação, permitindo maior agilidade na coleta, distribuição e tratamento de dados, com eficácia, escalabilidade e baixos custos. Enquanto a nuvem amplia sua área de atuação, sendo capaz de se relacionar com elementos do mundo real de maneira mais distribuída e dinâmica (SHARMA e KANTHA, 2020).

Entretanto, tamanha conectividade pode facilitar o acesso de informações confidenciais ou até a adulteração destas, além de haver probabilidade de *bugs* que cessem o envio de informações e que corrompam os dispositivos conectados. Portanto, a segurança e privacidade são as principais fragilidades desse sistema, que devem ser cuidadosamente supervisionadas a fim de minimizar as vulnerabilidades e aumentar a segurança do sistema (GILLIS, 2020).

2.2.2. Arduino

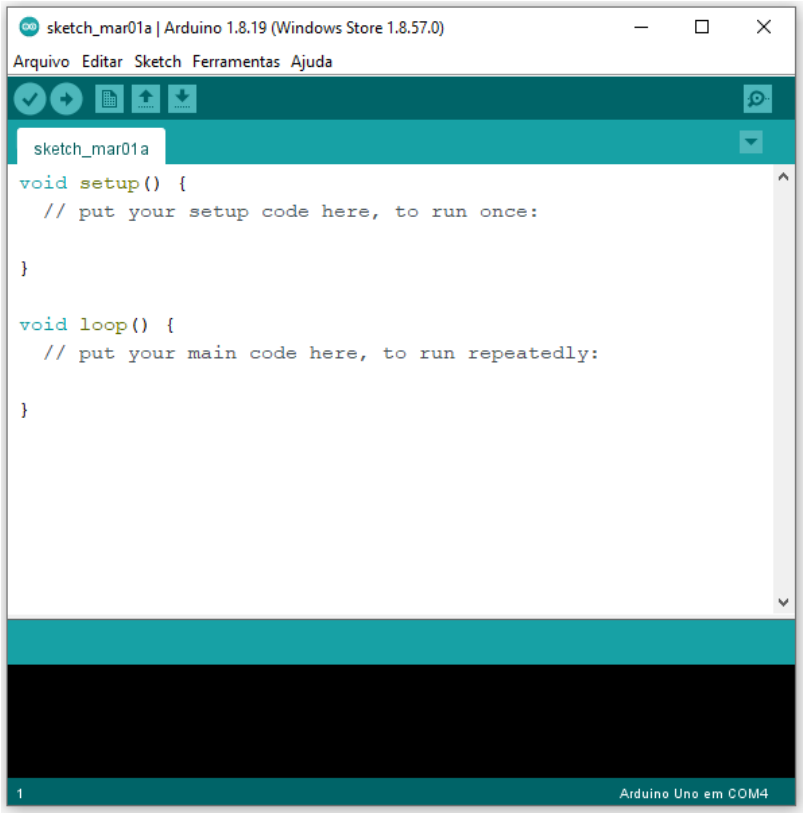
O Arduino é uma plataforma eletrônica de código aberto criada em 2005 com a finalidade de facilitar e reduzir o custo de dispositivos e sensores que interagem com o ambiente. As placas Arduino são capazes de ler entradas dos seus sensores e transformá-las em respostas de saída. Para isso, utilizam-se as linguagens de programação Arduino, baseada na conexão de fios, e do *software* Arduino IDE, com a linguagem C ou C++ (Arduino.cc, 2022; LOUIS, 2016).

O ambiente de desenvolvimento integrado do Arduino (Arduino IDE) apresenta diversas vantagens de uso, incluindo processamento rápido, interface de fácil entendimento, acesso gratuito e facilidade ao gravar os códigos na placa, bastando um cabo de carregamento USB (LOUIS, 2016).

O *software* é dividido em duas partes principais: as funções void setup() e void loop(). A primeira é chamada apenas uma vez, sempre que o *sketch* inicia, iniciando a velocidade do monitor serial, apresentando a inicialização das variáveis e das bibliotecas, bem como determinando os pinos como dispositivos de entrada e saída. A segunda função trabalha com os valores pré-inicializados, repetindo-se consecutivamente com um retardo (*delay*) enquanto a placa Arduino está conectada à fonte de alimentação (Arduino.cc, 2022).

Arduino UNO é a placa mais popular da marca. Esta detém 14 pinos digitais de entrada e saída, 6 entradas analógicas, 1 ressonador cerâmico de 16 MHz, 1 conexão USB, 1 conector de alimentação, 1 conector ICSP e 1 botão de reset. Além de possuir memória EEPROM ATmega328P, que permite com que os códigos salvos não sejam apagados quando a placa é removida da fonte de alimentação (Arduino.cc, 2022). Essa plataforma microcontroladora é usualmente escolhida devido aos seus benefícios de baixo custo, fácil manuseio e escalabilidade, programação integrada e ao seu baixo consumo de energia (MONK, 2013).

Figura 1 - Ambiente de desenvolvimento integrado do Arduino (Arduino IDE)



Fonte: A autora.

Como descrito, o Arduino UNO apresenta portas analógicas e digitais. As portas analógicas fornecem um valor entre 0 e 1023, que é correspondente à tensão existente em determinado pino naquele momento, isso porque o Arduino utiliza um conversor analógico digital para leitura e este trabalha com 10 bits (2^{10} = 1024). Já as portas digitais apenas informam se a entrada está ligada ou desligada, de acordo com a tensão de entrada de determinado pino. Este trabalha com sistema binário, considerando a porta ligada (1) em tensões superiores a 2,5 V e desligada (0) em tensões inferiores a 2,5 V (MONK, 2013; VIDAL, 2019).

2.2.2.1. Sensores de Arduino

Segundo Banzi e Shiloh (2015), os sensores são dispositivos eletrônicos capazes de converter medidas do mundo real em sinais elétricos, possibilitando o monitoramento dos dados por meio da interação meio-máquina.

2.2.2.1.1. Sensor de luminosidade

O LDR é um resistor dependente da luz, ou seja, ao receber um grande pacote de fótons oriundos da luz incidente, o sensor atua absorvendo os elétrons, que melhoram a sua condutividade e, consequentemente, reduzem a sua resistência. Assim, quanto maior a luminosidade, menor a resistência do sensor. É importante salientar que o LDR não é capaz de informar a quantidade de luz de um sistema (Lux), apenas se o ambiente está claro ou escuro (VIDAL, 2019).

2.2.2.1.2. Sensor de temperatura

Sensores de temperatura são dispositivos capazes de verificar o grau de aquecimento de determinado sistema. Em geral, os sensores de Arduino são conectados através de portas analógicas, para decodificação do sinal, porém o sensor de temperatura DS18B20 é um dispositivo externo ao Arduino que funciona com tensões de alimentação entre 3 e 5 V e é capaz de ler a temperatura do processo, interpretá-la e enviá-la para o dispositivo Arduino, através da conexão com uma porta digital (MADEIRA, 2018).

Como o sensor DS18B20 atua com o sistema 1 Wire, para a correta funcionalidade do código do ambiente Arduino IDE é necessária a adição das bibliotecas OneWire.h e DallasTemperature.h, ambas desenvolvidas pela empresa Dallas Semiconductor, para criar uma comunicação em barramento que permite com que em uma única linha de transmissão de dados, vários dispositivos sejam conectados (MADEIRA, 2018).

2.2.2.1.3. Sensor de pH

Sensores de pH são capazes de medir o potencial hidrogeniônico da água. O sensor de pH PH4502C com eletrodo BNC foi elaborado especificamente para conexão com a entrada analógica do Arduino. Este vem embebido em solução de KCl (3M) e a sua calibração pode ser realizada utilizando essa solução. Segundo a literatura, KCl (3M) apresenta um pH igual a 7, sendo assim, basta alterar a variável de calibração diretamente no código a fim de obter 7 como valor de saída. Cabe ressaltar que a tensão do sistema pode variar entre 0 e 5 V, já o pH pode variar entre 0 e 14. Sendo assim, o pH 7 (valor intermediário) equivale à tensão de 2,5 V no sistema (DFRobot, 2022; SUHANKO, 2021).

2.2.2.1.4. Sensor de turbidez

A turbidez se refere à suspensão de partículas, representada por sedimentos, sujidades ou até biomassa. O sensor de turbidez LGZD para Arduino possui um sistema óptico na sua extremidade inferior, formado por um LED emissor de luz infravermelho e um fototransistor receptor de luz. Esse sistema é capaz de medir a transmitância e a taxa de dispersão da luz emitida, enviando um valor de voltagem como resposta. Dessa forma, quanto maior a quantidade de sólidos suspensos ou quanto mais densa a cultura celular, menor o valor resultante, que varia entre 0 e 1023 (NGUYEN e RITTMANN, 2018; STRAUB, 2020).

2.2.2.1.5. Conectividade à internet

Com o intuito de garantir a comunicação entre hardware e *software*, ou

seja, de transferir os dados coletados pelos sensores de Arduino para a nuvem, é necessário utilizar um microcontrolador de Arduino, o NodeMCU ESP8266. Esse processador permite a conexão do sistema Arduino ao WiFi por um preço acessível. É importante salientar que o NodeMCU trabalha na tensão de 3,3 V, ou seja, a voltagem de 5 V que é de praxe ser usada em trabalhos com Arduino deve ser evitada. Para conexão do NodeMCU à *protoboard*, é necessário utilizar o ESP-01, uma placa de 8 pinos que é capaz de converter a tensão de 5 V, oriunda do arduino, em 3,3 V (KOLBAN, 2016).

2.2.2.2. Arduino aplicado a processos microalgais

Nos últimos anos o Arduino vem sendo aplicado - em certos casos associado a IoT - em cultivos microalgais, mais especificamente no monitoramento do crescimento celular, de forma a maximizar o processo biotecnológico (WANG *et al.*, 2022).

Nguyen e Rittmann (2018) monitoraram a concentração de biomassa no cultivo da microalga *Synechocystis sp*. em fotobiorreator (PBR) com o auxílio de um sensor de turbidez TSD-10 associado ao Arduino Mega. Para isso, utilizaram uma bomba peristáltica, a fim de coletar as amostras para medição de turbidez, sem a formação de bolhas e de headspace. A calibração do sensor foi realizada por meio da correlação entre a absorbância 730 nm (espectrofotômetro comercial) e a voltagem do sensor. O seguinte coeficiente de determinação foi obtido: R^2 de 0,95. Dessa forma, o turbidímetro utilizado demonstrou confiabilidade.

Flores *et al.* (2020) construíram um sensor de turbidez eficaz associado ao Arduino. Para isso, mantiveram um tubo de borossilicato dentro de uma câmera isolada de luz, a fim de fazer com que a solução microalgal de *Spirulina platensis* fluísse por ele, onde seria possível medir a concentração de biomassa algal. Assim, os autores utilizaram um sistema (formado por um sensor de luz TEMT600 e um LED) capaz de captar a quantidade de luz que permeava o fluido, semelhante a um espectrofotômetro.

Ariawan e Makalew (2018) monitoraram o cultivo de *Spirulina spp.* por meio do Arduino e IoT para análise dos dados em tempo real. Foi utilizado o sensor de temperatura DS18B20, em caso de temperaturas superiores a 30°C um ventilador era ligado automaticamente. Além do uso do sensor de intensidade ultravioleta ML8511,

que foi capaz de relacionar linearmente a intensidade UV com a leitura de tensão do sensor.

Tham *et al.* (2022) implementaram sensores Arduino de temperatura, pH e nível de água no cultivo de *Spirulina platensis*, conectando-os ao Blynk - um aplicativo capaz de monitorar os dados gerados pelos sensores - por meio do NodeMCU. Os autores comprovaram alta precisão dos equipamentos utilizados, à exceção do sensor de nível de líquido.

O cultivo de microalgas envolve diversos parâmetros que exigem monitoramento. No entanto, há poucas evidências na literatura sobre o uso de Arduino e a implementação da IoT em processos biotecnológicos com a *Chlorella*. Demonstrando assim uma alternativa em ascensão nesta área de pesquisa.

2.3. APLICAÇÃO *WEB*

Os serviços *web* têm se tornado cada vez mais corriqueiros. Eles são formados pela combinação entre *frontend* e *backend*. O primeiro é responsável pela interação com o usuário, apresentando gráficos, formulários, botões e outros tipos de comunicações, em geral, desenvolvido por uma combinação de linguagens, como HTML, CSS e JavaScript. Por outro lado, o *backend* corresponde à parte interna da aplicação, normalmente formado por servidor, aplicativo e banco de dados, que garante o armazenamento de dados na nuvem (ABDULLAH e ZEKI, 2014).

2.3.1. Programas utilizados

O JavaScript é uma linguagem de alto nível para criação de páginas *web* e *softwares*. O JavaScript destaca-se por ser a linguagem de programação mais utilizada atualmente. No *frontend* permite a adição de funcionalidades complexas, como atualizações em tempo real, animações e gráficos interativos. Enquanto no *backend* auxilia no processamento de informações do banco de dados, com o auxílio do Node.js (Mozilla, 2022; NOLETO, 2022).

Node.js é um ambiente multiplataforma que concede a criação do *backend* de aplicações com JavaScript. O Node é bastante vantajoso por permitir o desenvolvimento de todo o servidor com a linguagem JavaScript, não sendo necessário alterá-la para publicação na *web*, garantindo maior escalabilidade (Mozilla,

2022).

Na construção do *frontend*, *frameworks* podem ser utilizados para aperfeiçoar a aplicação, como é o caso do React.js, um *framework* desenvolvido pelo Facebook que permite usar os recursos JavaScript na construção de interfaces de usuário (React, 2022) e o Apache EChartsTM, um *framework* de visualização JavaScript que possibilita a criação de gráficos interativos, personalizáveis e altamente responsivos (ECharts, 2022).

Enquanto para o *backend* destaca-se o Express, um *framework* para o Node.js, que fornece recursos para criação de aplicativos *web* em JavaScript. Ele é o mais utilizado, por proporcionar o gerenciamento de requisições HTTP e POST, além de permitir trabalhar com *cookies*, *login* de usuário, URL e seus parâmetros (Express, 2022; Mozilla, 2022).

O *Visual Studio Code* é um editor de códigos gratuito da Microsoft. Por possuir uma plataforma de código aberto que possui diversas extensões e suporta várias linguagens de programação, é o editor de código mais utilizado na atualidade (Visual Studio Code, 2022).

O GitHub é o maior *software* de desenvolvimento do mundo, detendo mais de 200 milhões de repositórios, por ser possível guardar códigos das mais variadas linguagens. Essa plataforma na nuvem tem como objetivo criar um ambiente colaborativo, acessível e gratuito para que os desenvolvedores aprendam e contribuam com códigos (GitHub, 2022).

2.3.2. Banco de dados

Banco de dados (BD) é um conjunto de informações/dados organizados e armazenados eletronicamente, que podem ser acessados, modificados e controlados facilmente (Oracle, 2022).

