CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UNIEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

GLEIDSON BONIFÁCIO SILVA
SÁVIO FIGUEIREDO DE MORAIS

HIDROPONIA: MONITORAMENTO REMOTO

ANÁPOLIS - GO
2019

GLEIDSON BONIFÁCIO SILVA
SÁVIO FIGUEIREDO DE MORAIS

HIDROPONIA: MONITORAMENTO REMOTO

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador (a): Prof. Me. Alexandre Moraes Tannús.

ANÁPOLIS - GO
2019

GLEIDSON BONIFÁCIO SILVA
SÁVIO FIGUEIREDO DE MORAIS

HIDROPONIA: MONITORAMENTO REMOTO

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 29 de novembro de 2019, composta por:

Alexandre Moraes Tannus
Alexandre Moraes Tannus
Presidente da Banca

Millys Fabrielle Araujo Carvalhaes
Prof(a). Convidado(a)

William Pereira dos Santos Júnior
Prof(a). Convidado(a)

RESUMO

Este estudo apresenta o desenvolvimento de um sistema de monitoramento automatizado para o cultivo hidropônico, que conta com o acesso remoto via aplicação web das informações extraídas a partir de sensores *arduino*. O objetivo principal do trabalho foi utilizar um circuito microcontrolador que recebe dados coletados a partir de sensores implantados dentro do reservatório do sistema hidropônico. O foco do projeto foi desenvolver uma solução para monitorar remotamente o cultivo através de uma plataforma web, poupando, assim, tempo por parte do hidrocultor. Partindo do princípio que as medições podem ocorrer remotamente, através do sistema desenvolvido, ao invés de localmente, como ocorre no cultivo hidropônico tradicional, a solução desenvolvida proporciona agilidade, praticidade e ganhos produtivos para o hidrocultor.

Palavras Chaves: Automatização, *Arduino,* Cultivo hidropônico, Monitoramento remoto.

ABSTRACT

This study presents the develop of an automation monitoring system for hydroponic cultivation, which relies on remote access via web application of information extracted from arduino sensors. The main purpose of this work was used a microcontroller circuit that receives data collect from sensors implanted inside the hydroponic system reservoir. The focus of the project was develop a solution to monitor cultivation remotely through a web platform, consequently, saving time for the farmer. Assuming that measurements can take place remotely through the developed system rather than locally, as occur in the hydroponic traditional cultivate system, the developed solution provides agility, practicality, and productive gains for the farmer.

Keywords*: Automatization, Arduino, Hydroponic cultivation, Remote monitoring.*

LISTA DE FIGURAS

Página

LISTA DE TABELAS

Página

LISTA DE ABREVIATURAS, SIGLAS E CONVENÇÕES

°C Símbolo de grau Celsius.

A Unidade de Medida Ampére.

AC *Alternating current* (Corrente alternada).

BIT *Binary Digit* (dígito binário).

BNC *Bayonet Neil Concelman* (Conector para cabos coaxiais).

CI Circuito Integrado.

CLOCK Frequência que a CPU realiza os cálculos.

cm Centímetros.

CPU *Central Processing Unit* (Unidade Central de Processamento).

DC *Direct current* (Corrente contínua).

EEPROM *Electrically-Erasable Programmable Read-Only Memory* (Memória Volátil Programável e Apagável Eletricamente).

ETHERNET Arquitetura de interconexão para redes locais.

g Gramas.

GND *Terra*.

HTML *Hyper Text Markup Language* (Linguagem de Marcação de Hipertexto).

HTTP *Hyper Text Transfer Protocol* (Protocolo de Transferência de Hipertexto).

ID Sigla para *identity* (identidade).

IDE *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado).

kB *kilobyte*.

LED *Light Emitting Diode* (diodo emissor de luz).

m Metro.

MA Miliampère, equivalente a um milésimo de ampère.

MHz Mega Hertz**.**

ml Mililitro.

mm Milímetro.

MOHM Megaohm.

MVC Arquitetura Modelo-Visão-Controlador

OHMS Unidade de medida da resistência elétrica.

PH Potencial Hidrogeniônico (mede o grau de acidez de soluções).

PHP *Hypertext Preprocessor* **(**linguagem para desenvolvimento web).

PIN *Personal Identification Number* (Número de Identificação Pessoal).

PWM *Pulse Width Modulation* (Modulação de Largura de Pulso).

R Resistência elétrica medida em *ohm.*

RAM *Random Access Memory* ou (Memória de acesso aleatório).

s Segundo.

SQL *Structured Query Language* (Linguagem de Consulta Estruturada).

SRAM *Static Random*.

UML *Unified Modeling Language* (Linguagem de Modelagem Unificada).

V Símbolo da Unidade de Medida da Tensão Elétrica Volt.

VDD Pino de tensão de alimentação positiva

W Watt.

WEB Sistema de interligação de documentos e recursos através da Internet.

SUMÁRIO

Página

1. INTRODUÇÃO

A hidroponia é uma técnica de cultivo de plantas sem solo, esse tipo destaca-se dos demais devido as vantagens apresentadas, considerando que: A hidroponia é uma técnica de cultivo que visa obter produtos com excelente qualidade, sabor e aspectos externos superiores aos obtidos com agricultura tradicional, oferecendo menor risco de contaminações de doenças endêmicas, (SANTOS et. al., 2002). O cultivo em hidroponia é uma técnica de produção agrícola adequada às exigências de alta qualidade e produtividade com mínimo desperdício de água e nutrientes. Este sistema de cultivo vem crescendo, substancialmente, no Brasil e se apresenta como alternativa, proporcionando maior rendimento e qualidade da produção, bem como a redução da ocorrência de doenças.

Para o sucesso e bom desenvolvimento da cultura hidropônica se faz necessária uma constante manutenção da concentração de nutrientes na solução. Os principais elementos para o crescimento da planta devem estar sempre presentes na solução fornecida, respeitando uma amplitude de variação que se distancie da falta ou do excesso de minerais, (FURLANI et. al., 2009). Ao analisar está técnica de cultivo, percebeu-se à necessidade de monitoramento constante do plantio, para verificar as condições de desenvolvimento das plantas, e realização dos procedimentos de correção e boas práticas de manejo, visando o sucesso da cultura hidropônica, (BEZERRA NETO e BARRETO, 2012).

Logo, a automação na hidroponia tem como objetivo a redução de custos e de erros no controle da produção. Ressaltando que um controle correto dos processos em um sistema de cultivo hidropônico proporciona uma maior produtividade e melhor qualidade do produto, (SANTOS, 2015). Considerando as técnicas de cultivo hidropônico tradicionais, o seguinte problema foi levantado: como otimizar o cultivo hidropônico através do monitoramento remoto utilizando um sistema de hardware e software?

O objetivo principal deste trabalho é o monitoramento remoto do cultivo hidropônico. Este trabalho buscou também: verificar as ferramentas necessárias para o desenvolvimento de um protótipo de monitoramento do cultivo hidropônico; construir um protótipo para monitoramento e medição dos elementos essenciais do cultivo hidropônico; desenvolver uma aplicação Web para auxiliar o hidrocultor na tomada de decisões sobre o cultivo hidropônico; integrar o protótipo de monitoramento ao sistema Web para gerenciamento do cultivo em tempo real; otimizar o plantio através de sistema de hardware e software.

Observando que a demanda por alimentos fez com que crescesse a necessidade de

aumento da produção, exigindo, assim, o aperfeiçoamento e transformação da agricultura, soluções que aprimorem o cultivo se fazem necessárias. Conforme a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA, [s.d.]) coloca que “Para avançar e acompanhar a demanda atual é necessário que o país aumente significativamente a produção”.

Nesse sentido não há como prescindir de tecnologias e conhecimentos da automação e agricultura de precisão que aperfeiçoem e tornem o processo de produção no campo mais eficiente para aumentar o desempenho tanto qualitativo como quantitativo de forma expressiva. Logo, utilizar a automação nos processos da agricultura visa não somente suprir uma necessidade de mercado, mas também aprimorar o produto final (EMBRAPA, [s.d.]).

Portanto criar um processo de monitoramento remoto do cultivo hidropônico, contribui para otimizar a produção e o plantio. Partindo do princípio que esse tipo de cultivo necessita de acompanhamento constante, e todas as informações a respeito do plantio e crescimento das plantas, nesse sistema, precisam ser acompanhadas em tempo real, devido as vulnerabilidades existentes (BEZERRA NETO e BARRETO, 2012).

Assim, a automatização do monitoramento do cultivo hidropônico aumenta a produtividade e qualidade das plantas. Ressaltando algumas das vantagens já encontradas nesse tipo de cultivo: melhor controle sobre a composição dos nutrientes fornecidos às plantas; redução no ciclo da cultura e maior produtividade; menor consumo de água e de fertilizantes; melhor controle fitossanitário; redução em alguns tratos culturais; redução de riscos climáticos; melhor qualidade e preço do produto; produção próximo ao consumo (BEZERRA NETO e BARRETO, 2012).

2. REFERENCIAL TEÓRICO

Este referencial teórico aborda os seguintes assuntos: hidroponia, *arduino*, sensores e desenvolvimento de *software*. Assim atribuindo as especificações técnicas do *hardware* e *software*, além de definir de forma teórica o desenvolvimento do sistema de monitoramento. Para compreender melhor o sistema criado é necessário entender o problema solucionado, bem como as definições técnicas utilizadas no desenvolvimento do estudo.

2.1 Hidroponia

Com a hidroponia, é possível cultivar, de maneira bastante adensada, hortaliças e outros vegetais, alcançando-se um grau de produtividade bastante elevado. É muito útil em locais onde não há disponibilidade de solo adequado para o cultivo, ou para regiões de climas mais inóspitos, pois o cultivo hidropônico pode ser feito em locais fechados, como estufas (RURALNEWS, 2017). Sendo assim a hidroponia é extremamente viável para a produção de alimentos em locais em que o solo não provém os minerais e substancias que determinados vegetais necessitam para se desenvolver.

Ao levantar os dados sobre o cultivo hidropônico observou-se que os cuidados com a solução aquosa e com a maneira com que as plantas são manipuladas devem ser feitos de maneira extremamente eficiente para conseguir vegetais saudáveis e de alta qualidade. Assim como afirma Melonio (2012, n. p.) que "[...] Qualquer falha ou erro de manejo pode acarretar um prejuízo bem maior e mais grave do que na agricultura tradicional, pois o sistema hidropônico é muito mais vulnerável".