Existem diversos tipos de banco de dados, porém o mais popular é o BD relacional, que fornece acesso a pontos de dados que são relacionados entre si por meio de tabelas, facilitando o acesso às informações. Em geral, em BD relacionais utiliza-se SQL (Linguagem de Consulta Estruturada), uma linguagem padrão para realizar queries, ou seja, comandos que ao serem executados retornam informações armazenadas nos banco de dados (CLEMENTE, 2019; SILVEIRA, 2019; Oracle, 2022).

O PostgreSQL é um banco de dados relacional altamente popular que protege a integridade e segurança dos dados armazenados, de uso gratuito, de código aberto e que utiliza a linguagem SQL (PostgreSQL Global Development Group, 2022).

2.3.3. Heroku

Ao finalizar os códigos da aplicação é necessário publicá-la na *web*, para viabilizar o acesso das pessoas a ela. O Heroku é uma plataforma na nuvem que permite aos desenvolvedores implantar, gerenciar e dimensionar aplicações de maneira simples e gratuita, além de aceitar diversas linguagens de programação e garantir segurança dos dados (Capterra, 2022).

Usualmente, associa-se ao Heroku um repositório do GitHub a fim de construir o aplicativo, assim, é possível gerenciar diretamente o código-fonte no GitHub, com atualização instantânea no Heroku. Os Buildpacks são responsáveis pela criação de slugs, ou seja, pacotes contendo o código-fonte e as suas dependências (linguagem, estrutura, tempo de execução) montados e compilados, prontos para execução (Heroku Dev Center, 2022).

O PostgreSQL pode ser acessado diretamente pelo Heroku, por meio do Heroku Postgres. Basta baixar um *driver* PostgreSQL e informar ao Heroku a variável DATABASE_URL criada no aplicativo PostgreSQL, ou seja, a URL que permite localizar o aplicativo para que este possa ser acessado e executado. O Heroku Postgres limita o uso de até 10 mil linhas de banco de dados (Heroku Dev Center, 2022).

Com o intuito de fazer a leitura do BD, é conveniente o uso do *framework* knex.js, um leitor de Node.js que é compatível com o banco de dados PostgreSQL (Knex.js, 2022).

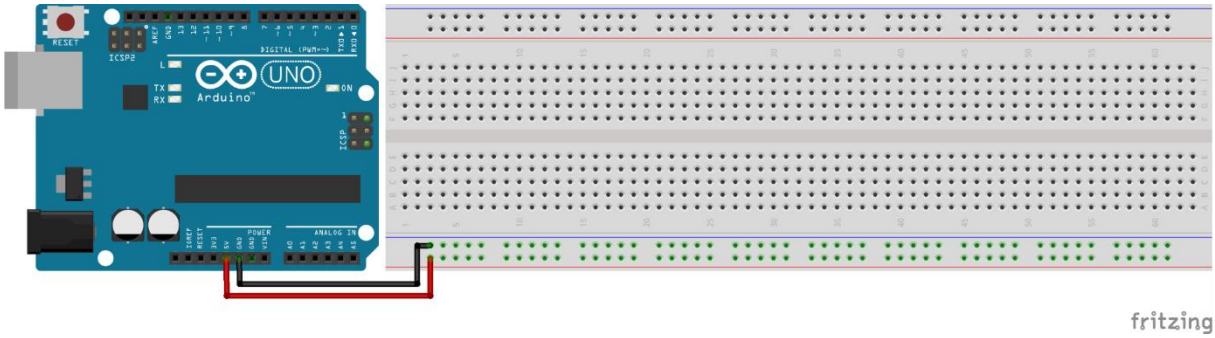
3. MATERIAL E MÉTODOS

3.1. DESENVOLVIMENTO DE SENSORES DE BAIXO CUSTO PARA O MONITORAMENTO DO CRESCIMENTO DA MICROALGA *C. vulgaris*

3.1.1. Arduino

A base do sistema de baixo custo foi feita pela ligação de um microcontrolador Arduino UNO (R3 ATMEGA328) a uma *protoboard* (830 Pontos MB-102) por meio de *jumpers* de ligação (macho-macho) nas entradas 5 V (Figura 2 - fio vermelho) e GND (Figura 2 - fio preto).

Figura 2 - Conexão básica entre Arduino e *protoboard*



Fonte: A autora.

O sistema foi energizado por meio de um conector USB, que acompanha o Arduino. Este pode ser conectado a um computador ou diretamente na tomada.

Os sensores de Arduino foram conectados a essa base do sistema, são eles: o sensor de luminosidade fotoresistor LDR (Figura 3a), o sensor de temperatura DS18B20 à prova d'água de 2 metros (Figura 3b), o sensor de pH PH4502C com eletrodo BNC (Figura 3c) e o sensor de turbidez para monitoramento de água LGZD (Figura 3d).

O LDR foi conectado a um sistema de divisão de tensão. Dessa forma, há uma entrada de tensão de 5 V e, ao final, uma conexão à terra (GND). A leitura do sensor de luminosidade foi mantida na entrada analógica A0 do Arduino, em paralelo com um resistor de 10 kΩ.

Figura 3 - Sensores de Arduino

Fonte: Saravati, 2022.

O sistema do termistor foi montado seguindo a premissa de que a resistência do sensor causaria uma queda de tensão e esta seria lida pela entrada digital 8. Assim, foi adicionado um resistor de 4,7 kΩ e o sistema foi conectado às portas de entrada e saída de 5 V e GND, respectivamente.

O pHmetro foi conectado de maneira simples, apenas pela ligação na fonte de 5 V de alimentação, no terra (GND) e na porta analógica A1 de leitura do Arduino.

Por fim, o turbidímetro foi conectado de maneira semelhante ao sensor de pH. Sendo alimentado por 5V e aterrado no GND, com porta analógica A2 de leitura do Arduino.

Foi utilizado o ambiente de desenvolvimento integrado Arduino IDE na versão Arduino 1.8.19.

Os sensores de luz, pH e turbidez, como conectados a entradas analógicas, retornam um valor entre 0 e 1023, como explicado na seção 2.2.2. Sendo assim, foi necessário fazer um cálculo de conversão para obter o valor correspondente de tensão (Equação 1). Caso a leitura da entrada analógica seja 0, será equivalente a 0 V, e quando a leitura for de 1024, será 5 V.

$$\text{voltagem} = \text{valorSensor} * \left(\frac{5\ \text{V}}{1024}\right) \quad (1)$$

As especificações técnicas dos sensores estão descritas no Anexo A.

3.1.2. Conectividade à internet

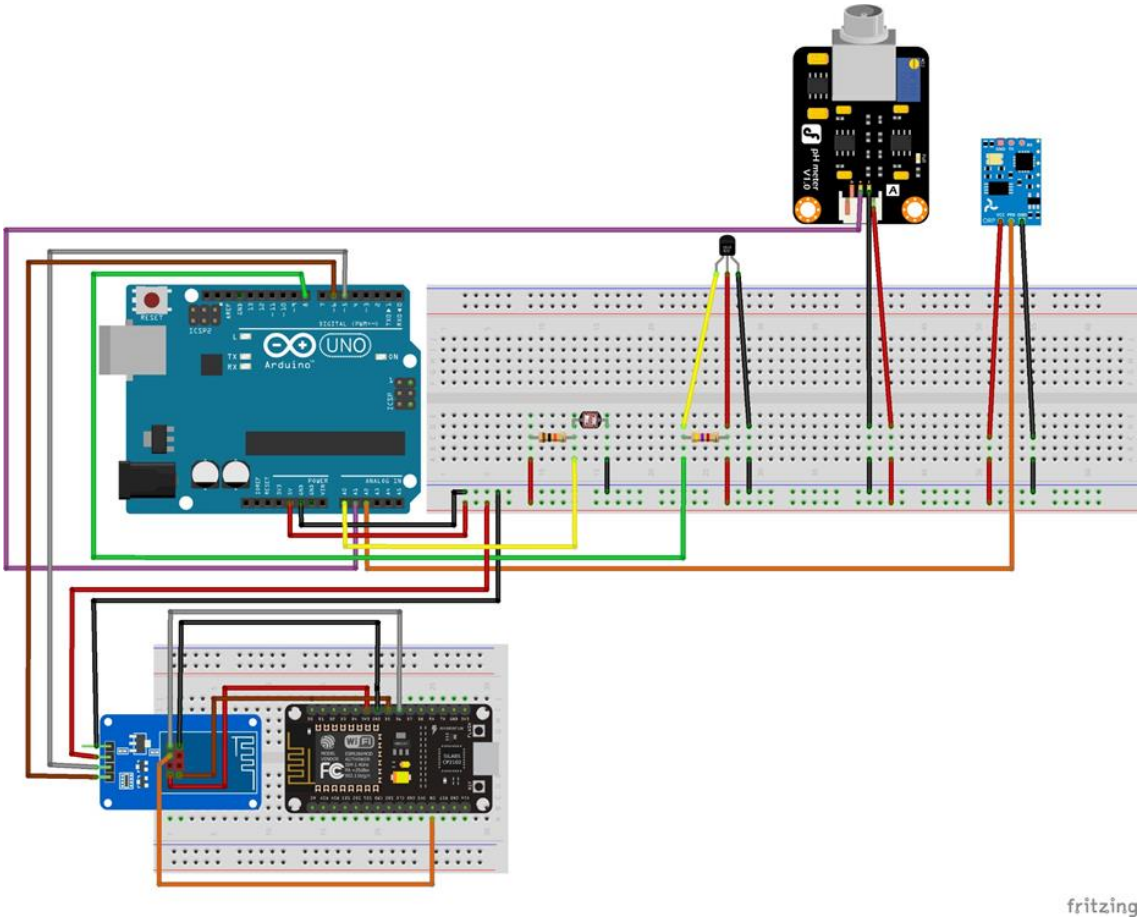
Com o intuito de enviar as informações captadas pelo Arduino e seus sensores para a nuvem, utilizou-se o módulo de WiFi NodeMCU ESP8266. O dispositivo necessita de uma tensão de alimentação de 3,3 V, por isso houve a adição

do ESP-01, um conversor de tensão de 5 para 3,3 V, que pode ser facilmente conectado ao sistema.

O NodeMCU foi conectado ao ESP-01 através das portas de alimentação (3 V), terra (GND), RX (receptor da comunicação), TX (transmissor da comunicação) e EN. Enquanto o ESP-01 foi vinculado à *protoboard* principal para alimentação de 5V e aterramento (GND), além de ser ligado diretamente ao Arduino nas portas digitais 5 e 6, que correspondem, respectivamente, aos terminais de recepção (RX) e transmissão (TX) da comunicação entre o sistema de Internet e o dispositivo Arduino. É importante salientar que a ligação de RX de um dispositivo sempre deve ser feita com a ligação TX do outro equipamento e vice-versa, a fim de garantir a correta comunicabilidade entre ambos.

Após todas as ligações explicitadas nas seções 3.1.1 e 3.1.2 deste trabalho, é possível garantirmos o nosso esquema final de ligação do sistema Arduino, como é mostrado na Figura 4.

Figura 4 - Esquema de ligação final do Arduino

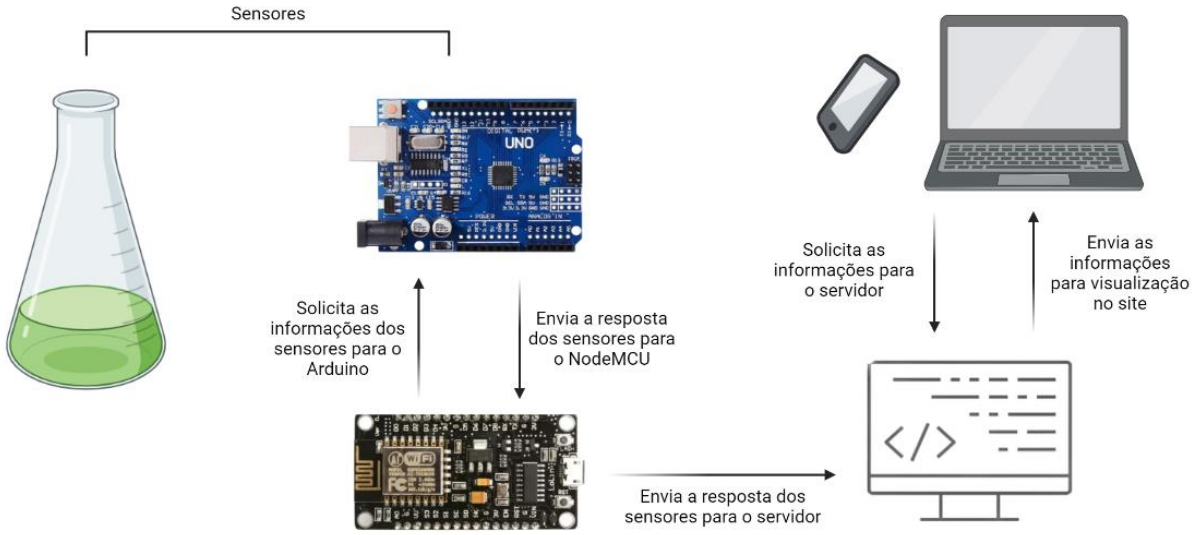


Fonte: A autora.

3.1.3. Aplicação *web*

O desenvolvimento do *frontend* da aplicação foi feito utilizando o Create React App, baseado no interpretador de código JavaScript (Node.js). A criação de gráficos no *frontend* foi feita a partir do *framework* Apache ECharts. O *backend* foi desenvolvido utilizando o Express do Node.js. Os códigos foram editados com o *Visual Studio Code* e ambos foram versionados e guardados em repositórios do GitHub. Por fim, a aplicação foi publicada no Heroku, que aponta a *branch master* dos repositórios da plataforma da Microsoft.

Figura 5 - Esquema prático



Fonte: A autora.

As tecnologias escolhidas são devido ao meu domínio nas linguagens JavaScript (aplicação) e C (Arduino).

3.2. AVALIAÇÃO DO CRESCIMENTO DA MICROALGA *C. vulgaris*

3.2.1. Microalga

A *C. vulgaris* utilizada foi doada pelo Laboratório Dempster do Departamento de Engenharia Química da Escola Politécnica da Universidade de São

Paulo (USP). A manutenção e o repique da microalga foram realizados em câmara de fluxo laminar (Veco CFLV-09) do Laboratório de Engenharia Biológica do Departamento de Engenharia Química e de Alimentos da Universidade Federal de Santa Catarina (UFSC).

3.2.2. Meio de cultura

O cultivo foi realizado em meio mineral WC, visto o seu popular uso como meio de cultivo para a *C. vulgaris*. Este foi preparado em laboratório pela adição de 1 mL de cada uma das seguintes substâncias: traços de metal, $MgSO_4$, Na_2SiO_3, $NaHCO_3$, $CaCl_2$, K_2HPO_4 e $NaNO_3$, em 1 litro de água destilada e autoclavada.

3.2.3. Inóculo

A microalga foi inicialmente cultivada em frascos Erlenmeyer de 125 mL, contendo meio mineral WC, com adição de antibióticos estreptomicina e ampicilina, ambos de concentração 5 g/L. Estes foram dispostos em *shaker* (Nova Ética 430), sob agitação de 100 rpm, temperatura de 35°C e aplicação de fotoperíodo de 12:12.

A preparação do inóculo foi realizada em câmara de fluxo laminar (Veco CFLV-09), previamente desinfetada com álcool 70% e UV por 15 minutos.

Figura 6 - Inóculo

Fonte: A autora.

3.2.4. Calibração dos sensores de pH e turbidez

O turbidímetro em questão funciona a uma tensão máxima de 4,2 V, ou seja, quando presente em água pura, apresenta esse valor máximo de voltagem. Para garantir isso, é necessário mergulhar o turbidímetro em água e criar um fator de calibração, fazendo com que o cálculo da voltagem retorne o valor de 4,2 V.

$$\text{voltagemTurbidez} = \text{sensorTurbidez} * \left(\frac{5\ \text{V}}{1024}\right) * \text{calibraçãoTurbidez} \qquad (2)$$

A partir da Equação 2 temos: “voltagemTurbidez” sendo o valor de 4,2 V, “sensorTurbidez” o valor de 0 a 1023 que o sensor retorna e “calibraçãoTurbidez” o valor de calibração calculado, que é igual a 1,129032258.

A calibração do sensor de turbidez ocorreu tendo a absorbância do espectrofotômetro (Bel Photonics 1105), comprimento de onda de 600 nm, como referência. Para isso, retirou-se uma alíquota de 10 mL da solução microalgal que foi diluída sucessivamente. Todas as diluições foram medidas em espectrofotômetro e no turbidímetro e, com os valores obtidos, elaborou-se uma curva.

3.2.5. Inoculação e sensores de Arduino

A inoculação foi realizada em 5 frascos Erlenmeyer de 1L previamente autoclavados (Phoenix AV-137) a 120 °C, numa pressão de 1 kgf/cm², por 15 minutos.

Ao primeiro Erlenmeyer adicionou-se apenas o meio de cultura (meio mineral WC), a fim de manter uma solução branco controle. Enquanto nos outros 4 frascos foram adicionados o meio de cultivo (meio mineral WC) e o inóculo nas proporções de 90% e 10%, respectivamente.

Dentre os frascos que continham a solução microalgal, foram inseridos sensores de Arduino separadamente, a fim de analisar a variação nos parâmetros durante o cultivo, são eles: os sensores de pH, temperatura e turbidez. O quarto frasco não conteve sensor a fim de manter uma solução padrão de controle.

Figura 7 - Soluções microalgais analisadas no presente trabalho

Fonte: A autora.

3.2.6. Cultivo

O ensaio foi realizado em *shaker* (Nova Ética 430). As amostras foram mantidas a 35 °C, 100 rpm e sob efeito de fotoperíodo de 12:12, por 7 dias. O sensor de luminosidade LDR foi fixado na parede interna do equipamento.

Figura 8 - Vista superior do local de cultivo

Fonte: A autora.

Durante o período de cultivo foram retirados, diariamente, 4 mL de cada frasco, a fim de analisar o crescimento da *C. vulgaris* através de equipamentos comerciais do laboratório.

A pesquisa iniciou por meio da análise das amostras em espectrofotômetro (Bel Photonics 1105), comprimento de onda de 600 nm, e em pHmetro (Kasvi ATC). Os valores foram utilizados para construir gráficos em função do tempo.

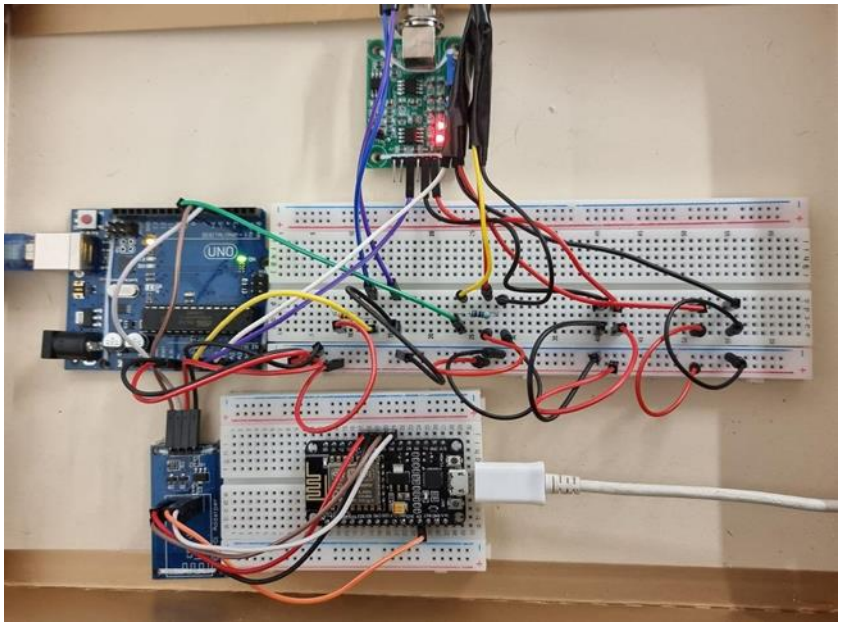
4. RESULTADOS E DISCUSSÃO

4.1. PROGRAMAÇÃO

4.1.1. Arduino

O estudo iniciou com as ligações de fios e sensores no microcontrolador Arduino e na *protoboard*, como descrito na Seção 3 (Material e Métodos). Seguido do desenvolvendo dos códigos do Arduino no *software* Arduino IDE. Estes estão apresentados nos Apêndices A (código de programação do Arduino) e B (código de programação do NodeMCU ESP8266).

Figura 9 - Conexão de fios no Arduino



Fonte: A autora.

Ao escrever o código para o Arduino, foram inicialmente definidas as bibliotecas externas à plataforma e as portas analógicas e digitais utilizadas, nomeando-as para facilitar a localização dos sensores.

A seguir, na função void setup() foi iniciado o monitor serial na velocidade de 9.600 bps, os terminais de recepção (RX) e transmissão (TX) de operação do Arduino e o sensor de temperatura. Além de configurar todos os sensores como dispositivos de entrada.

Na função void loop() foi requisitada a leitura dos sensores conectados ao sistema Arduino para realizar a conversão em valores de tensão, para assim calcular o valor das variáveis estudadas. Nessa função, também foi elaborado um objeto que armazena as rotas dos sensores de Arduino utilizados, possibilitando o envio dos valores, sempre que há conexão com a Internet. A função void loop() foi definida para se repetir a cada 10 segundos.

A seguir, foi desenvolvido o código do NodeMCU ESP8266, responsável por se conectar ao WiFi e enviar as rotas pré-estabelecidas no código do Arduino para a nuvem.

Inicialmente foi redigido um código para conexão direta ao WiFi. Para isso, foi necessário usar as bibliotecas ESP8266WiFi.h, ESP8266HTTPClient.h, WiFiClient.h e SoftwareSerial.h. Em seguida, foram redigidas as constantes de nome da rede (ssid) e senha (password) da WiFi, armazenando nelas as informações correspondentes, bem como uma *string* (server_url) com a URL do *backend* da aplicação.

Na função void setup() foi iniciado o monitor serial na velocidade de 9.600 bps e a biblioteca SoftwareSerial. Dessa forma, iniciou-se a conexão ao WiFi, utilizando os parâmetros (ssid e password) supracitados.

Na função void loop() foi criado uma estrutura condicional para que sempre que o NodeMCU estivesse conectado ao WiFi executasse os passos seguintes, são eles: a captação da resposta das rotas criadas no código principal do Arduino e a criação de uma *string* (server_path) responsável por levar as informações (temperatura, pH, turbidez e luminosidade) para a aplicação na nuvem. O envio de informações ocorre com o auxílio de um GET, um método HTTP pouco exigente derivado da biblioteca ESP8266HTTPClient. Assim, foi possível destinar os parâmetros (temperatura, pH, turbidez e luminosidade) para o *endpoint*, ou seja, para o *backend* da aplicação em questão, que está localizado no server_url. À função void loop() foi atribuído um *delay* de 20 minutos, fazendo com que a aplicação obtivesse menos de 10 mil linhas, o que é correspondente a limitação de linhas gratuitas no banco de dados.

Entretanto, o código não foi capaz de rodar na WiFi da Universidade, visto que funciona com autenticação, utilizando o modo de segurança: WPA2-Enterprise e autenticação da fase 2: PAP. Para ser capaz de conectar, o código necessitou passar por leves alterações, como a inclusão de arquivos em C próprios do ESP8266, são

eles: user_interface.h (definições para o sistema operacional e WiFi), wpa2_enterprise.h (permite acessar modo de segurança WPA2-Enterprise) e c_types.h (definições da linguagem C). Foi ainda necessário criar as constantes username[ ] e identity[ ], além das constantes que armazenavam nome da rede (ssid) e senha (password).

Com o sistema Arduino pronto, incluindo comunicação de fios e desenvolvimento de código, seguiu-se para a aplicação destes no cultivo microalgal.

4.1.2. Aplicação web

A aplicação foi elaborada utilizando o *Visual Studio Code*. Todos os códigos estão apresentados nos Apêndices C (código de programação do *backend*) e D (código de programação do *frontend*).

Inicialmente foi desenvolvido o *backend*. Para isso, foi instalado o Node por meio do comando “npm install” no Comander do computador.

O arquivo api.js foi criado e então efetuada a requisição do *framework* Express para fazer o gerenciamento da aplicação *web*, do routes.js para gerenciar os *endpoints* da aplicação, e, por último, a requisição do CORS para ajustar os protocolos de segurança dos *endpoints*. Essa última requisição foi feita, já que geralmente os navegadores impedem o acesso a recursos que não estão localizados na mesma origem, como um método de segurança para o *backend*, porém às vezes esse acesso é necessário, para isso utiliza-se o CORS (Compartilhamento de Recursos de Origem Cruzada), permitindo o acesso e garantindo a segurança. Em seguida, foi adicionada a função app.use(), onde situam-se as informações de segurança dos *endpoints*, descrevendo quem é permitido acessá-los. E, para finalizar, a aplicação foi requisitada a usar as rotas (routes.js), configurando a porta local (localhost) como sendo 3333 e solicitando ao Heroku a porta utilizada por ele.

No arquivo routes.js foi utilizado o router do Express e configurado as seguintes rotas: /data (para receber os dados do Arduino), /get-data (para solicitar os dados no *frontend*) e /drop-table (para limpar o banco de dados para um novo experimento). Todas essas rotas utilizam as funções presentes no arquivo DataController.js.

No arquivo DataController.js foi criada a função index(), que busca os dados na tabela (data_table) e envia de volta para quem solicita, acessando o

endpoint (get-data). Na função create(), o Arduino envia os parâmetros coletados pelos sensores, só aceitando se tiver todos os parâmetros solicitados, e depositando-os no data_table. Por fim, a função drop table() foi criada para apagar todos os dados contidos na tabela (data_table).

Finalizando os códigos do servidor, iniciou-se o desenvolvimento da interface. Para isso, foi criada uma nova pasta intitulada "*Frontend*" no *Visual Studio Code*.

Para começar, o componente Layout.jsx foi gerado com a função deleteData() que cria o botão para deletar os dados, além de apresentar a função toggleModal() que cria uma "*modal box*" perguntando se o usuário realmente deseja apagar os dados. Além disso, foi desenvolvido a navegação entre as páginas de início e contato, renderizando-as de acordo com a URL que estiver a aplicação, e elaborado o cabeçalho e o rodapé.

O Home.jsx também é um componente React.js que, ao ser renderizado, executa a função getData(), chamando os dados na tela para serem apresentados. Nele iniciam-se as funções recursivas (recursivefunction() e secondaryrecursivefunction()), ou seja, as funções que chamam a elas mesmas uma ou mais vezes, atuando com um *loop* inifinito com um *delay* de 50 segundos. Elas buscam os dados, atualizando o *frontend* da aplicação em questão. Todas as funções citadas no Home.jsx chamam funções do api.js que requisitam os dados. A seguir, foram elaboradas as constantes que contém o último valor coletado pelos sensores, além de configurações do HomeCard.jsx setadas à mão. Além disso, foi criada a função exportData(), que é executada ao clicar no botão de *download* do arquivo na aplicação, exportando o documento no formato CSV (Valores Separados por Vírgula) e foi criado um return() com as configurações de visualização do botão.

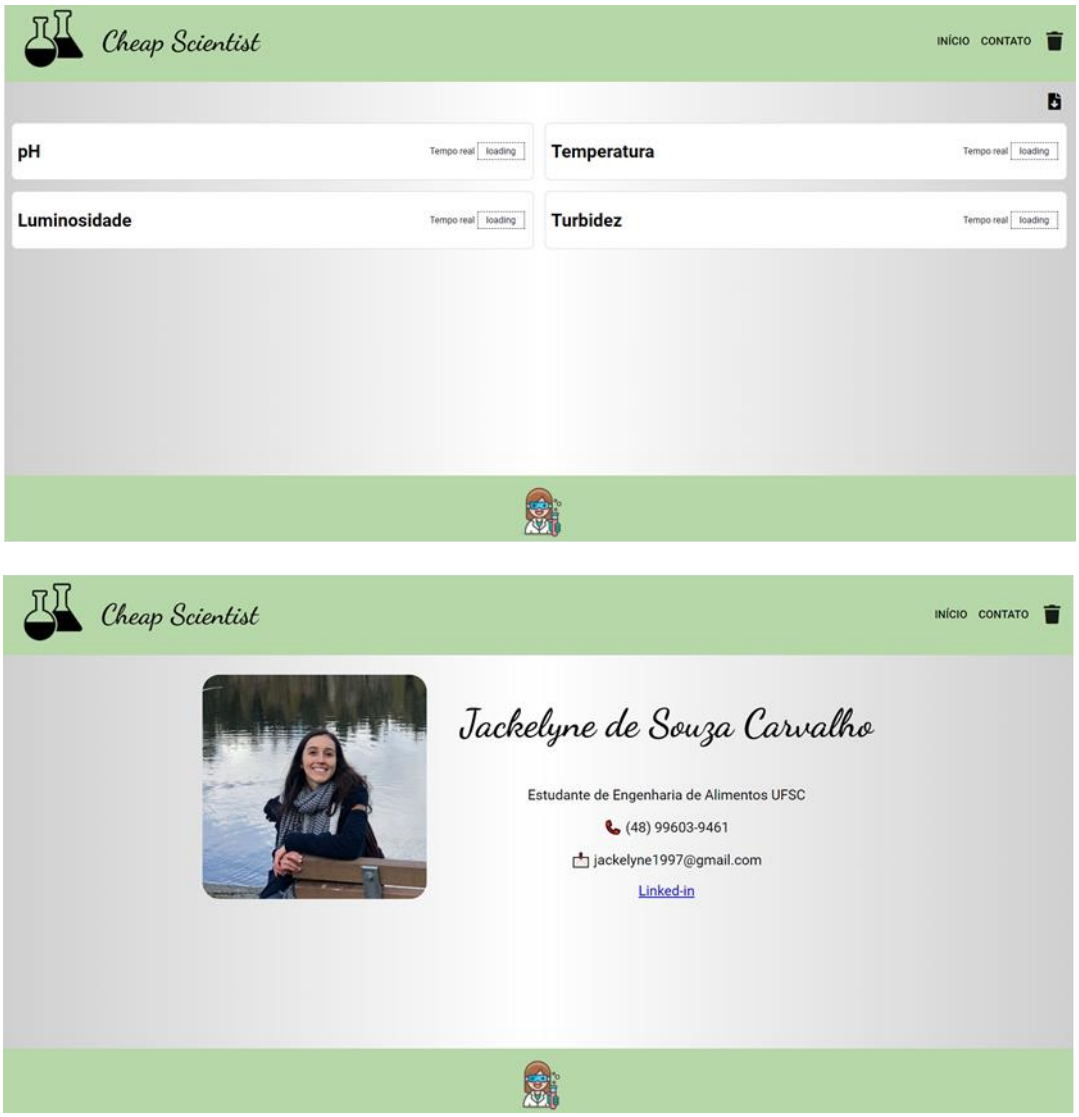
O HomeCard.jsx é um componente React.js, que recebe os dados da solicitação realizada ao *backend*, renderizando-os em gráficos, com o auxílio do *framework* ECharts. As configurações dos gráficos foram feitas na função getOption().