2.2 *Black board* R3 e A*rduino* Mega 2560 R3

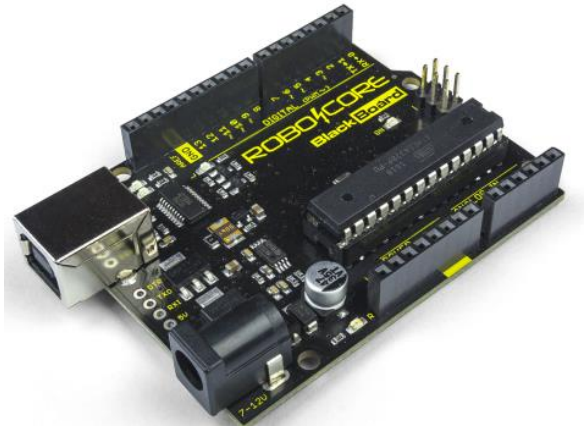
O *arduino* é uma plataforma de computação física de código aberto para a criação de objetos interativos independentes ou em colaboração com o *software* de computador. Ele foi projetado para artistas, designers e outros profissionais que queiram incorporar a computação física e seus projetos sem que para isso precisem ter se formado em engenharia elétrica (BANZI e SHILOH, 2015). Segundo Thomsen (2014, n. p.)

> O *Arduino* foi criado em 2005 por um grupo de 5 pesquisadores: Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de hardware livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o *Arduino*, partindo do mesmo hardware básico.

O *arduino* é uma plataforma *open source* de prototipagem eletrônica com *hardware* e *software* flexíveis e fáceis de usar, destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos (MOTA, 2017). De acordo com Tecnoveste (2016, n. p.) "*Open Source*, ou Código aberto, é um modelo de desenvolvimento que promove o licenciamento livre para design e desenvolvimento de produtos, assegurando aos seus criadores os créditos e aos usuários o direito de uso, redistribuição e adaptação [...]".

Sendo assim, foi possível constatar que a plataforma *arduino* trouxe maior praticidade para que qualquer pessoa que tenha um pequeno domínio de informática possa ser capaz de criar seus projetos, possibilitado justamente por ser uma plataforma de *open source*. Na figura 1, abaixo, temos a *blackboard* que inicialmente foi utilizada no desenvolvimento do estudo. Essa teve por função controlar os dados e armazenar o código com a lógica de programação responsável por gerir o protótipo que foi proposto.

Figura 1 - *Black board*.



Fonte: (ROBOCORE, 2019a).

A placa *Black Board*, é uma placa *arduino* fabricada pela empresa RoboCore no Brasil, ela é bem similar a placa *arduino* Uno R3, porém contando com algumas melhorias feitas pela equipe da RoboCore (SOUZA, 2014a). A tabela 1, abaixo, é composta pelos dados técnicos da placa *arduino* representada na figura 1, na página anterior.

Tabela 1 - Dados técnicos da *black board* uno R3.

Dimensões	68 x 53 x 10 mm
Microcontrolador	ATmega328P
Memória Flash	32 KB (dos quais 0,5 KB são usados pelo *bootloader*).
Memória SRAM	2 KB
Memória EEPROM	1 KB
Frequência de *clock*	16 MHz
Tensão de operação	5 V
Tensão de alimentação	7 a 12 V (recomendada)
Saídas digitais I/O Pin:	20 (dos quais 6 oferecem saída PWM)

Fonte: (ROBOCORE, 2019b).

A linguagem de programação utilizada no *arduino* é a linguagem C++ (com pequenas modificações), que é uma linguagem muito tradicional e conhecida (CHAVIER, 2019). Por ser uma linguagem muito popular acaba facilitando o desenvolvimento do código do projeto.

Segundo Souza (2014a, n. p.) "A *BlackBoard* é capaz de fornecer maior corrente em seu pino de alimentação 5 Volts para circuitos externos quando comparada com a placa *arduino* UNO [...]". Com o decorrer do estudo, durante a fase de desenvolvimento e sincronização do sistema web proposto com a placa *arduino*, percebeu se que a placa *arduino BlackBoard* não possuía a quantidade de pinos necessárias para a execução e sincronização dos periféricos *arduinos*, por este motivo foi necessário a busca por uma placa microcontroladora de maior capacidade.

A placa *arduino* mega 2560 foi selecionada para fazer a substituição. Na tabela 2 estão dispostas todas as especificações técnicas do *arduino* mega, representado pela figura 2, ambas na página seguinte.

Figura 2 - *Arduino* mega 2560 R3.

Fonte: (SOUZA, 2014b).

Tabela 2 - Dados técnicos do *arduino* mega 2560 R3.

Dimensões	120x53x10mm
Microcontrolador	ATmega2560
Memória Flash	256kb
Memória SRAM	8 KB (ATmega2560)
Memória EEPROM	4KB (ATmega2560)
Frequência de clock	12MHz
Tensão de operação	5V
Tensão de alimentação	7-12V
Saídas digitais I/O Pin:	54 (dos quais 14 oferecem saída PWM)

Fonte: (SOUZA, 2014b).

Sendo assim, a placa *arduino* mega 2560 foi selecionada pois possui 54 pinos de entradas e saídas digitais onde 15 destes podem ser utilizados como saídas PWM, possui também 16 entradas analógicas e 4 portas de comunicação serial (SOUZA, 2014b).

2.3 Bomba de água RS-385

Ao verificar quais equipamentos seriam necessários para a execução do estudo, constatou-se a necessidade de uma bomba de água para circular e gerar oxigênio na água. Pois um dos primeiros sintomas decorrentes da falta de oxigenação em uma cultura hidropônica é o apodrecimento das raízes, que faz com que a planta tenha dificuldades em absorver os nutrientes, provocando assim a morte desta. Da mesma forma, o aparecimento de micro-

organismos que passam para o sistema fechado de circulação de água, causam os mesmos problemas as plantas (GROHO, [s.d.]).

Segundo Balduzzi (2014, n. p.) “[...] a oxigenação da solução é muito importante. É preciso utilizar uma boa água e oxigenar a solução constantemente para obter um bom nível de absorção dos nutrientes [...]”. A oxigenação pode ser feita durante a circulação da solução no retorno ao reservatório ou com a aplicação de ar comprimido ou oxigênio.

Com base nessas informações foi necessário a utilização de uma bomba de água. Após um estudo de qual seria a melhor alternativa para o projeto executado, o modelo RS-385 foi a melhor opção para o mesmo, na figura 3, abaixo, está disposta a imagem da bomba de água RS-385.

Figura 3 - Minibomba de água para irrigação 12V.

Fonte: (MULTIPEÇAS, 2019).

A Minibomba de Água RS385 foi criada especialmente para o desenvolvimento de projetos de prototipagem, incluído automação residencial (demótica) e protótipos robóticos baseados em plataformas microcontroladoras. Com um motor de tamanho adequado a Minibomba de Água RS385 é capaz de impulsionar entre 1500ml a 2000ml por minuto, sendo destacada pela sua eficiência e precisão durante sua execução em conjunto com o *arduino*, por exemplo (UZINAINFO, 2019a). Na tabela 3, na página seguinte, estão as especificações técnicas da bomba RS385 conforme os dados do fornecedor.

Tabela 3 - Dados técnicos da bomba de água RS385

Modelo	RS385
Materiais	Metal e plástico
Tensão nominal	12V
Voltagem adequada	DC 9 a 15V;
Corrente em carga	0,6ª
Corrente em máxima eficiência	~2ª
Altura de aspiração máxima	2m
Elevação máxima	3m
Vazão de água máxima	~1,5 a 2 l/m
Diâmetro de entrada e saída	~7,6mm
Diâmetro do motor	28,6mm
Comprimento da bomba	90mm

Fonte: (UZINAINFO, 2019a).

2.4 Sensor DS18B20

O sensor DBS18B20 tem a função monitorar a temperatura da solução nutritiva, pois como afirma Lonax-admin (2019a) "[...] a temperatura da água deve ser mantida em torno dos 25 °C e não ultrapassar os 28 °C. Porém, dependendo da hortaliça cultivada, esses valores podem ser ajustados. Em regiões mais quentes, os vegetais absorvem uma quantia maior de água do que de nutrientes e é preciso oferecer líquidos nutritivos mais diluídos.

Na figura 4, na página seguinte, apresenta-se a imagem do sensor DS18B20 que foi utilizado no projeto. O sensor de temperatura DS18B20 é um componente eletrônico digital desenvolvido para ser aplicado nos mais diversos ambientes, pois é capaz de medir a temperatura em locais úmidos, inclusive estando submerso na água. Para que ele entre em funcionamento é necessário estar conectado junto a uma plataforma de prototipagem, por exemplo, o *arduino* (UZINAINFO, 2019b). Ao analisar o sensor representado na figura 4 foi constatado que por ser a prova d'agua ele seria viável para medir a temperatura da solução aquosa. A tabela 4, na página seguinte, especifica os dados técnicos do sensor DS18B20 ilustrado na figura 4.

Figura 4 - Sensor DS18B20 à prova d'água.

Fonte: (ROBOCORE, 2019c).

Tabela 4 - Dados técnicos do sensor de temperatura DS18B20.

Dimensão da ponta	Ponta com 6mm de diâmetro e 50mm de comprimento
Diâmetro do cabo	4mm
Comprimento do cabo	95cm
Tensão de alimentação	3.0 VDC a 5.5 VDC
Precisão	±0.5°C de -10°C a +85°C
Variações de temperaturas	-55°C a +125°C
Resolução	9 ou 12 bits
Interface 1 fio (1 Wire)	precisa de somente 1 porta digital
ID	único de 64 bits

Fonte: (ROBOCORE, 2019c).

Na tabela 5, na página seguinte, está a descrição de cada pino referente ao sensor de temperatura. Onde a faixa de valor de tensão de alimentação desse sensor é de 3V à 5,5V. Para esse sensor é necessária apenas a conexão da linha de dados e do ponto de referência zero (GND) da alimentação do dispositivo Mestre 1-Wire, utilizando, assim, o modo de alimentação parasita (o qual opera com um capacitor que é carregado enquanto a linha de dados está em nível alto e supre a corrente necessária para o sensor nos momentos em que a linha se encontra em nível baixo). Porém o usuário também pode usar o pino VDD para alimentação com um ponto de tensão que esteja entre os valores de operação (APRENDENDOFISICA, 2019).

Tabela 5 - Descrição dos pinos do DS18B20.