Em seguida, foi gerado o arquivo api.js, a fim de configurar as funções para acessar os *endpoints* a partir do *frontend*. A função getHomeData() foi criada para buscar os dados no *endpoint* para mostrar no *frontend*, enquanto a função dropTable(), como já explicado, derruba os dados coletados, para isso, a função chama o *endpoint*, que apaga definitivamente as informações do banco de dados.

Por último, o componente Contato.jsx foi criado, onde foi desenvolvido uma

página visual e estática com as minhas informações de contato, caso alguém tenha interesse em debater sobre a aplicação ou o experimento desenvolvido.

Figura 10 – Aparência final da aplicação *web*



Fonte: A autora.

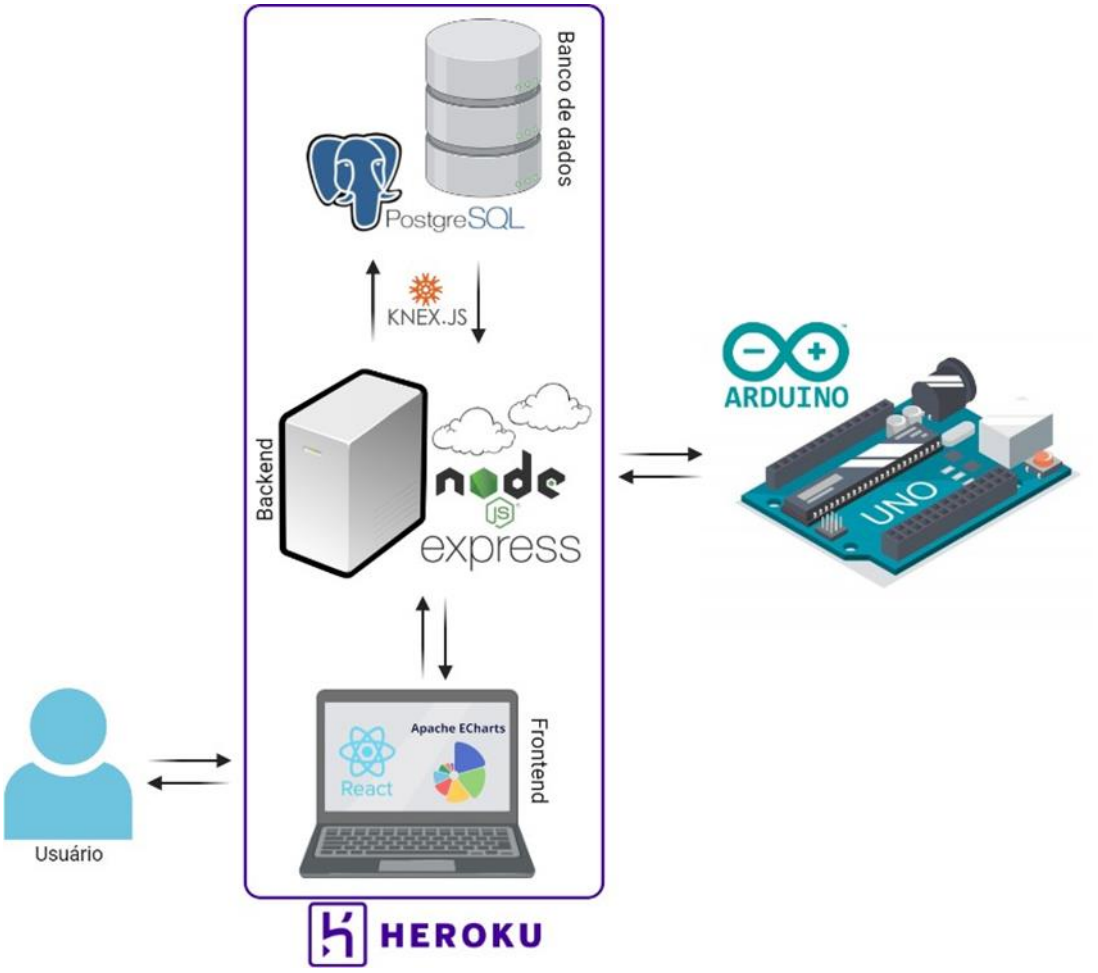
Criaram-se, então, 2 repositórios no GitHub a fim de armazenar os códigos do *backend* e *frontend* da aplicação e permitir o acesso do Heroku a eles. Com os códigos na nuvem, seguiu-se para a publicação destes.

Utilizou-se a plataforma do Heroku linkada ao GitHub para fazer o *deploy* automático do sistema *web*. Para isso, duas aplicações no Heroku foram criadas, uma para o *backend* e outra para o *frontend*, e executou-se o "*Deploy Branch master*".

Na aplicação *backend* do Heroku foi criado um banco de dados para a

aplicação. Para isso, utilizou-se o Heroku Postgres que, ao ser criado, gerou uma URL para a sua localização. Em seguida, as variáveis de configuração (*Config vars*) foram definidas, indicando parâmetros como a URL do *frontend*, do *backend* e do banco de dados do Postgres utilizado. Dessa forma, ocorreu a correta implementação da aplicação na nuvem.

Figura 11 - Desenvolvimento da aplicação: tecnologias e comunicações



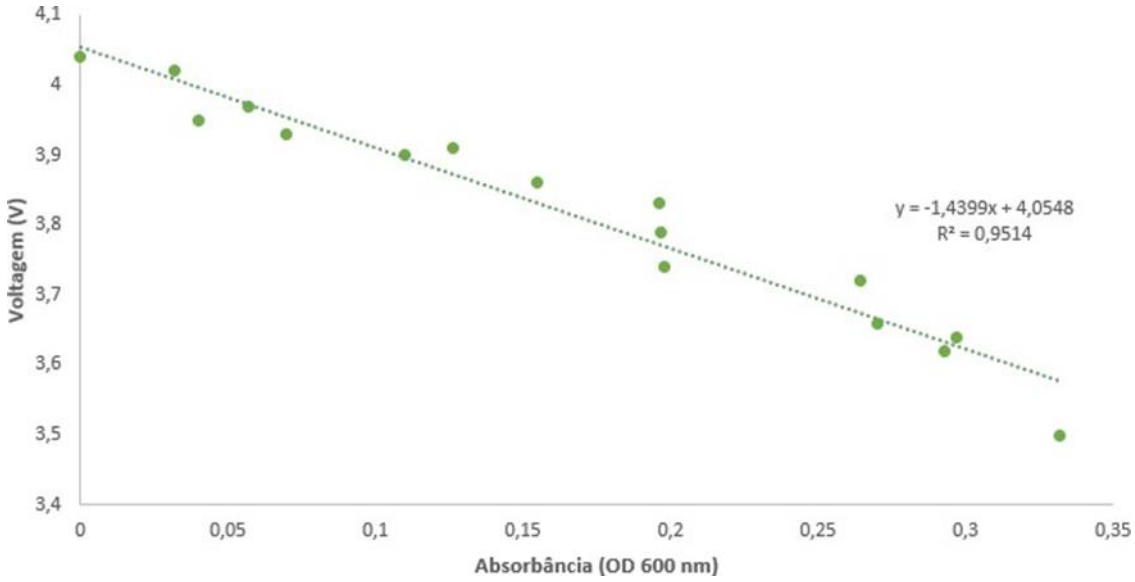
Fonte: A autora.

4.2. AJUSTE DA MICROALGA

Para iniciar os estudos relacionados à *Chlorella vulgaris*, construiu-se um gráfico relacionando a tensão (V) retornada pelo sensor de turbidez do Arduino com a absorbância medida (comprimento de onda de 600 nm), obtendo um coeficiente de determinação relativamente bom (R^2 = 0,9514) e provando um bom ajuste da curva construída. A equação da melhor reta foi adicionada ao código Arduino, de forma com

que a aplicação informasse os valores de absorbância, no lugar de voltagem.

Figura 12 - Relação entre voltagem do sensor de turbidez do Arduino e absorbância (600 nm) para *Chlorella vulgaris*



Fonte: A autora.

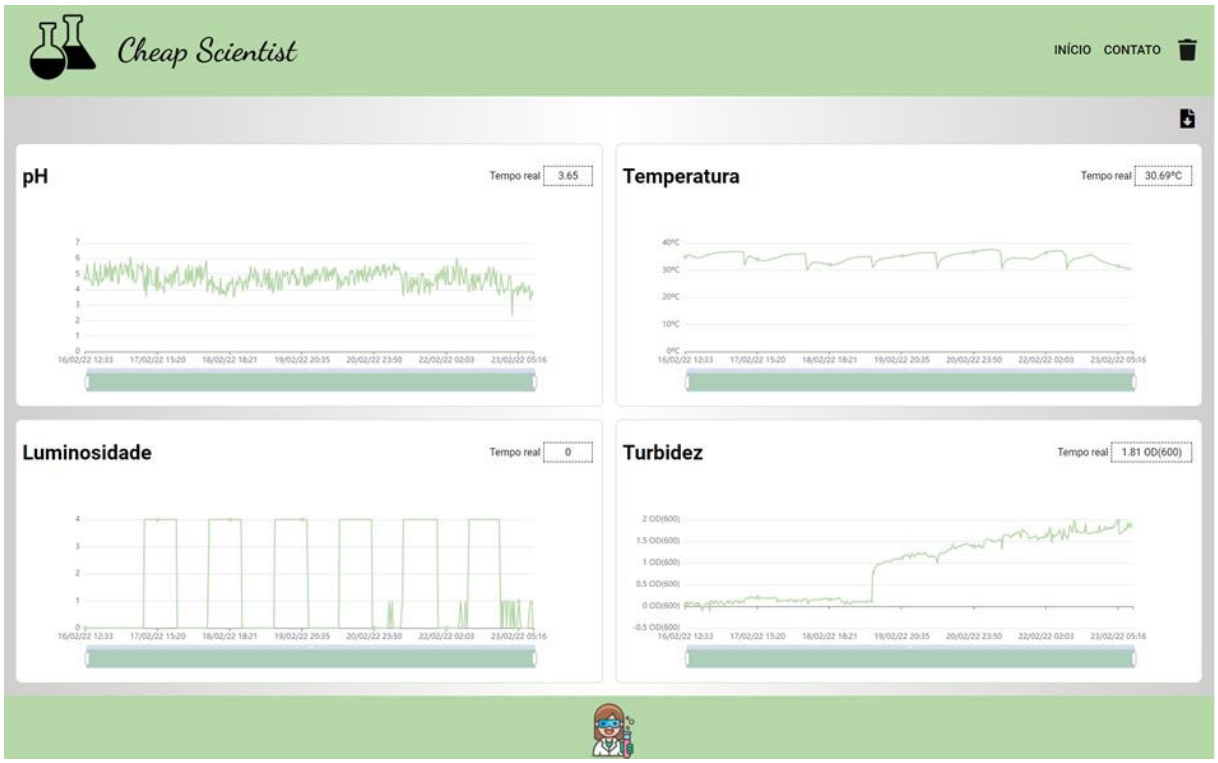
4.3. MONITORAMENTO DOS PARÂMETROS

O cultivo da microalga com a utilização dos sensores de Arduino foi realizado por 7 dias e acompanhado tanto pela aplicação *web* (Figura 13), quanto por coletas diárias para verificação de pH e absorbância (Figura 14).

A retirada dos 4 mL diários dos frascos de cultivo foi realizada dentro da cabine de fluxo laminar, dessa forma, os sensores precisavam ser desconectados do sistema Arduino. Para evitar o envio de valores que não condiziam com a realidade, pela desconexão de fios, sempre realizava a desenergização do microcontrolador.

Entretanto, como é possível observar na Figura 13, a temperatura apresentou picos mínimos diários, sempre condizentes com o horário de coleta. Isso porque o *shaker* era aberto para retirada e colocação dos frascos Erlenmeyer. Dessa forma, o equipamento sofria um abaixamento até uma temperatura próxima a 30 °C, levando um tempo para restabelecer a temperatura setada inicialmente. Para os outros parâmetros isso não foi notado.

Figura 13 - Acompanhamento do cultivo pela aplicação *web*



Fonte: A autora.

O pH (Figura 13) apresentou pontos bastante dispersos com o passar do período de crescimento, não havendo tanta linearidade quanto é observado com a temperatura. O menor valor captado foi de 2,32 e o maior foi de 6,13.

A luminosidade (Figura 13) apresentou bastante consistência nos valores fornecidos, com um valor mínimo de voltagem nos períodos de luminosidade alta e com máximo de voltagem nos períodos sem luz. É possível analisar que a partir do dia 21/02/2022 até o final do experimento iniciaram pequenos aumentos na voltagem, ou seja, diminuição no fornecimento de luz. O que pode ser explicado pelas quedas de energia que ocorreram nesses dias, afetando a fonte de iluminação do sistema.

Por fim, a turbidez, assim como o pH, apresentou notório desvio entre os pontos subsequentes. Nas primeiras 24 horas exibiu um aumento de 0,04 até aproximadamente 0,1 de absorbância (considerando o comprimento de onda de 600 nm). Contudo, no primeiro ponto após a primeira coleta, os valores dos sensores apresentaram um acréscimo, alcançando 0,2 de absorbância, demonstrando um inusitado comportamento. Esse fenômeno se tornou mais visível após a coleta do dia 19/02/2022, quando o valor saltou para 0,82. A partir desse ponto, a densidade óptica aumentou permanentemente, chegando a alcançar 1,98.

Outro fator a ser considerado no turbidímetro é a iluminação. Como o sensor mede a taxa de dispersão da luz emitida, o fotoperíodo deveria ter interferido nos valores de turbidez, entretanto, não foi visto relação entre os fatores em questão.

Em geral, a aplicação foi muito válida, pois permitiu o monitoramento do experimento distante fisicamente do laboratório. Além disso, foi possível notar a estabilidade de todo o sistema frente à quedas de energia, visto que este voltou a funcionar mesmo após os ocorridos. Ou seja, tanto os sensores voltaram a captar os dados, como o NodeMCU voltou a se conectar com os sistemas WiFi e Arduino, permitindo essa troca de informações.