Pino	Descrição
1-GND	Ligação do CI ao terra de alimentação
2-DQ	3.0 VD Pino bidirecional de transição de dados do barramento 1-WireC a 5.5 VDC
3-VDD	±0.5°C de -10°C a +85°C Ligação opcional a uma tensão de 3V à 5,5V. Caso o DS18B20 esteja operando no modo de operação parasita, este pino deve ser conectado ao terra.

Fonte: (OLARIA.UCPEL, 2019).

2.5 Sensor de Ph BNC

Para garantir que a solução nutritiva tenha a concentração ideal para que a planta possa adquirir todos os nutrientes que ela necessite foi utilizado o modelo de sensor de Ph que se encontra na figura 5, abaixo, para medir o nível de acidez da água.

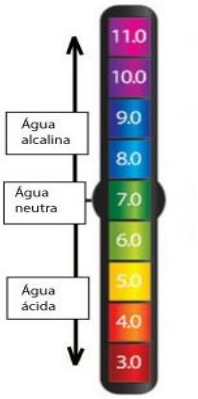
O Sensor de PH *arduino* é indicado para verificar e permitir a manutenção dos níveis de PH dentro de escalas seguras, pode ser usado em piscinas, aquários, caixas de água, rios, lagos e muito mais, permitindo manter proporções aceitáveis para pessoas, animais e plantas (ARDUINO-SANTA-EFIGENIA, 2019a). Segundo Lonax-admin (2018a, n. p.) “ [...] Substâncias com PH entre 0 e 7 são ácidas, Já aquelas que têm PH igual a 7 são neutras, e valores acima de 7 indicam o caráter básico da solução”. O monitoramento do PH é uma atividade que não pode faltar em uma hidrocultura, à medida que os nutrientes são absorvidos, o PH da solução é alterado.

Figura 5 - Sensor de PH *arduino* com módulo de leitura.

Fonte: (ARDUINO-SANTA-EFIGENIA, 2019a).

Na figura 6, abaixo, está descrito o parâmetro utilizado pelo sensor de PH para medir o nível de acidez da água:

Figura 6 - Parâmetro para a leitura da acidez da água.



Fonte: (ARDUINO-SANTA-EFIGENIA, 2019a).

Na tabela 6, abaixo, está descrita as especificações técnicas do sensor de PH com módulo BNC ilustrado na figura 5, na página anterior.

Tabela 6 - Especificação técnica do leitor de PH com módulo BNC.

Modelo	PH-4502C
Tensão de aquecimento	5 ±0.2V (AC/DC)
Corrente de trabalho	5-10mA
Faixa de temperatura	0-60℃
Tempo de resposta	5S
Tempo de sedimentação	60S
Potência do componente	0,5 W
Saída	Analógica
Faixa de medição	0,00 ~ 14,00 pH
Zero pontos	7 +-0.5ph
Erro alcalino	0.2pH
Resistência interna	250MOhm
Blocos de terminais	Plug BNC
Comprimento do cabo	1 metro
Dimensões do módulo (CxLxE)	2x32x13mm
Dimensões do sensor (CxD)	170×12,5mm
Peso	75g

Fonte: (ARDUINO-SANTA-EFIGENIA, 2019a).

Na tabela 7, abaixo, está descrita a pinagem do sensor de PH com módulo BNC ilustrado na figura 5, na página 22.

Tabela 7 - Descrição da pinagem do leitor de Ph com módulo BNC.

Pino	Descrição
TO	saída de temperatura.
DO	Saída de 3.3V (do potenciômetro do limite do pH).
PO	Saída analógica Ph - Arduino A0.
GND	Gnd para sonda PH (pode vir do pino GND do Arduino).
GND	Gnd para placa (também pode vir do pino GND do Arduino).
VCC	5V DC (pode vir do pino Arduino 5V).
POT 1	Deslocamento de leitura analógica (próximo ao BNC).
POT 2	Configuração de limite de PH.

Fonte: (ARDUINO-SANTA-EFIGENIA, 2019b).

2.6 Sensor de nível de água

Para controlar o nível de água no reservatório e garantir que o nível do mesmo sempre permaneça aceitável, foram utilizados 3 sensores do tipo boia, como o que está apresentado na figura 7, abaixo.

Figura 7 - Sensor de nível de agua tipo boia vertical.

Fonte: (VIDA-DE-SILICIO, 2019).

Na tabela 8, abaixo, estão as especificações do sensor do tipo boia ilustrado na figura 7, na página anterior.

Tabela 8 - Especificação técnica sensor tipo boia.

Comprimento total	508mm.
Corrente de chaveamento (máx.)	0,5A.
Tensão do contato aberto (máx.)	220V AC.
Resistência Contato (máx.)	100M Ohms.
Temperatura de operação	-20 ~ +85°C.
Extensão do fio	35cm.
Diâmetro da bóia	20mm.

Fonte: (VIDA-DE-SILICIO, 2019).

A tabela 9, abaixo, exibe a pinagem do sensor do tipo boia, especificando a função de cada pino durante a ligação na placa microcontroladora *arduino* da tabela 9 (página 16).

Tabela 9 - Descrição da pinagem do sensor do tipo boia.

GND	Ligação do CI ao terra de alimentação.
VCC	5V DC (pode vir do pino Arduino 5V).
PORTA DIGITAL	Conexão dos dados transmitidos.

Fonte: (AUTORES, 2019).

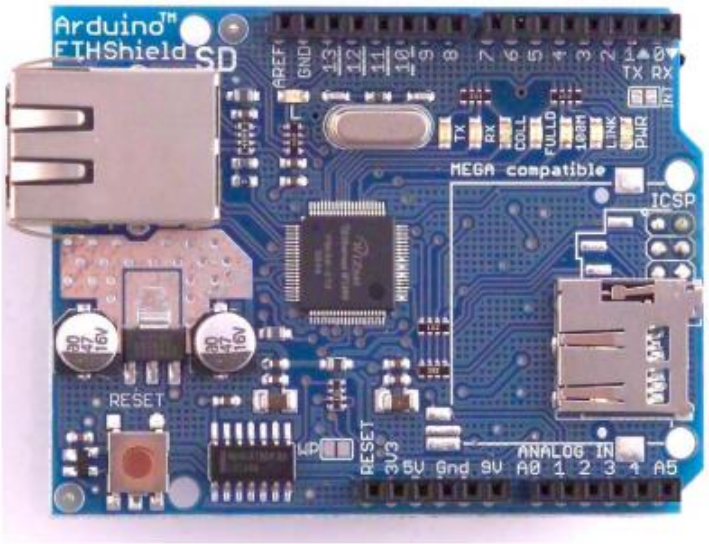
2.7 *Shield Ethernet* w5100

Para sincronizar todos os sensores que compuseram o projeto foi necessário enviar os dados da placa microcontroladora *arduino* para o sistema web, para este propósito foi utilizado o *shield ethernet* w5100, ilustrado na figura 8, na página seguinte.

O *shield Ethernet* é uma placa *arduino* que permite conexão à internet. Ele é baseado no Wiznet W5100 chip de *Ethernet*. Ele suporta até quatro conexões de soquete simultâneas, além de possibilitar o uso da biblioteca *ethernet* para escrever esboços que se conectam à internet usando o *shield* (ARDUINO et al., 2010). Segue na Tabela 10, na página seguinte, os

dados técnicos do *shield ethernet*.

Figura 8 - *Shield Ethernet* w5100.



Fonte: (ARDUINO et al., 2010).

Tabela 10 - Dados técnicos do *shield Ethernet* w5500.

CPU	w5500 32-bit processo
Tensão operacional	5V
Corrente de operação	Valor médio 80 mA
Tamanho do pacote	QFN32-pin (5 mm x 5 mm)

Fonte: (ARDUINO et al., 2010).

2.8 Monitoramento remoto no cultivo hidropônico

Segundo Sperafico (2018, n. p.) “Os benefícios da tecnologia moderna para os setores produtivos e a sociedade são cada vez mais decisivos para a elevação da produtividade agropecuária, da lucratividade do agronegócio, da produção de alimentos e do bem-estar da população urbana e rural”. Desse modo, conclui-se que quanto mais tecnologia é empregada no agronegócio maior o ganho, seja para qual for o tipo de cultivo.

Levando em consideração que a hidroponia requer pessoal capacitado e atento, além de

precisar monitorar constantemente os indicadores da solução nutritiva, o desenvolvimento de um sistema para transmitir tais informações remotamente faz se necessário. Pois, todos os envolvidos no cultivo devem se manter antenados as melhores práticas além de possuírem equipamentos de qualidade para o trabalho (LONAX-ADMIN, 2018b).

2.9 Engenharia de s*oftware*

A engenharia de *software* tem por objetivo apoiar o desenvolvimento profissional de *software*, mais do que a programação individual. Ela inclui técnicas que apoiam especificação, projeto e evolução de programas, que normalmente não são relevantes para o desenvolvimento de *software* pessoal (SOMMERVILLE, 2011). A engenharia de *software* é ainda, e um processo abrangente, um conjunto de métodos (práticas) e um leque de ferramentas que possibilitam aos profissionais desenvolverem *software* de altíssima qualidade (PRESSMAN, 2011).

Muitas pessoas pensam que *software* é simplesmente outra palavra para programas de computador (SOMMERVILLE, 2011). No entanto, quando falamos de engenharia de *software*, não se trata apenas do programa em si, mas de toda a documentação associada e dados de configurações necessárias para fazer o programa operar corretamente. Um sistema de *software* desenvolvido profissionalmente é, com frequência, mais do que apenas um programa; ele normalmente consiste em uma série de programas separados e arquivos de configuração que são usados para configurar esses programas. Isso pode incluir documentação do sistema, que descreve a sua estrutura; documentação do usuário, que explica como usar o sistema; e sites, para usuários baixarem a informação recente do produto (PRESSMAN, 2011).

2.9.1. Processos de *software*

Segundo Pressman (2011, p. 52) o processo de *software* é o conjunto de "[...] atividades, ações e tarefas necessárias para desenvolver um *software* de qualidade [...]". É importante seguir passos previsíveis na elaboração de um sistema ou produto que garanta um resultado de alta qualidade (PRESSMAN, 2011). Esses passos que devem ser seguidos é denominando de processo de *software*. Os processos de *software*, às vezes, são categorizados como dirigidos a planos ou processos ágeis. Processos dirigidos a planos são aqueles em que todas as atividades são planejadas com antecedência, e o progresso é avaliado por comparação com o planejamento inicial (SOMMERVILLE, 2011).