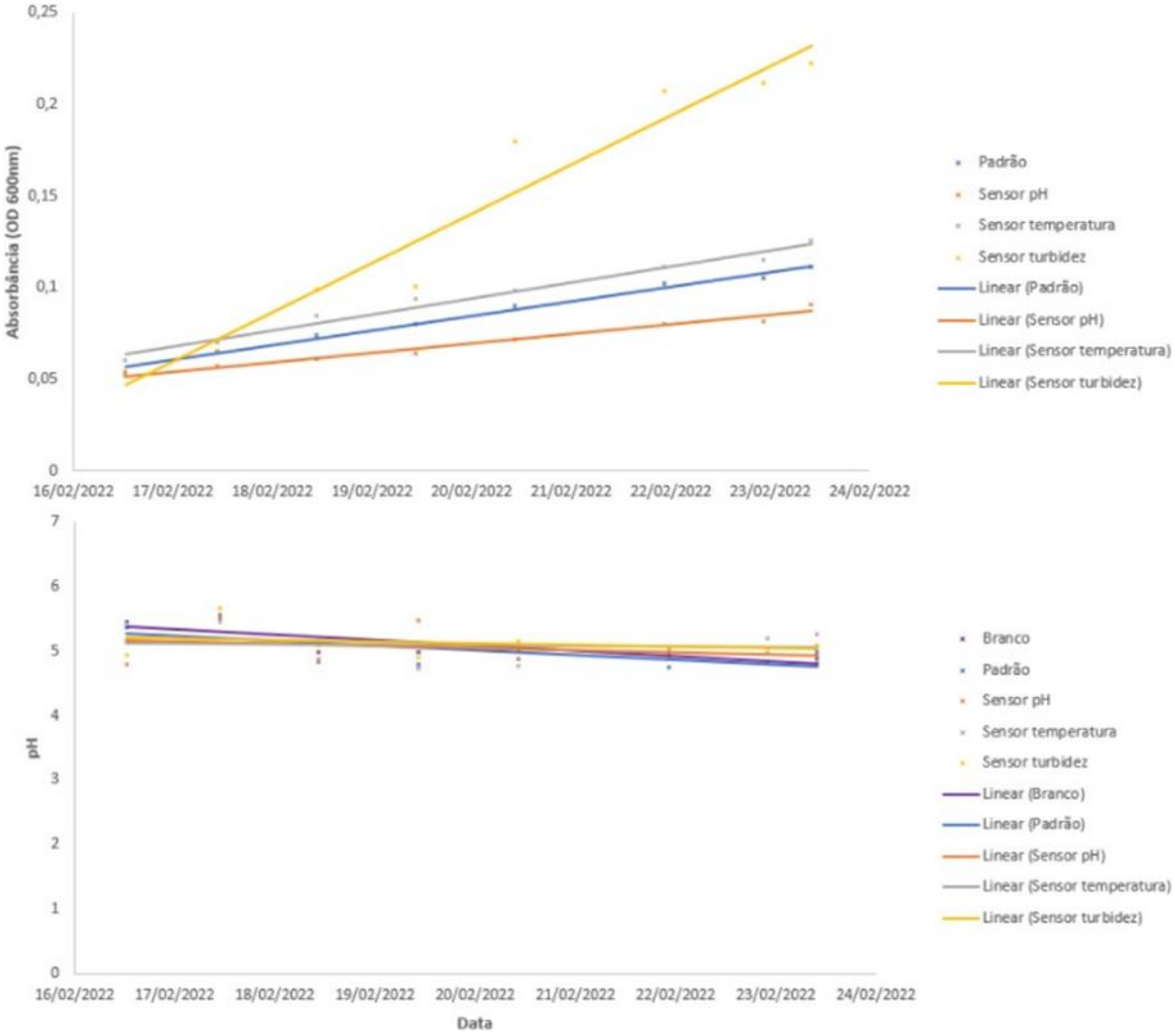
Com relação a contaminação, problema recorrente no cultivo microalgal, os sensores de baixo custo mostram-se promissores. A higienização dos dispositivos com álcool 70% e UV (20 minutos) foi suficiente, visto que não houve crescimento de fungos. Além de que o acompanhamento do cultivo pela aplicação desenvolvida diminuiu consideravelmente as chances de contaminação, sem a necessidade de abrir os frascos diariamente para coleta de amostras.

Os dados medidos por meio dos equipamentos tradicionais foram utilizados para construir os gráficos de absorbância e pH ao longo do período estudado, como ilustrado na Figura 14.

Em relação ao pH, as amostras apresentaram um comportamento muito semelhante entre si, com valores em torno de 5.

Todos os frascos iniciaram com valores de absorbância muito próximos, entre 0,052 e 0,06, que cresceram constantemente ao decorrer do cultivo. A amostra contendo o pHmetro apresentou o menor crescimento celular, não alcançando nem 0,1 de absorbância. A amostra padrão e a que continha o sensor de temperatura apresentaram um desenvolvimento muito similar e interessante, chegando a alcançar 0,111 e 0,126, respectivamente. O frasco contendo o sensor de turbidez, assim como já relatado na Figura 13, teve um aumento exacerbado, apresentando valores superiores a 0,2.

Figura 14 - Acompanhamento do cultivo pelos equipamentos comerciais do laboratório



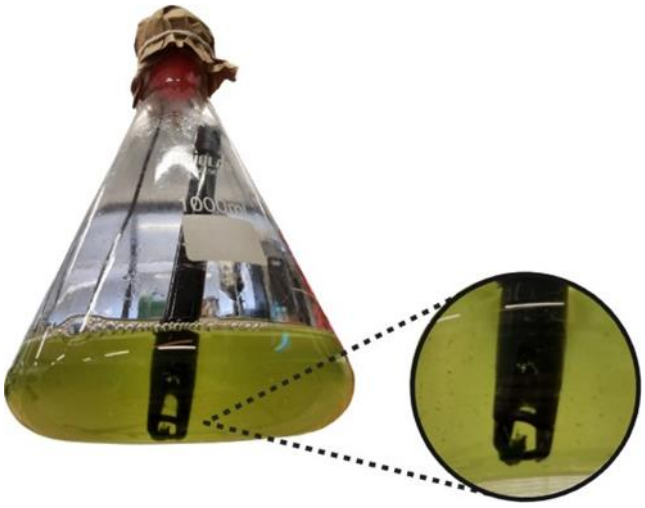
Fonte: A autora.

Os comportamentos analisados na Figura 14 também são visíveis por meio da Figura 15, onde é possível notar a variação de coloração dos 5 frascos com o passar do tempo. O branco permaneceu sem turbidez durante todo o período de estudo. O padrão e os frascos contendo os sensores de temperatura e pH apresentaram um notável aumento na coloração esverdeada, esse último com uma menor intensidade. Uma hipótese para a menor turbidez na amostra contendo o pHmetro, é o acúmulo de biomassa na parte inferior do sensor, como é visto na Figura 16, o que impossibilita a luminosidade chegar em todas as células aglomeradas, reduzindo a taxa de crescimento da microalga. É curioso que isso só ocorreu com esse sensor e que, mesmo em constante agitação no *shaker* e durante as coletas diárias, isso retornava a ocorrer.

Figura 15 - Mudança na coloração durante o cultivo

Fonte: A autora.

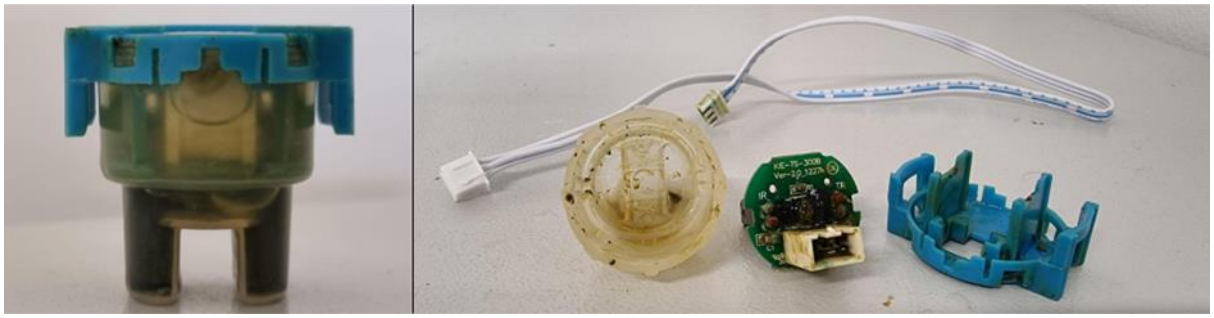
Figura 16 - Acúmulo de biomassa no pHmetro



Fonte: A autora.

Além disso, é aparente (Figura 15) que a partir do quarto dia a coloração da amostra contendo o sensor de turbidez foi mudando para uma coloração amarelada, algo inusitado visto que a microalga cultivada tem uma coloração esverdeada, própria do gênero *Chlorella*. A explicação mais plausível para esse fenômeno foi a entrada da solução microalgal no interior do sensor de turbidez, algo que não deveria ocorrer, mas que foi ocasionado em razão da agitação do frasco de cultivo no *shaker* e dos deslocamentos diários para retirada de alíquota. Provavelmente, o líquido provocou danos à parte eletrônica do turbidímetro e, como se trata de uma solução ácida, houve a formação de ligações iônicas, causando uma maior condutividade elétrica e, consequentemente, causando curto circuito no sensor.

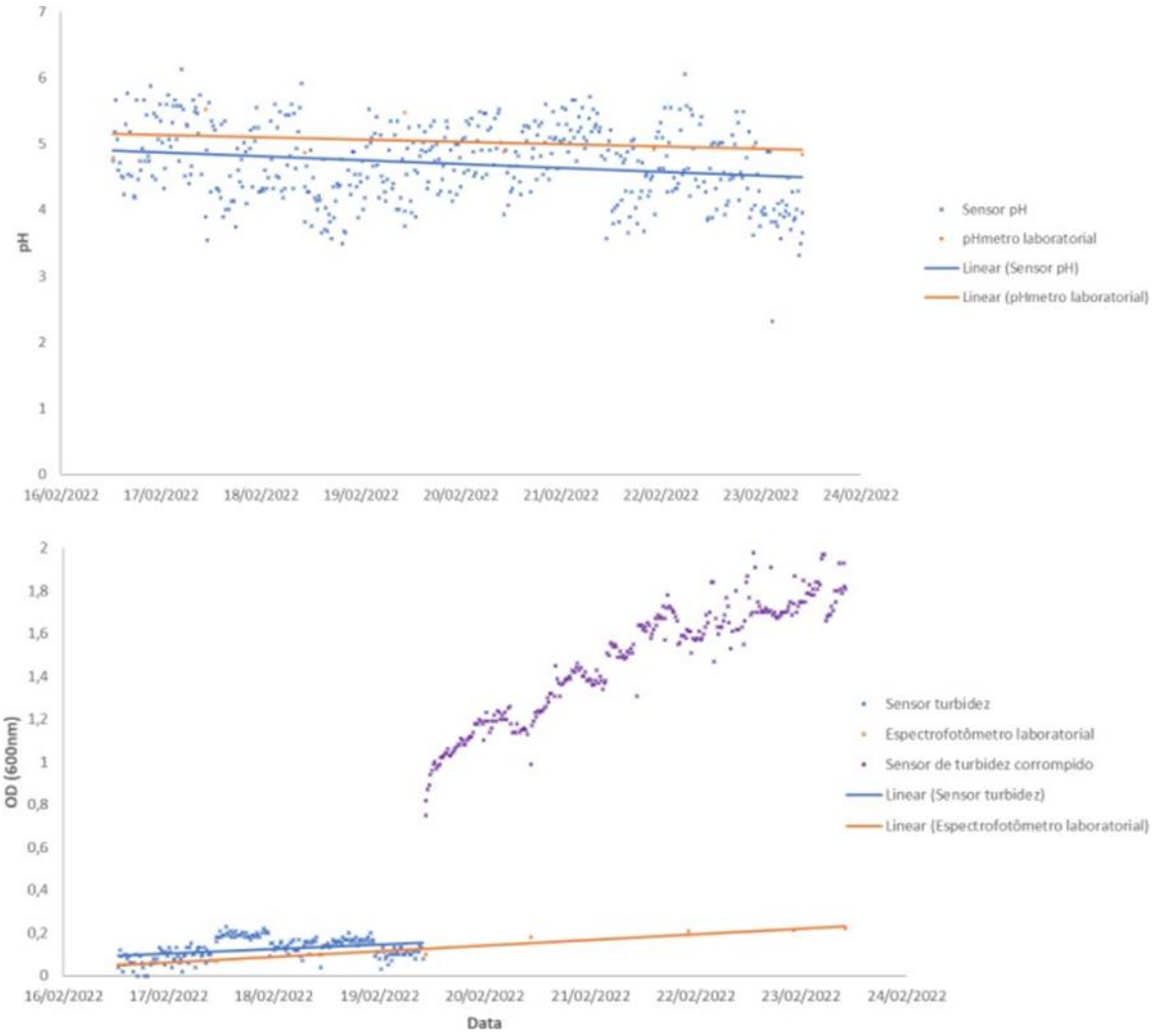
Figura 17 - Turbidímetro corrompido



Fonte: A autora.

Comparativamente com os equipamentos comerciais laboratoriais utilizados (pHmetro e espectrofotômetro), os sensores de Arduino apresentaram grande eficácia, como visto na Figura 18. O sensor de pH do Arduino apresentou um comportamento quase paralelo com o pHmetro laboratorial, com um desvio superior durante toda a reta. Assim, uma alteração na calibração do pHmetro poderia ser capaz de reduzir esse erro a praticamente zero. O turbidímetro do Arduino apresentou um comportamento semelhante nos 3 primeiros dias de cultivo, com valores muito próximos aos captados pelo equipamento laboratorial. Porém, a partir do terceiro dia ocorreu uma alteração brusca no valor obtido pelo sensor, em resposta à entrada de solução no equipamento. Portanto, enquanto o equipamento operou sem interferência do meio, este apresentou grande conformidade com o esperado, demonstrando grandes chances de ser possível aproveitá-lo em trabalhos futuros.

Figura 18 - Variação dos parâmetros coletados por sensores de Arduino e equipamentos comerciais de laboratório



Fonte: A autora.

4.4. CUSTOS DOS SENSORES À BASE DE ARDUINO

Para montar o sistema Arduino utilizado neste projeto foi investido um valor total de R$ 374,09. Os valores estão descritos na Tabela 1, bem como os custos dos equipamentos comerciais semelhantes aos utilizados nos sensores de Arduino.