2.9.2. Modelo de Processo de software

Um modelo de processo de *software* é uma representação simplificada de um processo de *software*. Cada modelo representa uma perspectiva particular de um processo e, portanto, fornece informações parciais sobre ele. Por exemplo, um modelo de atividade do processo pode mostrar as atividades e sua sequência, mas não mostrar os papéis das pessoas envolvidas (SOMMERVILLE, 2011).

2.9.3. Modelo de processo prototipagem

Um protótipo é uma versão inicial de um sistema de *software*, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções (SOMMERVILLE, 2011). O desenvolvimento rápido e iterativo do protótipo é essencial para que os custos sejam controlados e os *stakeholders* do sistema possam experimentá-lo no início do processo de *software*. Um protótipo de *software* pode ser usado em um processo de desenvolvimento de *software* para ajudar a antecipar as mudanças que podem ser requisitadas. No processo de engenharia de requisitos, um protótipo pode ajudar na elicitação e validação de requisitos de sistema já no processo de projeto de sistema, um protótipo pode ser usado para estudar soluções específicas do *software* e para apoiar o projeto de interface de usuário (SOMMERVILLE, 2011).

2.10 Engenharia de requisitos

Engenharia de requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema. A engenharia de requisitos é um estágio particularmente crítico do processo de *software*, pois erros nessa fase inevitavelmente geram problemas no projeto e na implementação do sistema (SOMMERVILLE, 2011).

2.11 Levantamento de Requisitos

O levantamento de requisitos (também chamado elicitação de requisitos) combina

elementos de resolução de problemas, elaboração, negociação e especificação. Para encorajar uma abordagem colaborativa e orientada às equipes em relação ao levantamento de requisitos, os interessados trabalham juntos para identificar o problema, propor elementos da solução, negociar diferentes abordagens e especificar um conjunto preliminar de requisitos da solução (PRESSMAN, 2011). No levantamento de requisitos ocorre uma derivação dos requisitos do sistema por meio da observação dos sistemas existentes, além de discussões com os potenciais usuários e compradores, análise de tarefas, entre outras etapas (SOMMERVILLE, 2011). Essa parte do processo pode envolver o desenvolvimento de um ou mais modelos de sistemas e protótipos, os quais ajudam a entender o sistema a ser especificado.

2.11.1. Requisitos funcionais

São declarações de serviços que o sistema deve fornecer, de como este deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer (SOMMERVILLE, 2011).

2.11.2. Requisitos Não Funcionais

Os requisitos não funcionais, como o nome sugere, são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários, eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área (SOMMERVILLE, 2011).

2.12 UML

A UML (*Unified Modeling Language* ou Linguagem de Modelagem Unificada) é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de Orientação a Objetos. Esta linguagem tornou-se a linguagem-padrão de modelagem de *software* nos últimos anos, adotada internacionalmente pela indústria de engenharia de *software* (GUEDES, 2011). Cada diagrama da UML analisa o sistema, ou parte dele, sob uma determinada óptica, o objetivo é fornecer múltiplas visões do sistema a ser modelado,

analisando-o e modelando-o sob diversos aspectos, procurando-se, assim, atingir a completude da modelagem, permitindo que cada diagrama complemente os outros (GUEDES, 2011).

É como se o sistema fosse modelado em camadas, sendo que alguns diagramas enfocam o sistema de forma mais geral, apresentando uma visão externa do sistema, como é o objetivo do Diagrama de Casos de Uso, enquanto outros oferecem uma visão de uma camada mais profunda do *software*, apresentando um enfoque mais técnico ou ainda visualizando apenas uma característica específica do sistema ou um determinado processo. A utilização de diversos diagramas permite que falhas sejam descobertas, diminuindo a possibilidade da ocorrência de erros futuros (GUEDES, 2011).

2.12.1. Diagrama de caso de Uso

O Diagrama de Casos de Uso é o diagrama mais geral e informal da UML, utilizado normalmente nas fases de Levantamento e Análise de Requisitos do sistema, embora venha a ser consultado durante todo o processo de modelagem e possa servir de base para outros diagramas (PRESSMAN, 2011). Apresenta uma linguagem simples e de fácil compreensão para que os usuários possam ter uma ideia geral de como o sistema irá se comportar. Esse diagrama procura identificar os atores (usuários, outros sistemas ou até mesmo algum *hardware* especial), que utilizarão de alguma forma o *software*, bem como os serviços, ou seja, as opções que o sistema disponibilizará aos atores, conhecidas neste diagrama como casos de uso (GUEDES, 2011).

O primeiro passo ao escrever um caso de uso é definir o conjunto de “atores” envolvidos na história. Atores são as diferentes pessoas (ou dispositivos) que usam o sistema ou produto no contexto da função e comportamento a ser descritos. Os atores representam os papéis que pessoas (ou dispositivos) desempenham enquanto o sistema opera (PRESSMAN, 2011). Definido de maneira um pouco mais formal, ator é qualquer coisa que se comunica com o sistema ou o produto e que é externa ao sistema em si. Todo ator possui uma ou mais metas ao usar o sistema.

2.12.2. Diagrama de classe

O Diagrama de classes é o mais utilizado e o mais importante da UML. Serve de apoio para a maioria dos demais diagramas. Como o próprio nome diz, define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe possui, além de estabelecer como as classes se relacionam e trocam informações entre si (GUEDES, 2011).

2.13 PHP

A abreviação PHP vem de "*Hypertext PreProcessor*", que é uma linguagem de programação de código aberto muito utilizada para a criação de *scripts* que são executados no servidor web para a manipulação de páginas HTML (*Hyper Text Markup Language* ou Linguagem de Marcação de Hipertexto) (ARROYO e SANTOS, 2002, p5).

Arroyo e Santos (2002, p.5) apresentam inúmeras vantagens na utilização de PHP para o desenvolvimento web, tais como:

- É uma linguagem de fácil aprendizado.
- Tem performance e estabilidade excelentes.
- Seu código é aberto, não é preciso pagar por sua utilização, e é possível alterá-lo na medida da necessidade de cada usuário.
- Tem suporte nos principais servidores web do mercado, principalmente no servidor web Apache (o mais utilizado no mundo).
- Suporta conexão com os bancos de dados mais utilizados do mercado, como por exemplo, MySQL, PostgreSQL, Oracle e DB2.
- É multiplataforma, tem suporte nos sistemas operacionais mais utilizados no mercado.
- Suporta uma variedade grande de padrões e protocolos, como o XML, DOM, IMAP, POP3, LDAP, HTTP, entre outros.
- Não precisa ser compilado.

Praciano (2014) acrescenta que "[...] Além de ser muito fácil de instalar, em qualquer plataforma (inclusive Windows), o PHP foi projetado para rodar sobre o Linux e o Apache ambos de código aberto e livres". Por fim, devido as vantagens apresentadas, o PHP foi escolhido para o desenvolvimento do projeto.

2.14 Laravel

Laravel é um *framework* PHP utilizado para o desenvolvimento web, que utiliza a arquitetura MVC e tem como principal característica desenvolver aplicações seguras e performáticas de forma rápida, com código limpo e simples, já que ele incentiva o uso de boas práticas de programação e utiliza o padrão PSR-2 como guia para estilo de escrita do código.

Para a criação de interface gráfica, o Laravel utiliza uma *engine* de *template* chamada *Blade*, que traz uma gama de ferramentas que ajudam a criar interfaces bonitas e funcionais, de forma rápida e evitando a duplicação de código (WENDELL, 2015). O *Blade* traz uma gama de ferramentas para auxiliar na criação de interfaces gráficas ricas e funcionais, com o conceito de evitar a duplicação de código (ADRIEL, 2015).

O *framework* também se utiliza de uma interface de linha de comando, chamada de Artisan CLI. Esta interface permite realizar diversas ações na aplicação, dentre elas, a configuração de ambiente, verificação de rotas, interação com aplicação e a criação de vários tipos de arquivos, como: *migrations*, *controller*, etc. (ADRIEL, 2015). Com as *migrations* o Laravel define como o banco de dados deve ser criado através de arquivos PHP. Através do Artisan, o desenvolvedor pode criar, alterar ou excluir tabelas no banco de dados de forma rápida, fácil e intuitiva (ADRIEL, 2015).

2.15 MySQL

MySQL é um sistema *open source* de gerenciamento de banco de dados relacional que utiliza SQL (*Structured Query Language*), uma linguagem padrão para acessar sistemas de gerenciamento de banco de dados relacional (*Relational Database Management Systems* - RDBMS). Por ser uma ferramenta de código aberto, o MySQL é grátis e pode ser adaptado de acordo com as necessidades do desenvolvedor (WELLING e THOMSON, 2005).

Esse sistema foi escolhido para o projeto devido a facilidade de integração com o PHP, o qual foi utilizado para o desenvolvimento do projeto. Outro motivador da escolha foi a facilidade do acesso à informação, partindo do princípio de que já existe uma vasta documentação online e fóruns de debate voltados para o assunto.

2.16 Apache

O Apache é um servidor de código aberto e nome oficial é Apache HTTP *Server*, mantido pela Apache *Software Foundation*, e alimenta cerca de 46% de todos os sites hospedados na internet. O Apache permite que donos de sites mostrem e mantenham seus conteúdos na internet – daí o nome de "servidor de internet".

O Apache é um dos mais antigos e confiáveis servidores de internet, e também um servidor físico. Ele é um *software* que é executado em um servidor. A função deste *software* é estabelecer uma conexão entre o servidor e os navegadores de sites (Firefox, Google *Chrom*e, etc.) enquanto puxa e entrega arquivos entre eles (estrutura cliente-servidor). O Apache é um *software* multiplataforma. Portanto, ele funciona tanto em servidor Unix quanto em servidor Windows (ANDREI, 2019). A escolha do Apache para utilização no projeto ocorreu por suportar a linguagem PHP e o banco de dados MySQL.

2.17 Heroku

O Heroku foi criado em 2007 por três desenvolvedores norte-americanos: James Lindenbaum, Adam Wiggins e Orion Henry. Inicialmente o suporte era apenas para aplicativos desenvolvidos em *Ruby*, rodando no servidor web Rack. Em 2010 o projeto foi adquirido pela *Salesforce* e em 2011 iniciou um processo para suportar outras linguagens e *frameworks*. Atualmente o Heroku suporta *Ruby*, Java, Clojure, Python, Scala, PHP e Node e possui mais de três milhões de aplicações instaladas (DANIEL, 2013).

O Heroku funciona da seguinte maneira, ao realizar um *deploy* no Heroku ele empacota o código e as dependências do aplicativo em containers que são ambientes leves e isolados que fornecem processamento, memória, sistema operacional e um sistema de arquivos ("HEROKU", 2019). A escolha do Heroku para o desenvolvimento do sistema ocorreu pois o mesmo opera com elevado grau de escalabilidade e segurança.

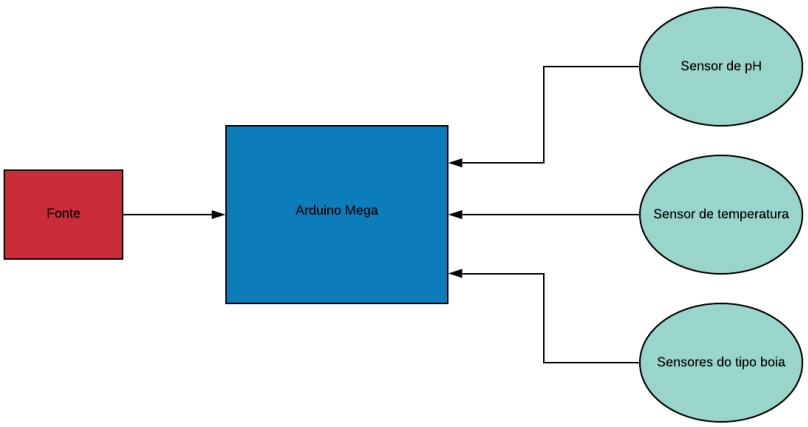
3. METODOLOGIA

Está pesquisa foi realizada com o propósito de desenvolvimento deste projeto. Os autores fizeram pesquisas cientificas e tecnológicas detalhadas e aprofundadas sobre os tipos de medição de sensores juntamente com tecnologias de compartilhamento de dados via rede *Ethernet*, além de pesquisas sobre hidroponia. Assim como foram realizados estudos e pesquisas de diversas tecnologias e linguagens como *arduino*, C++, PHP, UML e MySQL, possibilitando assim tanto o desenvolvimento do protótipo, que realiza as medições do cultivo hidropônico, quanto do sistema web de monitoramento remoto.

Para o desenvolvimento do protótipo foi necessário o uso de metodologias que facilitassem tanto a codificação como a montagem dos periféricos, desse modo, foram utilizados fluxogramas para simplificar a visualização da lógica de programação. Como também foram desenvolvidos diagramas de blocos para obter uma visualização esquemática simplificada do protótipo, de modo a facilitar a montagem dos sensores a placa microcontroladora.

Para desenvolver o protótipo de medição, primeiramente foi necessário testar todos os sensores, para isso a primeira parte do projeto que foi apresentado no trabalho de conclusão de curso 1 consistiu em realizar medições com amostragem de dados locais. Assim, foi necessário gerar fluxogramas e diagramas de blocos que permitiram abstrair a lógica *arduino* que viria a ser codificada. Na figura 9, abaixo, está o diagrama de blocos que foi utilizado para obter a visão esquemática do protótipo de teste dos sensores.

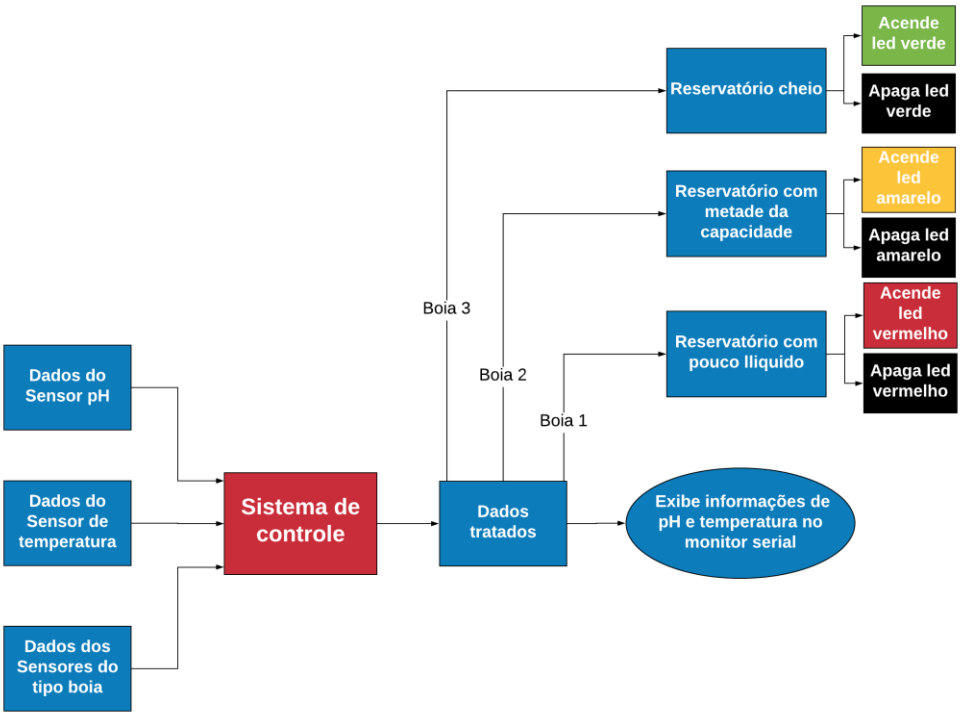
Figura 9- Diagrama de blocos para teste dos sensores.



Fonte: (AUTORES, 2019).

Enquanto que o fluxograma apresentado na figura 10, abaixo, foi utilizado para gerar a lógica de programação *arduino* necessária para testar os sensores, utilizando LEDs e o monitor serial da IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) *arduino* para exibir os dados do cultivo hidropônico localmente.

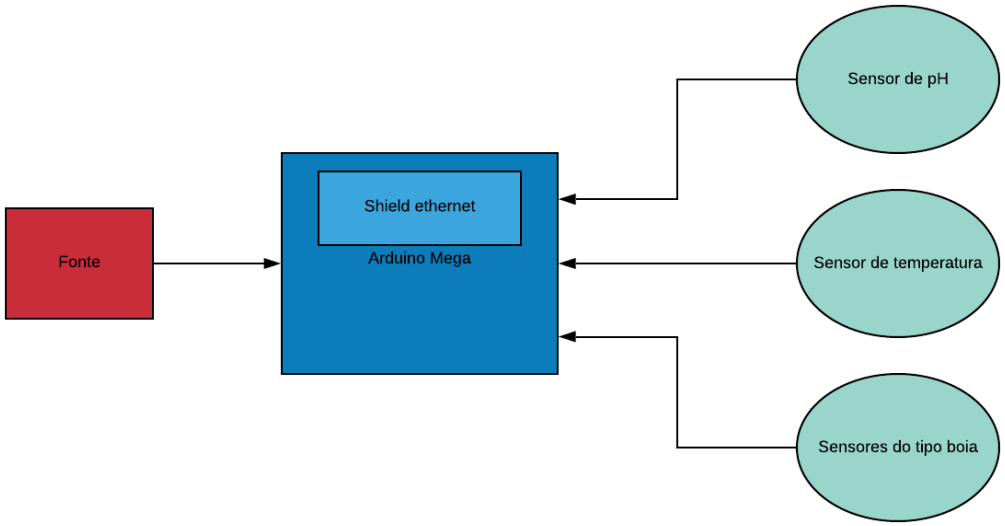
Figura 10 - Fluxograma para teste dos sensores.



Fonte: (AUTORES, 2019).

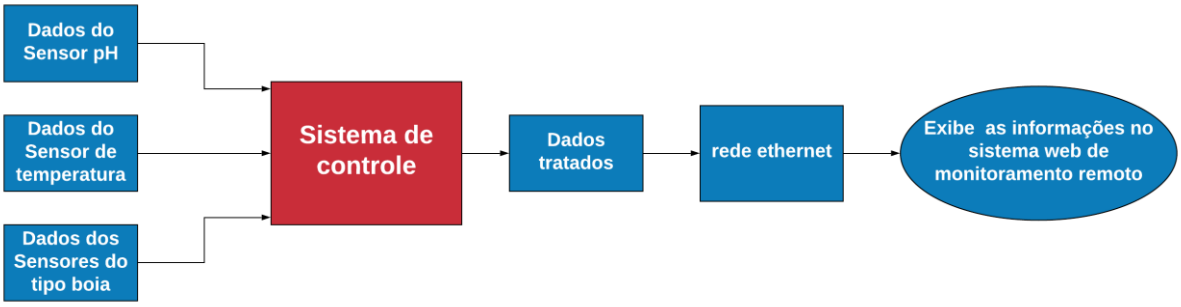
Para alcançar o principal objetivo do projeto, monitorar remotamente o cultivo hidropônico, foi gerado um diagrama de blocos, como também o fluxograma que possui a lógica de programação necessária para compartilhar os dados do protótipo de medição com o sistema web. A figura 11, na página seguinte, apresenta o diagrama de blocos com a inclusão do *shield ethernet* que possibilitará o compartilhamento dos dados do cultivo com o sistema web, para utilizar tal função foi gerado o fluxograma da figura 12, na página seguinte, para o desenvolvimento da lógica.

Figura 11 - Diagrama de blocos com *shield ethernet*.



Fonte: (AUTORES, 2019).

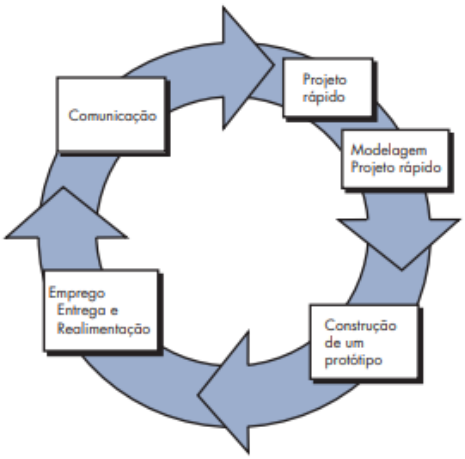
Figura 12- Fluxograma para compartilhamento dos dados na rede



Fonte: (AUTORES, 2019).

O modelo de processo de *software* utilizado neste projeto para o desenvolvimento do sistema web foi a prototipação, exemplificado na figura 13, abaixo, pois foi gerado um modelo inicial, ou seja, um protótipo que poderá ser aperfeiçoado. Um protótipo é uma versão inicial de um sistema de *software*, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções (SOMMERVILLE, 2011).