Tabela 1 - Valor investido no sistema de baixo custo

Sistema Arduino		**Equipamento Comercial Semelhante**		
Produto	**Valor**	**Produto**	**Valor**	**Fonte**
Kit Arduino Start • Placa Arduino UNO R3 • Cabo USB • *Protoboard* de 400 pontos • *Jumpers* • Resistores	R$ 118,27			
Protoboard 830 pontos	R$ 22,90			
NodeMCU ESP8266	R$ 27,32			
Adaptador para Módulo WiFi ESP8266 ESP-01	R$ 22,90			
Sensor de temperatura à prova d' água DS18B20	R$ 21,78	Datalogger de temperatura RC-4	R$ 199,90	*
Sensor de Turbidez LGZD	R$ 48,58	Espectrofotômetro Kasvi	R$ 4.740,21	**
Sensor de pH PH4502C com eletrodo BNC	R$ 111,59	PHmetro Kasvi	R$ 2.487,29	**
Sensor de luminosidade LDR	R$ 0,75	Relé Fotosoquete	R$ 29,90	***
Total	**R$ 374,09**	**Total**	**R$ 7.457,30**	
Economia	**95%**			

Fonte: A autora.

***** Elitech Brasil, 2022; ****** Prolab, 2022; ******* Cassol centerlar, 2022

A utilização do sistema de baixo custo citado é extremamente vantajosa financeiramente, visto que pode ser aplicado em projetos de grandes rendimentos, além de ter a capacidade de ser totalmente personalizado de acordo com o experimento. Porém, cabe ressaltar que o sistema Arduino não é capaz de substituir completamente os equipamentos laboratoriais comerciais, visto que estes são necessários para calibrar os sensores de Arduino utilizados. Além disso, mais estudos devem ser realizados para aprimorar o uso do sistema Arduino em processos biotecnológicos.

4.5. DESAFIOS

A realização desse estudo foi bem desafiadora, primeiramente por utilizar tecnologias não estudadas no curso de Engenharia de Alimentos, como também por relacionar temas (Arduino + desenvolvimento de aplicação *web* + *Chlorella vulgaris*) que apresentam pouquíssima correlação na literatura. As maiores dificuldades envolveram o desenvolvimento do código do NodeMCU, para conexão com WiFi, visto que a conexão com rede com autenticidade apresenta certas barreiras, como também o desenvolvimento do código que correlaciona o Arduino e a aplicação *web*.

4.6. APLICAÇÕES

O presente trabalho relacionou o uso do sistema de baixo custo Arduino com o cultivo microalgal, porém ele pode ser utilizado em variados processos biotecnológicos, bastando calibrar os sensores de acordo com a finalidade do uso.

4.7. LIMITAÇÕES

A principal limitação observada foi em relação ao uso do sensor de turbidez. O dispositivo apresentou altíssima fragilidade perante à umidade e facilidade na entrada de água no seu interior. Assim, alterações no dispositivo devem ser feitas para evitar tal acontecimento, garantindo maior segurança nos dados informados pelo sensor e evitando interferir no cultivo. Outra sugestão é a utilização de outro tipo de turbidímetro. Como é o caso do Guedes *et al.* (2017) que utilizou o sensor de

luminosidade LDR para propor uma correlação com a absorbância do processo. Para tal fim, os autores instalaram o dispositivo na parede oposta do fluxo luminoso e acompanharam o crescimento celular durante o cultivo de 15 dias. Ao final do processo traçaram uma curva relacionando os valores obtidos de luminosidade e absorbância e obtiveram uma curva com R^2 de 0,96, proporcionando uma correlação que se ajusta aos dados e validando o uso do sensor dessa forma.

Outra limitação observada foi em relação ao fornecimento de energia. Já que tanto o funcionamento dos sensores quanto o envio das informações para a aplicação *web* necessitavam de energização, as quedas de eletricidade podem ter interferido no processo. Uma alternativa é o uso de uma bateria de lítio para alimentação de energia ao sistema. Porém, como a falta de energia também interfere no sinal WiFi, pode ser associado um dispositivo que armazena os dados coletados no Arduino, de forma com que a perda de conexão com a Internet não interfira na coleta de informações.

4.8. FUTUROS ESTUDOS

Outros estudos são necessários para viabilizar o uso dos sensores de Arduino para avaliar o crescimento da *C. vulgaris*. Dessa forma, indico a adição de uma tela LCD para visualizar os valores no local de execução do experimento e o uso de um botão *play/pause*, não necessitando a desconexão do sistema Arduino da energia quando os sensores forem desconectados do dispositivo principal.

Em relação aos sensores, acredito ser importante a utilização de um sensor de intensidade de luz (em Lux), garantindo que a microalga está recebendo a luminosidade necessária para crescer. Além disso, como já citado, é importante a alteração do sensor de turbidez, evitando a entrada de meio no seu interior. E, para finalizar, é interessante configurar o Arduino para considerar um ponto como sendo a média de 3 medições, tornando o valor ainda mais real.

Quanto ao cultivo, é interessante o aumento no número de dias por ensaio para permitir que a microalga atinja e complete a sua fase exponencial de crescimento, bem como o ajuste do pH, para garantir um melhor crescimento microalgal.

5. CONCLUSÃO

O microcontrolador Arduino mostrou-se adequado, visto o baixo custo, boa precisão e exatidão, além de poder ser utilizado, potencialmente, em outros processos biotecnológicos. A criação da aplicação *web* garantiu uma análise em tempo real e *ex situ* do crescimento microalgal. A integração do sistema de baixo custo à IoT foi muito benéfica, evitando a contaminação das amostras e com potencial de ser associado à inteligência artificial. Assim, o trabalho foi capaz de comprovar a viabilidade no uso do sistema de baixo custo Arduino para avaliar o crescimento da *C. vulgaris*, por meio dos sensores de luminosidade, temperatura, pH e turbidez, além de garantir conexão com a nuvem, apresentando dados em tempo real.

REFERÊNCIAS

ABDULLAH, Hanin M; ZEKI, Ahmed M. **Frontend and Backend Web Technologies in Social Networking Sites: Facebook as an Example**. 3rd International Conference on Advanced Computer Science Applications and Technologies, Barém, 2014, doi: 10.1109/ACSAT.2014.22

Arduino.cc. Disponível em: https://www.arduino.cc/. Acessado em 01/03/2022.

ARIAWAN, Eko; MAKALEW, Stanley A. **Smart Micro Farm: Sustainable Algae *Spirulina* Growth Monitoring System**. International Conference on Information Technology and Electrical Engineering (ICITEE), Indonesia, 2018, doi: 10.1109/ICITEED.2018.8534904.

BANZI, Massimo; SHILOH, Michael. **Primeiros Passos com o Arduino** - 2ª Edição: A plataforma de prototipagem eletrônica open source. Novatec Editora, 2015.

BARGHBANI, Rouhollah; REZAEI, Karamatollah; JAVANSHIR, Aziz. **Investigating the Effects of Several Parameters on the Growth of Chlorella vulgaris Using Taguchi's Experimental Approach**. International Journal of Biotechnology for Wellness Industries, Iran, 2012, doi: 10.6000/1927-3037/2012.01.02.04

BATES, Harvey *et al.* **A guide to Open-JIP, a low-cost open-source chlorophyll fuorometer**. Photosynthesis Research, 2019, doi: 10.1007/s11120-019-00673-2.

Baú da Eletrônica. Disponível em: https://www.baudaeletronica.com.br Acesso em: 17/02/2022.

BOROWITZKA, Michael A. **Biology of Microalgae**. Microalgae in Health and Disease Prevention, 2018, doi: 10.1016/B978-0-12-811405-6.00003-7

BRÁNYIKOVÁ, Irena *et al.* **Microalgae - Novel Highly Efficient Starch Producers**. Biotechnology and Bioengineering, República Tcheca, 2011, doi: 10.1002/bit.23016.

BREUER, Guido *et al.* Effect of light intensity, pH, and temperature on triacylglycerol (TAG) accumulation induced by nitrogen starvation in *Scenedesmus obliquus*. Bioresource Technology, Holanda, 2013, doi: 10.1016/j.biortech.2013.05.105.

Capterra. Disponível em: https://www.capterra.com.br/. Acesso em: 02/03/2022.

CARRASQUILLA-BATISTA, Arys *et al.* **Characterization of biomass pellets from Chlorella vulgaris microalgal production using industrial wastewater**. International Conference in Energy and Sustainability in Small Developing Economies (ES2DE), Costa Rica, 2017, doi: 10.1109/ES2DE.2017.8015352.

Cassol Centerlar. Disponível em: https://www.cassol.com.br/. Acesso em: 05/03/2022.

Clemente, M. 2019. **O que é e como usar uma Query**. Documento da *Web*, RockContent. Disponível em: https://rockcontent.com/br/blog/query/. Acesso em: 02/03/2022.

DFRobot. **PH meter SKU SEN0161**. Disponível em: https://wiki.dfrobot.com/PH_ meter_SKU SEN0161_. Acesso em: 01/03/2022.

ECharts. Disponível em: https://echarts.apache.org/. Acesso em: 02/03/2022.

Eletrogate. Disponível em: https://www.eletrogate.com/. Acesso em: 17/02/2022.

Elitech Brasil. Disponível em: https://www.elitechbrasil.com.br/. Acesso em: 05/03/2022.

Express. Disponível em: https://expressjs.com/. Acesso em: 02/03/2022.

FLORES, Gerardo *et al*. **A turbidity sensor development based on NL-PI observers: Experimental application to the control of a Sinaloa's River *Spirulina* maxima cultivation.** Open Chemistry, México, 2020, doi: 10.1515/chem-

2020-0119.

GILLIS, Alexander S. 2021. **What is Internet of things (IoT)?**. TechTarget - IoTAgenda, 2022. Disponível em: https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. Acesso em: 27/02/2022.

GitHub. Disponível em: https://github.com/. Acesso em: 02/03/2022.

GUEDES, Ana Catarina; AMARO, Helena M.; MALCATA, Francisco Xavier. **Microalgae as Sources of Carotenoids**. Marine Drugs, Portugal, 2011, doi: 10.3390/md9040625.

GUEDES, Vanessa Campos et al. Luminosity as function of optical density in microalgae culture. XXI Simpósio Nacional De Bioprocessos, Sergipe, 2017.

GUIRY, Michael D. e GUIRY, Gwendoline M. AlgaeBase, National University of Ireland, Galway. Disponível em: https://www.algaebase.org. Acesso em: 25/02/2022.

Heroku Dev Center. Disponível em: https://devcenter.heroku.com/. Acesso em: 02/03/2022.

International Data Corporation (IDC). Disponível em: https://www.idc.com/. Acesso em: 06/03/2022.

Knex.js. Disponível em: https://knexjs.org/. Acesso em: 02/03/2022.

KHOEYI, Zahra Amini; SEYFABADI, Jafar; RAMEZANPOUR, Zohreh. **Effect of light intensity and photoperiod on biomass and fatty acid composition of the microalgae, *Chlorella vulgaris***. Aquaculture International, Iran, 2012, doi: 10.1007/s10499-011-9440-1.

KHOO, Kuan Shiong *et al.* **Nanomaterials Utilization in Biomass for Biofuel and Bioenergy Production**. Energies, 2020, doi: 10.3390/en13040892.

KOLBAN, Neil. **Kolban's Book on ESP32 & ESP8266**. Estados Unidos, 2016.

LOUIS, Leo. **Working principle of Arduino and using it as a tool for study and research**. International Journal of Control, Automation, Communication and Systems (IJCACS), Índia, 2016, doi: 10.5121/ijcacs.2016.1203.

MADEIRA, Daniel. **DS18B20** - Sensor de temperatura inteligente. Portal Vida de Silício, 2018. Disponível em: https://portal.vidadesilicio.com.br/sensor -de-temperatura-ds18b20/. Acesso em: 01/03/2022.

MINHAS, Amritpreet *et al*. **A Review on the Assessment of Stress Conditions for Simultaneous Production of Microalgal Lipids and Carotenoids**. Frontiers in microbiology, 2016, 10.3389/fmicb.2016.00546.

MONK, Simon. **Programação com Arduino:** Começando com sketches. Bookman, Porto Alegre, 2013, ISBN: 978-85-8260-027-6.

MORAIS, Michele Greque *el al*. **Biologically Active Metabolits Synthesized by Microalgae**. BioMed Research International, Rio Grande do Sul, 2015, doi: 10.1155/2015/835761.

Mozilla. Disponível em: https://developer.mozilla.org/. Acesso em: 02/03/2022.

NGUYEN, Binh T., RITTMANN, Bruce E. **Low-cost optical sensor to automatically monitor and control biomass concentration in microalgal cultivation**. Algal Research, Estados Unidos, 2018, doi: 10.1016/j.algal.2018.03.013.

NOLETO, Cairo. **JavaScript:** o que é, aplicação e como aprender a linguagem JS. Trybe, s.d. Disponível em: https://blog.betrybe.com/javascript/. Acesso em: 01/03/2022.

Oracle. Disponível em: https://www.oracle.com/br/. Acesso em: 02/03/2022.

PERWEJ,Yusuf *et al.* **The Internet-of-Thing (IoT) Securiry:** A Technological Perspective and Review. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2019, doi: 10.32628/IJSRCSEIT.

PostgreSQL Global Development Group. Disponível em: https://www.postgresql.org/. Acesso em: 02/03/2022.

Prolab. Disponível em: https://www.lojaprolab.com.br/. Acesso em: 05/03/2022.

RAHMAT, Ayi *et al.* **Evaluation of System Performance for Microalga Cultivation in Photobioreactor with IOTs (Internet of Things)**. International Journal of Sciences: Basic and Applied Research (IJSBAR), 2020, ISSN: 2307-4531.

RATOMSKI, Patryk; HAWROT-PAW, Malgorzata. **Production of *Chlorella vulgaris* Biomass in Tubular Photobioreactors during Different Culture Conditions**. Applied Sciences, Polônia, 2021, doi: 10.3390/app11073106.

React. Disponível em: https://create-react-app.dev/. Acesso em: 02/03/2022.

RU, Irene Tiong Kai *et al*. ***Chlorella vulgaris*: a perspective on its potential for combining high biomass with high value bioproducts**. Applied Phycology, Malásia, 2020, doi: 10.1080/26388081.2020.1715256.

SAFI, Carl *et al.* **Morphology, composition, production, processing and applications of Chlorella vulgaris:** A review. Renewable and Sustainable Energy Reviews, França, 2014, doi: 10.1016/j.rser.2014.04.007.

SIGAMANI, Santhosh; NATARAJAN, Hemalatha. Bioactive compounds from Microalgae and its different applications - a review. Advances in Applied Science Research, Índia, 2016, ISSN: 0976-8610.

Saravati. Disponível em: https://www.saravati.com.br/. Acesso em: 17/02/2022.

SETHI, Pallavi; SARANGI, Smruti R. **Internet of Things:** Architectures, Protocols,

and Applications. Journal of Electrical and Computer Engineering, Índia, 2017, doi: 10.1155/2017/9324035.

SILVEIRA, Paulo. **O que é SQL?**. Alura, 2019. Disponível em: https://www.alura.com.br/artigos/o-que-e-sql. Acesso em: 02/03/2022.

Scopus. Disponível em: https://www.scopus.com/. Acesso em: 22/02/2022.

SHARMA, Priya; KANTHA, Parveen. **'Blynk' Cloud Server based Monitoring and Control using 'NodeMCU'**. International Research Journal of Engineering and Technology (IRJET), Índia, 2020.