Figura 13 - Modelo de Prototipagem



Fonte: (PRESSMAN, 2011, p. 63).

A interação desse modelo vai ocorrendo conforme se ajusta o protótipo com a necessidade real do usuário, sendo assim a escolha desse modelo de processo ocorreu devido à natureza do projeto. Uma vez que, foi necessário explorar as necessidades do cultivo hidropônico várias vertentes poderiam surgir ao longo do projeto. Desse modo, conforme a evolução do estudo foi possível alterar o sistema web de acordo com as mudanças necessárias.

Portanto, as alterações foram feitas durante a prototipagem, uma vez que o projeto não contou com um especialista em agronomia dentro da equipe (para informar durante a fase de planejamento as reais necessidades do cultivo hidropônico). Sendo assim, em estudos futuros caso busquem implantar o projeto, o modelo de prototipagem poderá ser inserido junto a outro ciclo de vida, dando continuidade ao projeto. Embora a prototipação possa ser utilizada como um modelo de processo isolado é mais comumente utilizada como uma técnica passível de ser implementada a outro modelo de processo (PRESSMAN, 2011).

4. DESENVOLVIMENTO

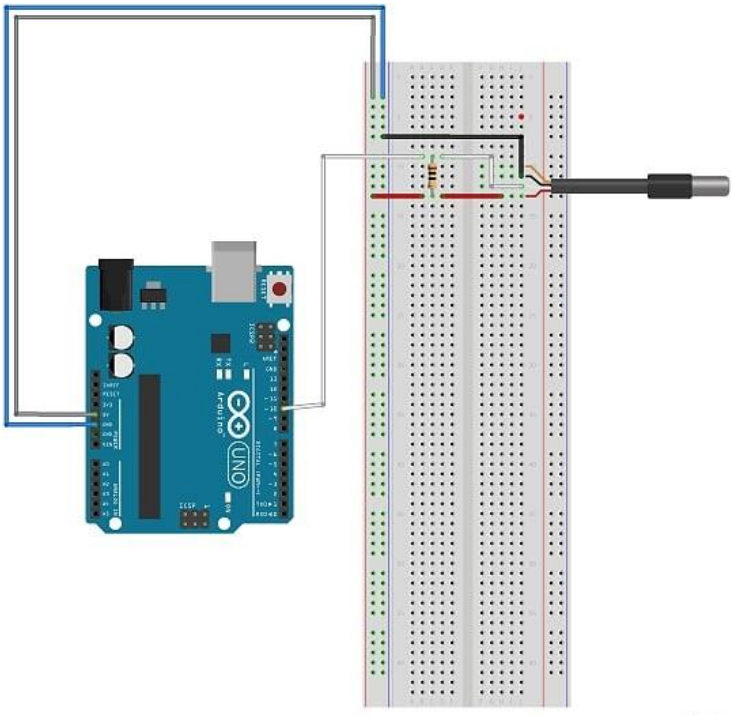
Durante a construção do protótipo que utiliza a plataforma *arduino*, para o monitoramento do cultivo hidropônico, foram feitas analises de quais equipamentos seriam necessários para execução do protótipo do *hardware* do projeto. Da mesma forma, foram analisadas as tecnologias necessárias para o desenvolvimento do sistema web responsável por exibir remotamente os dados do cultivo hidropônico, conforme foi citado no referencial teórico.

O protótipo trata os dados que são coletados pelos sensores possibilitando, assim, o envio desses dados para a rede, o qual é exibido remotamente através de um sistema web. Nos tópicos a seguir são descritos o desenvolvimento e a construção do *hardware*, bem como o desenvolvimento do sistema web.

4.1 Desenvolvimento do *hardware*

4.1.1. Esquema de ligação para teste do sensor de temperatura DS18B20

Figura 14- Esquema de ligação do sensor de temperatura DS18B20.



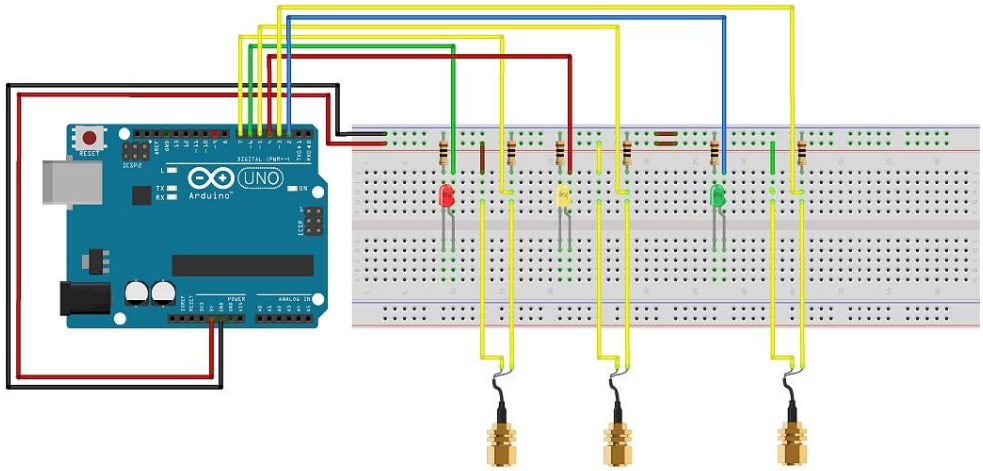
Fonte: (AUTORES, 2019).

Na figura 14, na página anterior, estão dispostas as ligações feitas para testar o funcionamento do sensor DS18B20, onde utilizou-se uma ligação serial nas entradas GND E 5V da *protoboard*. O dispositivo está ligado na porta digital número dez e foi utilizado um transistor de 100R. No anexo "A" está disposto o código do sensor de temperatura DSB18B20, que é responsável por tratar os dados recebidos a partir da lógica de programação encontrada no mesmo, exibindo assim a temperatura do reservatório onde encontra-se a solução.

4.1.2. Esquema de ligação dos sensores do tipo boia

Na figura 15, abaixo, estão dispostas as ligações feitas para testar o funcionamento dos sensores do tipo boia, onde utilizou-se uma ligação serial nas entradas GND E 5V na *protoboard*. Os dispositivos foram ligados nas portas digitais números três cinco e sete e foram inseridos ao equipamento LEDs para simbolizarem os estados do reservatório, sendo: verde (reservatório cheio), amarelo (liquido no meio o reservatório) e vermelho (liquido acabando). Os LEDs foram ligados respectivamente nas portas dois quatro e seis, e foram utilizados 5 transistores de 100R. No apêndice "A" está o código dos sensores do tipo boia de nível, que é responsável por tratar os dados recebidos pelas boias e exibir o status do reservatório com os LEDs mencionados anteriormente.

Figura 15- Esquema de ligação dos sensores do tipo boia.

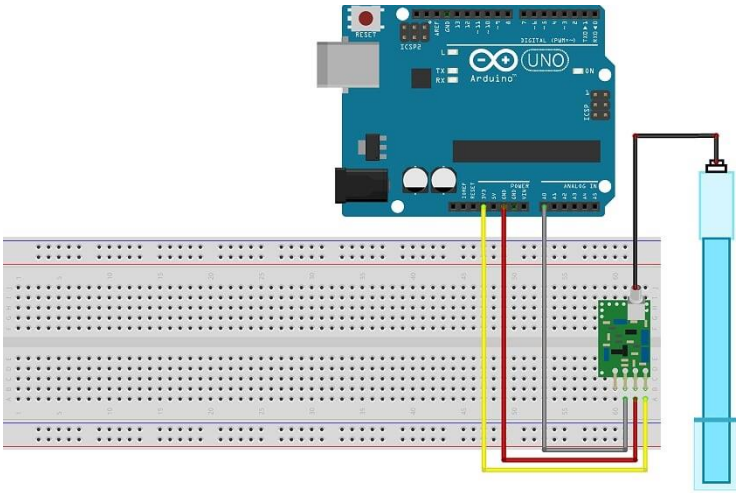


Fonte: (AUTORES, 2019).

4.1.3. Esquema de ligação do sensor de Ph com Módulo BNC

A figura 16, abaixo, mostra as ligações feitas para teste do sensor de PH, que é responsável por medir o nível de acidez da água. O dispositivo foi ligado utilizando a porta analógica A0 e as entradas GND E 3V. No anexo “B” está o código do sensor de PH, que é responsável por regular os dados recebidos a partir da lógica de programação e exibir o valor da acidez encontrado na solução aquosa do reservatório.

Figura 16 - Esquema de ligação do sensor de Ph com módulo BNC.

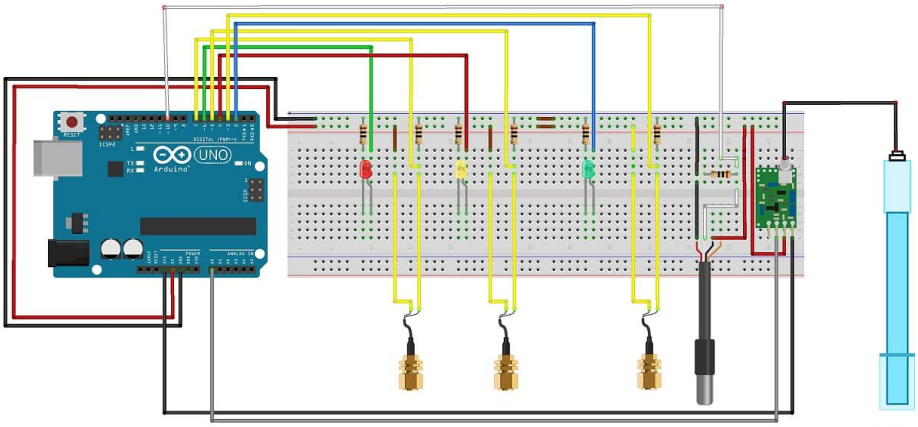


Fonte: (AUTORES, 2019).

4.1.4. Esquema de ligação para teste de funcionamento conjunto dos sensores

O esquema ilustrado na figura 17, na página seguinte, representa o teste da ligação em conjunto de todos os sensores mencionados no desenvolvimento. No apêndice “B” está o código da integração de todos os sensores, código esse que é a união da lógica de todos os sensores citados ao longo do desenvolvimento.

Figura 17 - Esquema de ligação com todos os sensores.



Fonte: (AUTORES, 2019).