STRAUB, Matheus Gebert. 2020. **Sensor de turbidez** - projeto de leitura da qualidade da água. UsinaInfo, 2020. Disponível em: https://www.usinainfo.com.br/blog/se nsor-de-turbidez-projeto-de-leitura-da-qualidade-da-agua/. Acesso em: 02/03/2022.

SUHANKO, Djames. 2021. **Como calibrar o PHmetro PH4502C**. Do bit ao byte, 2021. Disponível em: https://www.dobitaobyte.com.br/como-calibrar-o-phmetro-ph4502c/. Acesso em: 01/03/2022.

THAM, Pei En et al. **Sustainable smart photobioreactor for continuous cultivation of microalgae embedded with Internet of Things**. Bioresource Technology, Malásia, 2022, doi: 10.1016/j.biortech.2021.126558.

UMAR, Lazuardi *et al.* **An Arduino uno based biosensor for water pollution monitoring using immobilised algae Chlorella vulgaris**. International Journal on Smart Sensing and Intelligent Systems, Indonesia, 2017, doi: 10.21307/ijssis-2018-027

VIDAL, Vitor. **Apostila kit Arduino start**: Volume 1. Eletrogate, 2019.

VIDOTTI, Annamaria D.S. *et al*. **Analysis of autotrophic, mixotrophic and heterotrophic phenotypes in the microalgae *Chlorella vulgaris* using time-**

resolved proteomics and transcriptomics approaches. Algal Research, São Paulo, 2020, doi: 10.1016/j.algal.2020.102060.

VITOVÁ, Milada *et al.* **Accumulation of energy reserves in algae: From cell cycles to biotechnological applications**. Biotechnology Advances, 2015, doi: 10.1016/j.biotechadv.2015.04.012.

Visual Studio Code. Disponível em: https://code.visualstudio.com/. Acesso em: 02/03/2022.

WANG, Kexin et al. **How does the Internet of Things (IoT) help in microalgae biorefinery?**. Biotechnology Advances, 2022, doi: 10.1016/j.biotechadv.2021.107819.

YAMAMOTO, Maki *et al*. **Relationship between presence of a mother cell wall and speciation in the unicellular microalga *Nannochloris* (*Chlorophyta*).** Journal of Phycology, Japão, 2003, doi: 10.1046/j.1529-8817.2003.02052.x.

YAMAMOTO, Maki; FUJISHITA, Mariko; HIRATA, Aiko. **Regeneration and maturation of daughter cell walls in the autospore-forming green alga *Chlorella vulgaris (Chlorophyta, Trebouxiophyceae*).** The Journal of Plant Research, Japão, 2004, doi: 10.1007/s10265-004-0154-6.

APÊNDICE A - Código de programação do Arduino (Arduino IDE)

```
// ------ CONEXÃO COM WIFI ------

// Definição das bibliotecas utilizadas e portas para wifi
#include <SoftwareSerial.h>
#include <ArduinoJson.h>
SoftwareSerial s(5,6); //(RX,TX)                                    // define as portas digitais 5 e 6 como as entradas RX e TX, respectivamente

// ------ SENSOR DE LUZ ------

// Definição de portas e variáveis para luz
#define analogLDR A0                                                // define o pino analógico A0 para o sensor de luz
int sensorLDR = 0.0;                                                // variável inicial igual a zero
float voltagemLDR;


// ------ SENSOR DE TEMPERATURA ------

// Definição das bibliotecas utilizadas e portas para temperatura
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 8                                              // define a porta digital 8 para o sensor de temperatura
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress sensorTemperatura;

// ------ SENSOR DE pH ------

// Definição de portas e variáveis para pH
#define analogpH A1                                                 // define o pino analógico A1 para o sensor de pH
int sensorpH = 0.0;                                                 // variável inicial igual a zero
float voltagempH;
float desvio;
float calibracao;
float ph;
```

```
// ------ SENSOR DE TURBIDEZ ------
#define analogTurbidez A2                                   // define o pino analógico A2 para o sensor de turbidez
float sensorTurbidez = 0.0;                                 // variável inicial igual a 0
float voltagemTurbidez;
float sensorTurbidezNTU;
float calibracaoTurbidez = 1.129032258;                     // fator de calibração

// ------ GERAL ------
// Definição das configurações iniciais
void setup(){
  Serial.begin(9600);                                       // inicia monitor serial - velocidade 9600 bps
  s.begin(9600);

  // ------ SENSOR DE LUZ ------
  pinMode(analogLDR, INPUT);                                // configura o LDR como dispositivo de entrada

  // ------ SENSOR DE TEMPERATURA ------
  sensors.begin();                                          // inicia sensor de temperatura
  sensors.getAddress(sensorTemperatura, 0);                 // recebe localização do sensor de temperatura

  // ------ SENSOR DE pH ------
  pinMode(analogpH, INPUT);                                 // configura o pHmetro como dispositivo de entrada

  // ------ SENSOR DE TURBIDEZ ------
  pinMode(analogTurbidez, INPUT);                           // configura o turbidímetro como dispositivo de entrada
}

// ------ GERAL ------
// Definição das atividades em loop infinito
void loop(){

  // ------ SENSOR DE LUZ ------
  sensorLDR = analogRead(analogLDR);                        // leitura no pino analógico A0
  voltagemLDR = sensorLDR * (5/1024);                       // cálculo da tensão no LDR

  // ------ SENSOR DE TEMPERATURA ------
  sensors.requestTemperatures();                            // requisição da temperatura do "sensorTemperatura"
  float temperatura = sensors.getTempC(sensorTemperatura);  // armazena valor da temperatura na variável "temperatura"
```

```
  // ------ SENSOR DE pH ------
  sensorpH = analogRead(analogpH);                                    // leitura no pino analógico A1
  voltagempH = sensorpH * (5/1024);                                   // cálculo da tensão no pHmetro
  calibracao = 19.44;                                                 // valor de calibração atribuído
  ph = -5.70 * voltagempH + calibracao;                               // cálculo do valor do pH

  // ------ SENSOR DE TURBIDEZ ------
  sensorTurbidez = analogRead(analogTurbidez);                        // leitura no pino analógico A2
  voltagemTurbidez = sensorTurbidez * (5/1024) * calibracaoTurbidez;  // cálculo da tensão de turbidez
  sensorTurbidezNTU = ((-0.6607)*voltagemTurbidez)+2.6871;            // cálculo do resultado da turbidez a partir da equação...
                                                                      // ... de curva definida laboratorialmente
  // ------ CONEXÃO COM WIFI - ARDUINO SLAVE ------
  StaticJsonBuffer<1000>jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();                     // criação de um objeto com as rotas dos sensores de Arduino
  root["temperatura"] = temperatura;
  root["ph"] = ph;
  root["turbidez"] = sensorTurbidezNTU;
  root["luminosidade"] = voltagemLDR;
  if(s.available()>0);{
    root.printTo(s);                                                // se há conexão com a internet, envia os valores dos...
  }                                                                 // ... sensores pelas rotas pré-estabelecidas

  // ------ GERAL ------
  delay(10000);                                                     // (milissegundos)- atraso de 10 segundos
}
```

APÊNDICE B - Código de programação do NodeMCU ESP8266 (Arduino IDE)

```
// Definição das bibliotecas utilizadas
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <SoftwareSerial.h>
SoftwareSerial s(D5,D6); //(RX,TX)                    // define os pinos D5 e D6 do NodeMCU como RX e TX, respectivamente
#include <ArduinoJson.h>

extern "C" {                                          // inclui as bibliotecas externas da linguagem C
#include "user_interface.h"
#include "wpa2_enterprise.h"
#include "c_types.h"
}

char ssid[] = "eduroam";                              // nome da rede de conexão WiFi
char username[] = "jackelyne.souza";                  // usuário na autenticação do WiFi
char identity[] = "jackelyne.souza";                  // usuário na autenticação do WiFi
char password[] = "****";                             // senha correspondente ao usuário na autenticação WiFi

uint8_t target_esp_mac[6] = {0x24, 0x0a, 0xc4, 0x9a, 0x58, 0x28};

String server_url = "http://tcc-jah-backend.herokuapp.com/data";          // URL do endpoint

unsigned long lastTime = 0;                           // zera o tempo, gerando um número que pode ser armazenado em um int
unsigned long timerDelay = 5000;                      // timer de 5 segundos

void setup() {

  WiFi.mode(WIFI_STA);
  Serial.begin(9600);                                 // inicia monitor serial - velocidade 9600 Bps
  s.begin(9600);
  while (!Serial) continue;
  WiFi.begin(ssid, password);
  Serial.println("Connecting");
```

```
  Serial.setDebugOutput(true);
  wifi_set_opmode(STATION_MODE);                                    // inicio do processo de autenticação no WiFi
  struct station_config wifi_config;

  memset(&wifi_config, 0, sizeof(wifi_config));
  strcpy((char*)wifi_config.ssid, ssid);
  strcpy((char*)wifi_config.password, password);

  wifi_station_set_config(&wifi_config);
  wifi_set_macaddr(STATION_IF,target_esp_mac);

  wifi_station_set_wpa2_enterprise_auth(1);

  wifi_station_clear_cert_key();                                    // limpa os dados, para ter certeza que não há antigos
  wifi_station_clear_enterprise_ca_cert();
  wifi_station_clear_enterprise_identity();
  wifi_station_clear_enterprise_username();
  wifi_station_clear_enterprise_password();
  wifi_station_clear_enterprise_new_password();

  wifi_station_set_enterprise_identity((uint8*)identity, strlen(identity));
  wifi_station_set_enterprise_username((uint8*)username, strlen(username));
  wifi_station_set_enterprise_password((uint8*)password, strlen((char*)password));

void loop() {
  if ((millis() - lastTime) > timerDelay) {                         // envia uma solicitação HTTP, dependendo do timerDelay
    if(WiFi.status()== WL_CONNECTED){                               // checa se está conectado ao WiFi
      WiFiClient client;
      HTTPClient http;

  StaticJsonBuffer<200>jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(s);                     // criação de um objeto com as rotas dos senssores de Arduino
  if(root==JsonObject::invalid())
    return;

  Serial.print("temperatura: ");
  float temperatura=root["temperatura"];
  Serial.println(temperatura);
```

```
  Serial.print("ph: ");
  float ph=root["ph"];
  Serial.println(ph);

  Serial.print("turbidez: ");
  float turbidez=root["turbidez"];
  Serial.println(turbidez);

  Serial.print("luminosidade: ");
  int luminosidade=root["luminosidade"];
  Serial.println(luminosidade);


      char json_str[100];
      root.prettyPrintTo(json_str, sizeof(json_str));

      // String que armazena os parâmetros a serem enviados para o endpoint
      String server_path = server_url + "?temperatura=" + temperatura + "&ph=" + ph + "&turbidez=" + turbidez + "&luminosidade=" + luminosidade;

      http.begin(client, server_path);                                  // inicia a rota com o endpoint para envio dos parâmetros
      http.addHeader("Content-Type", "application/json");

      int httpCode = http.GET();

      Serial.println(http.getString());
      if (httpCode>0) {                                                 // verifica se está tudo certo com a conexão com o endpoint
        Serial.print("HTTP Response code: ");
        Serial.println(httpCode);
        String payload = http.getString();
        Serial.println(payload);
      } else {
        Serial.print("Error code: ");
        Serial.println(httpCode);
      }
      http.end();
    }
    else {
      Serial.println("WiFi Disconnected");
    }
    lastTime = millis();
  }

  delay(1200000);                                                   // (milissegundos)- atraso de 20 minutos
}
```

APÊNDICE C - Código de programação do *backend* (*Visual Studio Code*)

ARQUIVO: index.js

```
const express = require('express');
const routes = require('./routes');
const cors = require('cors');

const app = express();

app.use(express.json());

app.use((req, res, next) => {
    //allowing localhost:3000 conection
    res.header("Access-Control-Allow-Origin", "*");
    //methods
    res.header("Access-Control-Allow-Methods", 'GET,PATCH,POST,DELETE');
    res.header("Access-Control-Allow-Headers", ["Content-Type", "Authorization"]);
    app.use(cors());
    next();
});

app.use(routes);

// porta preenchida automaticamente pelo heroku em prod
const port = process.env.PORT || 3333;
app.listen(port, () => console.log("Listening on port", port));
```

ARQUIVO: routes.js

```
const express = require('express');


const DataController = require('./Controllers/DataController');


const routes = express.Router();


//DataController
routes.get('/data', DataController.create);
routes.get('/get-data', DataController.index);
routes.get('/drop-table', DataController.dropTable);


module.exports = routes;
```

ARQUIVO: DataController.js

```
const connection = require('../database/connection');
const crypto = require('crypto');


module.exports = {
    async index(request, response) {
        // const users = await connection('users').select('name', 'role', 'email', 'password')
        // return response.json(users)
        connection('data_table')
        .select('*')
        .then(result =>{
            const answ = {
                "data": result
            }
            return response.json(answ)
        }).catch(err => {
            return response.status(409).send({msg: err})
        });
    },
     async create(request, response){
        const { temperatura, ph, turbidez, luminosidade } = request.body;
        let hora = new Date().getTime();
        console.log(hora);
        if(!hora || !temperatura || !ph || !turbidez || !luminosidade) {
            return response.status(409).send({ msg: 'missing data' });
        }
        const id = crypto.randomBytes(4).toString('HEX');
        connection('data_table').insert({
            id,
            hora,
            temperatura,
            ph,
            turbidez,
            luminosidade
        }).then( result => {
            return response.status(201).send({ msg: 'Dado inserido no BD com sucesso', hora })
        }).catch(err => {
            return response.status(409).send({ msg: err });
        });
    },
    async dropTable(request, response){
        connection('data_table')
        .truncate()
        .then(result =>{
            return response.status(200).send({ msg: 'Tabela resetada' });
        }).catch(err => {
            return response.status(409).send({msg: err})
        });
    },
}
```

APÊNDICE D - Código de programação do *frontend* (*Visual Studio Code*)

ARQUIVO: Layout.jsx

```jsx
import React, { useEffect, useState } from 'react';
import styles from './styles.module.css';
import {Link, Route, BrowserRouter as Router, Switch} from "react-router-dom";
import Contato from "./Contato";
import Home from "./Home";
import ScientistImage from "../assets/img/scientist.png";
import Footer from "../components/Footer/Footer";
import {Button} from "@material-ui/core";
import {api} from "../api/api";
import {CircularProgress} from "@mui/material";

function Layout() {
    const [modalState, setModalState] = useState({loading: false, active: false, error: false});

    function toggleModal() {
        setModalState({...modalState, active: !modalState.active});
    }

    async function deleteData() {
        setModalState({...modalState, loading: true })
        const a = await api.dropTable();
        if(a) {
            document.location.reload(true)
        } else {
            setModalState({...modalState, loading: false, error: "algum erro ocorreu" })
        }
    }

    return (
        <div className={styles.app_wrapper}>
            <Router>
                <div className={styles.all_content}>
                    <div className={styles.header_main}>
                        {window.innerWidth >= 820 && <img height="120px" src={ScientistImage} />}
                        <div className={styles.title_nav}>
                            <div>
                                <h1 className={window.innerWidth < 820 ? styles.cheap_scientist_mobile :styles.cheap_scientist}>Cheap Scientist
                            </div>
                            <div className={styles.nav_main}>
                                <Link className={styles.nav_item} to="/inicio">INÍCIO</Link>
                                <Link className={styles.nav_item} to="/contato">CONTATO</Link>
                                <Button onClick={toggleModal}>
                                    <i className="fas fa-trash" style={{ fontSize: '34px'}}></i>
                                </Button>
                            </div>
                        </div>
                    </div>
                    <div className={styles.main_content}>
                        <Switch>
                            <Route path="/contato">
                                <Contato />
                            </Route>
                            <Route path="/">
                                <Home />
                            </Route>
                        </Switch>
                    </div>
                    <Footer />
                </div>
            </Router>
            {modalState.active && (
                <>
                    {!modalState.loading && !modalState.error && (<>
                        <div className={styles.modal}>
                            <div className={styles.modal_phrase}>Deseja deletar os dados do banco de dados?</div>
                            <div className={styles.modal_buttons}>
                                <Button onClick={deleteData} className={styles.confirm_button}>Sim</Button>
                                <Button onClick={toggleModal}>Não</Button>
                            </div>
                        </div>
                    </>)}
```

```
                    {modalState.error && (<>
                        <div className={styles.modal}>
                            <div className={styles.modal_phrase}>{modalState.error}</div>
                        </div>
                    </>)}
                    {modalState.loading && (<>
                        <div className={styles.modal}>
                            <CircularProgress />
                        </div>
                    </>)}
                    <div className={styles.to_close_modal} onClick={toggleModal} />
                </>
            )}
        </div>
    );
}

export default Layout;
```

ARQUIVO: Home.jsx

```
import React, {useEffect, useState} from 'react';
import HomeCard from "../components/HomeCard/HomeCard";
import styles from './styles.module.css';
import { api } from "../api/api";
import { Button } from '@mui/material';
import {format} from "date-fns";

const Home = () => {
    const [homeValues, setHomeValues] = useState({ pH: 'loading', temperatura: 'loading', luminosidade: 'loading', turbidez: 'loading', data: '

    useEffect(() => {
        getData();
        recursiveFunction();
    }, []);

    async function recursiveFunction() {
        setTimeout(() => {
            getData();
            secondaryRecursiveFunction();
        }, 50000);
    }

    async function secondaryRecursiveFunction() {
        setTimeout(() => {
            getData();
            recursiveFunction();
        }, 50000);
    }

    async function getData() {
        const answer = await api.getHomeData();
        if(!answer || answer.length === 0) return;
        setHomeValues({
            pH: `${answer[answer.length-1].ph}`,
            temperatura: `${answer[answer.length-1].temperatura}ºC`,
            luminosidade: `${answer[answer.length-1].luminosidade} `,
            turbidez: `${answer[answer.length-1].turbidez} OD(600)`,
            data: answer,
        });
    }

    const pH = {
        name: 'pH',
        value: homeValues.pH,
        varName: 'ph'
    }

    const temperatura = {
        name: 'Temperatura',
        value: homeValues.temperatura,
        varName: 'temperatura',
        formatter: 'ºC'
    }

```

```
    const luminosidade = {
        name: 'Luminosidade',
        value: homeValues.luminosidade,
        varName: 'luminosidade',
        formatter: ''
    }

    const turbidez = {
        name: 'Turbidez',
        value: homeValues.turbidez,
        varName: 'turbidez',
        formatter: ' OD(600)'
    }

    function exportData() {
        const { data } = homeValues;
        if(data === 'loading' && data.length === 0) return;
        const csvContent = `data:text/csv;charset=utf-8,${
            data.map((row, i) => {
                console.log(row);
                let formattedRow = row;
                delete formattedRow.id;
                formattedRow.hora = format(row.hora, 'dd/MM/yy hh:mm')
                let newRow = Object.values(row);
                newRow = newRow.join(';');
                if(i === 0) {
                    newRow = `${Object.keys(formattedRow).join(';')}
                    ${newRow}`
                }
                return newRow;
            }).join('\r\n')}`;
        const encodedUri = encodeURI(csvContent);
        const link = document.createElement('a');
        link.setAttribute('href', encodedUri);
        link.setAttribute('download', `data.csv`);
        document.body.appendChild(link); // Required for FF
        link.click();
        document.body.removeChild(link);

    }


    return (
        <>
            <div className={window.innerWidth < 820 ? styles.page_mobile :styles.page}>
                <div className={styles.export_button_container}>
                    <Button onClick={exportData} variant="text">
                        <span className={styles.export_button_icon}>
                            <i className="fas fa-file-download"></i>
                        </span>
                    </Button>
                </div>
                <div className={window.innerWidth < 820 ? styles.dual_card_holder_mobile :styles.dual_card_holder}>
                    <HomeCard config={pH} data={homeValues.data} />
                    <HomeCard config={temperatura} data={homeValues.data} />
                </div>
                <div className={window.innerWidth < 820 ? styles.dual_card_holder_mobile :styles.dual_card_holder}>
                    <HomeCard config={luminosidade} data={homeValues.data} />
                    <HomeCard config={turbidez} data={homeValues.data} />
                </div>
            </div>
        </>
    );
};

export default Home;
```

ARQUIVO: HomeCard.jsx

```
import React from 'react';
import styles from './styles.module.css';
import ReactECharts from 'echarts-for-react';
import { format } from 'date-fns';

```

```jsx
const HomeCard = ({ data, config }) => {
    if(data.length > 100) {
      const arr = data.map(item => {
        return {
          date: format(item.hora, 'dd/MM/yy HH:mm'),
          'valores sensores': item.turbidez
        }
      })
      console.log(arr)
    }
    function getOption () {
        return {
            color: ['#b6d7a8'],
            xAxis: {
              type: 'category',
              data: data.map(item => format(item.hora, 'dd/MM/yy HH:mm'))
            },
            yAxis: {
              type: 'value',
              axisLabel: {
                formatter: (item) => `${item}${config.formatter ? config.formatter : ''}`
              }
            },
            tooltip: {
                trigger: 'axis',
                formatter: `${config.name}:  {c}${config.formatter ? config.formatter : ''} <br/> {b}`
            },
            dataZoom: [
                {
                  type: 'slider',
                  start: 0,
                  end: 100,
                  fillerColor: '#b6d7a8'
                },
                {
                  start: 0,
                  end: 20
                }
              ],
            series: [
              {
                data: data.map(item => item[config.varName]),
                type: 'line'
              }
            ]
          }
    }
    return (
        <>
            <div className={styles.card_main}>
                <div className={styles.card_header}>
                    <h1>{config.name}</h1>
                    <div className={styles.value_container}>
                        <div>Tempo real</div>
                        <div className={styles.value_box}>{config.value}</div>
                    </div>
                </div>
                <div className={styles.link_container}>
                    {data !== 'loading' && data.length > 0 &&<ReactECharts
                        option={getOption()}
                        notMerge={true}
                        lazyUpdate={true}
                        theme={"theme_name"}
                    />}
                </div>
            </div>
        </>
    );
};

export default HomeCard;
```

ARQUIVO: api.js

```
const apiPath = process.env.REACT_APP_API;

const requestOptions = (method, bodyContent) => {
    let options = {};
    switch (method) {
        case 'POST':
            let postHeaders = {
                'Content-Type': 'application/json'
            }
            options = {
                method: method,
                headers: postHeaders,
                body: JSON.stringify(bodyContent)
            };
            break;
        case 'GET':
            let getHeaders = {
                'Content-Type': 'application/json',
            };
            options = {
                method: method,
                headers: getHeaders
            };
            break;
        case 'DELETE':
            let delHeaders = {
                'Content-Type': 'application/json'
            };
            options = {
                method: method,
                body: JSON.stringify(bodyContent),
                headers: delHeaders
            };
            break;
        case 'PATCH':
            let patchHeaders = {
                'Content-Type': 'application/json'
            };
            options = {
                method: method,
                body: JSON.stringify(bodyContent),
                headers: patchHeaders
            };
            break;
        default:
            let defaultHeaders = {
                'Content-Type': 'application/json'
            }
            options = {
                method: method,
                headers: defaultHeaders,
                body: JSON.stringify(bodyContent)
            };
            break;
    }

    return options;
}

const api = {
    getHomeData: () => {
        return new Promise((resolve, reject) => {
            const options = requestOptions('GET');
            fetch(`http://tcc-jah-backend.herokuapp.com/get-data`, options)
                .then(async response => {
                    const isJson = response.headers.get('content-type')?.includes('application/json');
                    const data = isJson && await response.json();
                    // check for error response
                    if (!response.ok) {
                        // get error message from body or default to response status
                        const error = { error: (data && data.msg) || response.statusText || response.status };
                        resolve(error);
                    }
                    resolve(data.data || data);
                })
                .catch(error => {
                    // don't return anything -> execution goes the normal way
                    console.error('There was an error!', error);
```

```
                });
            })
        },
        dropTable: () => {
            return new Promise((resolve, reject) => {
                const options = requestOptions('GET');
                fetch(`${apiPath || 'http://localhost:3333'}/drop-table`, options)
                    .then(async response => {
                        const isJson = response.headers.get('content-type')?.includes('application/json');
                        const data = isJson && await response.json();
                        // check for error response
                        if (!response.ok) {
                            // get error message from body or default to response status
                            const error = { error: (data && data.msg) || response.statusText || response.status };
                            resolve(error);
                        }
                        resolve(response.ok);
                    })
                    .catch(error => {
                        // don't return anything => execution goes the normal way
                        console.error('There was an error!', error);

                    });
            })
        },
    };

    export { api, apiPath, requestOptions };
```

ARQUIVO: Contato.jsx

```
import React from 'react';
import styles from './styles.module.css';
import contato_foto from "../assets/img/contato-foto.png";

const Contato = () => {
    return (
        <div className={window.innerWidth < 820 ? styles.page_mobile :styles.page}>
            <div className={window.innerWidth < 820 ? styles.contatos_mobile :styles.contatos}>
                {<img className={window.innerWidth < 820 ? styles.contato_foto_mobile :styles.contato_foto} src={contato_foto} />}
                <div className={window.innerWidth < 820 ? styles.contato_texto_mobile :styles.contato_texto}>
                    <p className={styles.meu_nome}>Jackelyne de Souza Carvalho</p>
                    <p>Estudante de Engenharia de Alimentos UFSC</p>
                    <p className={styles.telefone}>(48) 99603-9461</p>
                    <p className={styles.email}>jackelyne1997@gmail.com</p>
                    <p><script type="text/javascript" src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
                            <a id="link" href="https://www.linkedin.com/in/jackelyne-de-souza-carvalho" target="_blank">Linked-in</a>
                            <script type="text/javascript">
                                $('a#link').click(function(){ /* ... action ... */ })
                            </script></p>
                </div>
            </div>
        </div>


    );
};

export default Contato;
```

ANEXO A – Especificações técnicas dos equipamentos

Tabela 2 - Especificações técnicas do Arduino UNO R3

Tensão de operação	5 V
Tensão de entrada	7-12 V
Portas digitais	14 (6 podem ser usadas como PWM)
Portas analógicas	6
Corrente Pinos I/O	40 mA
Corrente Pinos 3,3 V	50 mA
Memória Flash	32 kB (0,5 kB usado no bootloader)
SRAM	2 kB
EEPROM	1 kB
Velocidade do Clock	16 MHz

Fonte: Eletrogate, 2022.

Tabela 3 - Especificações técnicas do sensor de temperatura

Tensão de alimentação	3 a 5,5 VDC
Precisão	±0,5°C de -10°C a +85°C
Lê temperaturas	-55°Ca 125°C
Resolução	De 9 ou 12 bits
Interface	1 fio (1 Wire), ou seja, precisa de somente 1 porta digital
ID único	64 bits
Pino digital	Suporta vários sensores em funcionamento simultaneamente
Alarme	De limite de temperatura
Tempo de saturação	Menor que 750 ms
Ponta de aço	Com 6 mm de diâmetro e 35 mm de comprimento
Diâmetro do cabo	4 mm
Comprimento do cabo	2 m

Fonte: Baú da Eletrônica, 2022.

Tabela 4 - Especificações técnicas do sensor de turbidez

Tensão	5 VDC
Corrente	30 mA (MAX)
Tempo de resposta	<500 ms
Resistência de Isolamento	100 MΩ (Mín)
Saída	Analógica (0 - 4,5 V) ou digital (alto 5 V / baixo 0 V)
Temperatura de Operação	-30°C a 80°C

Fonte: Baú da Eletrônica, 2022.

Tabela 5 - Especificações técnicas do sensor de pH

Tensão de aquecimento	5 ± 0,2 V (AC/DC)
Corrente de trabalho	5 - 10 mA
Faixa de temperatura	0°C a 60°C
Tempo de resposta	5 s
Tempo de sedimentação	60 s
Componente potência	0,5 v
Saída	Analógica
Faixa de medição	0,00 a 14,00 pH
Zero pontos	7 ± 0,5 pH
Erro alcalino	0,2 pH
Temperatura de operação	-10°C a 50°C (temperatura nominal de 20°C)
Umidade de operação	95% RH (umidade nominal 65% RH)
Vida útil	3 anos
Resistência interna	<250 MΩ
Comprimento do cabo	90 cm
Tamanho do módulo	66 mm largura x 33 mm profundidade x 20 mm altura
Tamanho do sensor	160 mm largura x 26 mm profundidade x 26 mm altura
Peso	82 g

Fonte: Saravati, 2022.

Tabela 6 - Especificações técnicas do sensor de luminosidade

Resistência na luz em 10 Lux	8 a 20 KΩ
Resistência no escuro em 0 Lux	1 MΩ (Mín)
Pico de resposta espectral (25°C)	540 nm
Valor gamma em 100 a 10 Lux	0,7
Tensão máxima	150 VDC
Potência máxima	100 mW
Temperatura de operação	-30°C a 70°C
Comprimento com terminais	32 mm
Tamanho	5 mm largura x 4 mm profundidade x 3 mm altura
Peso	0,4 g

Fonte: Saravati, 2022.

Tabela 7 - Especificações técnicas do NodeMCU

MCU integrado	32 bits de baixa frequência
Conversor analógico digital	10 bits integrado
Clock	80 a 160 MHz
Interface USB-serial	CH340G
Modos de operação	STA/AP/STA+AP
Tensão de operação	4,5 a 9 V
Taxa de transferência	110 a 460.800 bps
Tamanho	50 mm largura x 27 mm profundidade x 7 mm altura

Fonte: Saravati, 2022.

Tabela 8 - Especificações técnicas do ESP-01

Tensão de operação	4,5 a 5,5 VDC
Nível de interface lógica	3,3 e 5 V
Pinos	RX, TX, VCC, GND
Tamanho	48 mm largura x 25 mm profundidade x 12 mm altura
Peso	0,4 g

Fonte: Saravati, 2022.