4.1.5. Resultados alcançados a partir dos testes de funcionamento dos sensores

Nas figuras 18, 19, 20, 21 e 22, nas páginas seguintes, estão as imagens exibindo cada parte do desenvolvimento do *hardware* para o protótipo do projeto, que teve como intuito testar todos os componentes de medição, desenvolvendo, assim, parte do protótipo de monitoramento. Dispondo de todos os sensores que foram citados ao longo do referencial teórico e do desenvolvimento, como as boias, o sensor de PH e o sensor de temperatura.

No protótipo desenvolvido o nível de água do reservatório é exibido por LED'S como mostra a figura 21, para essa representação foi utilizado um padrão de cores, verde, amarelo e vermelho. Caso o reservatório estiver cheio todos os LED'S estarão ligados; quando o nível estiver na metade apenas o LED verde apagará e somente o amarelo e o vermelho permanecerão ligados; se o reservatório estiver na capacidade mínima, ou seja, quase acabando, apenas o LED vermelho permanecera aceso.

Os dados de temperatura e de PH da água são informados no monitor serial da IDE *arduino*, onde são atualizados a todo o momento para que os dados sempre estejam de acordo com a situação atual do líquido presente no reservatório, como é exibido na figura 22.

Figura 18- Protótipo.

Fonte: (AUTORES, 2019).

Figura 19 - Inserindo o sensor de PH no reservatório.

Fonte: (AUTORES, 2019).

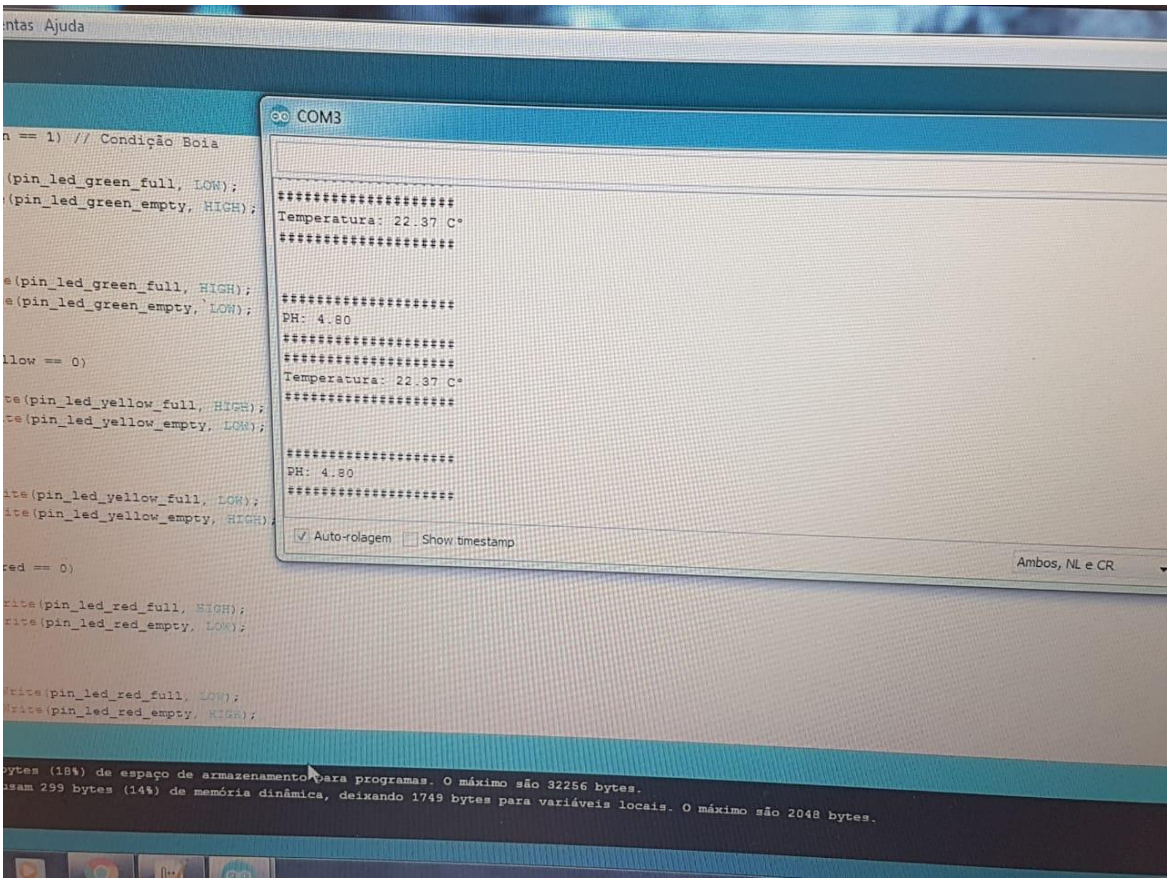
Figura 20 - Posicionando o sensor de temperatura no reservatório.

Fonte: (AUTORES, 2019).

Figura 21 - LED'S demonstrando que o reservatório está cheio.

Fonte: (AUTORES, 2019).

Figura 22 - Monitor serial da IDE *arduino* demonstrando o valor do PH e a temperatura da água.



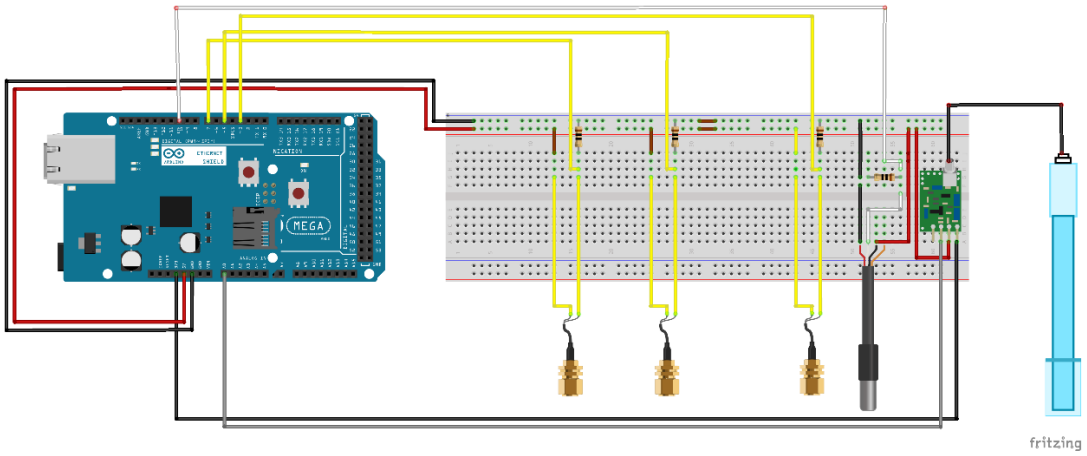
Fonte: (AUTORES, 2019).

4.1.6. Esquema de ligação do protótipo com a placa *Ethernet* integrada

O esquema ilustrado na figura 23, na página seguinte, representa a ligação em conjunto de todos os sensores mencionados no desenvolvimento do *hardware*, estes que integram a placa *arduino* uno juntamente com o *shield Ethernet*. No apêndice “D” encontra-se o link do GitHub com o projeto completo, onde também se encontra o código *arduino* desenvolvido.

O protótipo de monitoramento citado anteriormente nos subtópicos do item 4.1 foi atualizado para uma versão que conta com compartilhamento de dados pela rede *Ethernet*, deixando, assim, de mostrar apenas localmente as informações do cultivo hidropônico como foi apresentado nas figuras 18, 19, 20, 21 e 22, onde todos os dados do cultivo eram apresentados localmente.

Figura 23 - Esquema de ligação com *shield Ethernet* integrado.



Fonte: (AUTORES, 2019).

4.2 Desenvolvimento do *software*

4.2.1. Requisitos não Funcionais do sistema

Tabela 11 - Requisitos não funcionais.

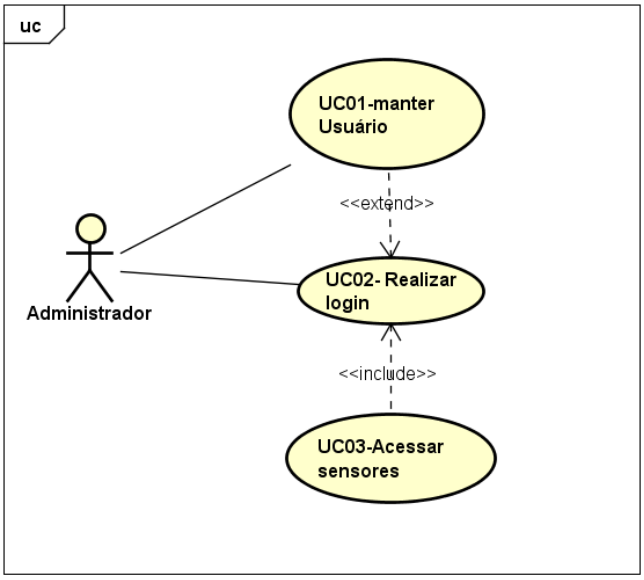
Requisito Não Funcional 01 – Confiabilidade
Este requisito também visa garantir a diminuição da taxa de falhas do sistema, garantindo a disponibilidade do mesmo para o usuário.
Requisito Não Funcional 02 – Usabilidade
Facilidade no entendimento do funcionamento do sistema para com o usuário, facilidade para achar ferramentas do sistema, quanto mais autoexplicativo o sistema for melhor, a funcionalidade visa o tempo gasto pelo usuário para achar uma certa ferramenta do sistema ou o tempo necessário para entende-lo. Como por exemplo: Um dado que o usuário necessite de um determinado sensor, e ele não o encontre em um tempo aceitável de uso do sistema, sendo assim com base no que visa o requisito de usabilidade esse sistema não seria auto explicativo sendo classificado como um sistema de difícil compreensão para o usuário, desse modo não atendendo o requisito de usabilidade.
Requisito Não Funcional 03 – Eficiência
O sistema deverá possuir um bom tempo de resposta para os dados coletados dos sensores, além de mostrar esses dados para o usuário de maneira precisa e eficiente.
Requisito Não Funcional 04 – Manutenibilidade
O sistema deverá ser fácil de manter e atualizar assim reduzindo o tempo das manutenções e a sua dificuldade.

Fonte: (AUTORES, 2019).

Na tabela 11, na página anterior, encontram-se os requisitos não funcionais utilizados para o desenvolvimento do sistema web para monitoramento remoto do cultivo hidropônico. Sendo esses requisitos confiabilidade, usabilidade, eficiência e manutenibilidade.

4.2.2. Diagrama de caso de uso do sistema

Figura 24 - Caso de uso.



Fonte: (AUTORES, 2019).

Tabela 12 - Descrição do caso de uso.

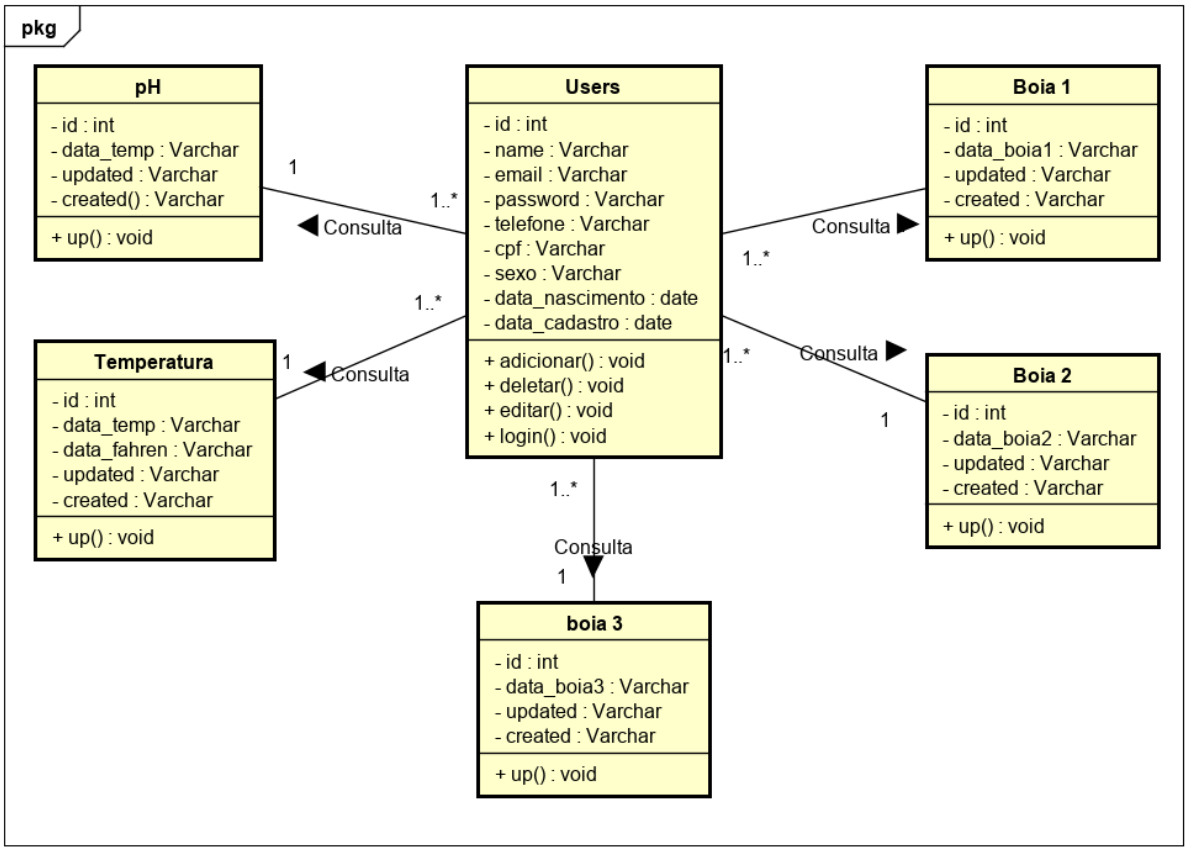
Caso de uso UC01 - Manter usuário
Se o administrador estiver logado no sistema ele terá a permissão de realizar o cadastro de novos usuários no sistema. O sistema disponibilizará um formulário contendo informações relacionadas ao novo administrador. Os dados do novo administrador são: nome, e-mail, senha, data de nascimento, cpf, sexo, data de cadastro, telefone. O novo usuário será um administrador.
Caso de uso UC02 – Realizar *login*
O administrador poderá efetuar o *login* no sistema
Caso de uso UC03 – Acessar sensores
Se o administrador estiver logado no sistema ele poderá visualizar e consultar os dados de medição de qualquer um dos sensores, seja o sensor de ph, temperatura ou de nível de água.

Fonte: (AUTORES, 2019).

Na figura 24, na página anterior, é exibido o caso de uso utilizado no desenvolvimento do sistema web, contendo o ator administrador e as ações realizadas pelo mesmo. Enquanto que na tabela 12, na página anterior, está detalhada cada ação presente no caso de uso mostrada na figura 24.

4.2.3. Diagrama de classe do sistema

Figura 25 - Diagrama de classes.



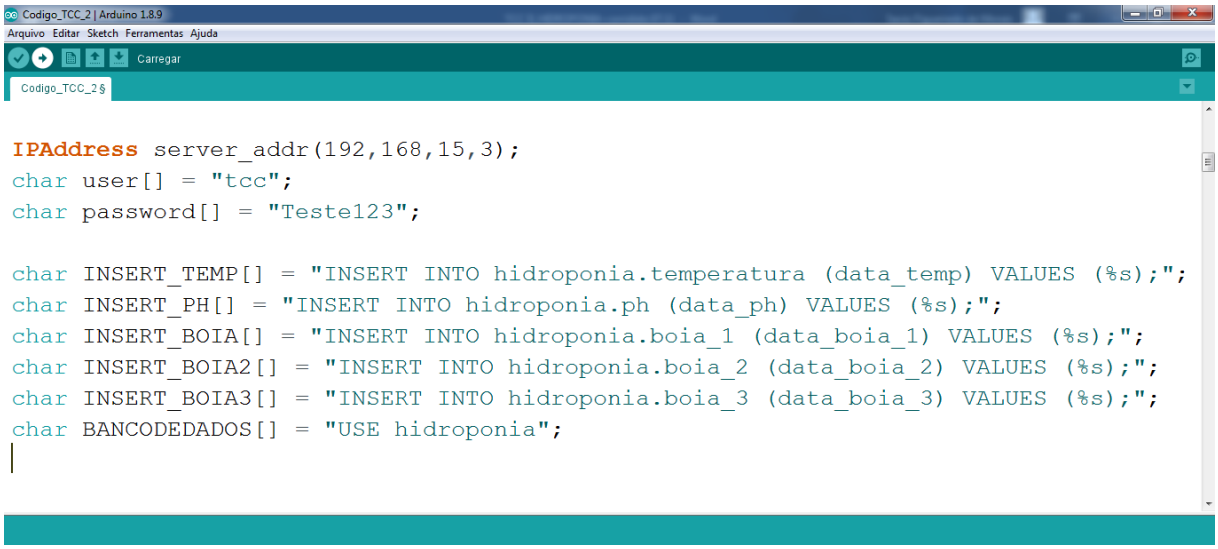
Fonte: (AUTORES, 2019).

O diagrama de classes ilustrado na figura 25, acima, foi utilizado para descrever os objetos do sistema web e o relacionamento entre eles, assim como demonstrar a arquitetura do sistema desenvolvido.

4.2.4. Códigos gerados a partir do processo de desenvolvimento do sistema web

Na Figura 26 é apresentada a tela da IDE do Arduino com um trecho do código escrito para o sistema web de monitoramento remoto. Para visualização do código completo é necessário consultar o link do GitHub que está disponível no apêndice “D”. Neste trecho de código é demonstrado o envio dos dados coletados pelos sensores para o banco de dados.

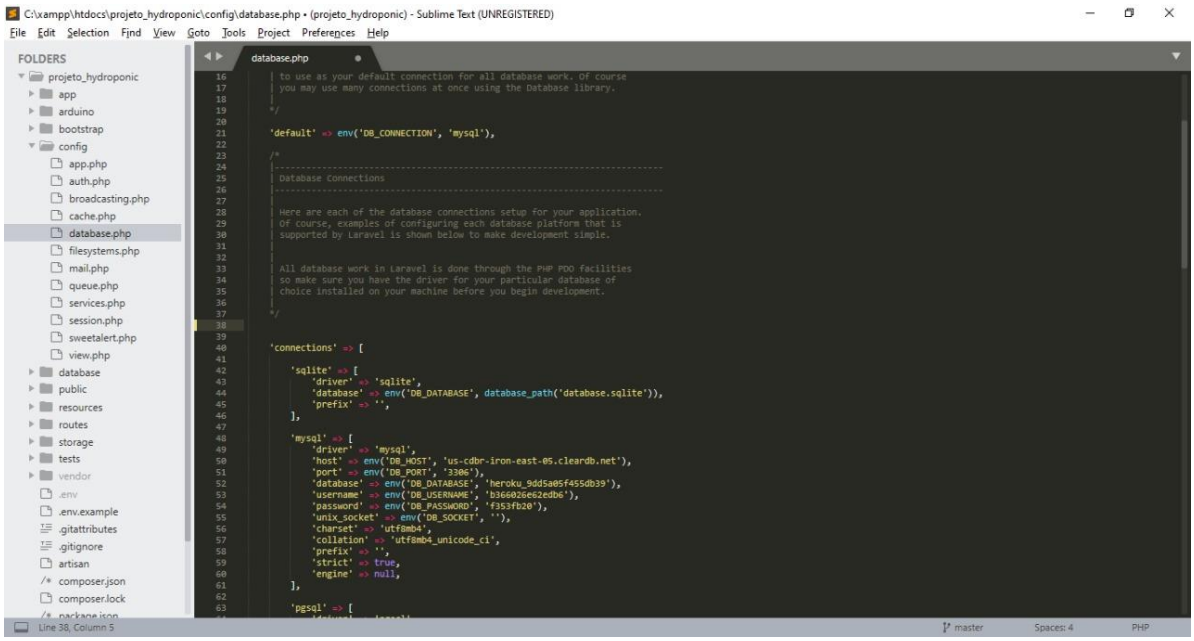
Figura 26 - IDE arduino com trecho do código gerado.



Fonte: (AUTORES, 2019).

Também Foram criados o SQL das tabelas do usuário e dos sensores para manipular o banco de dados, trechos destes estão dispostos no apêndice “C”, assim como as representações lógicas dos mesmos, para acessar o código na integra é necessário acessar o link do GitHub no apêndice “D”. Na Figura 27, na próxima página, é demonstrado a tela da IDE de desenvolvimento com o trecho do código usado para realizar a conexão com banco de dados do Heroku, que é uma plataforma em nuvem capaz de suportar várias linguagens de programação. Para visualização do código completo é necessário consultar o link disponibilizado do GitHub, no apêndice “D”.

Figura 27 - Conexão com o heroku

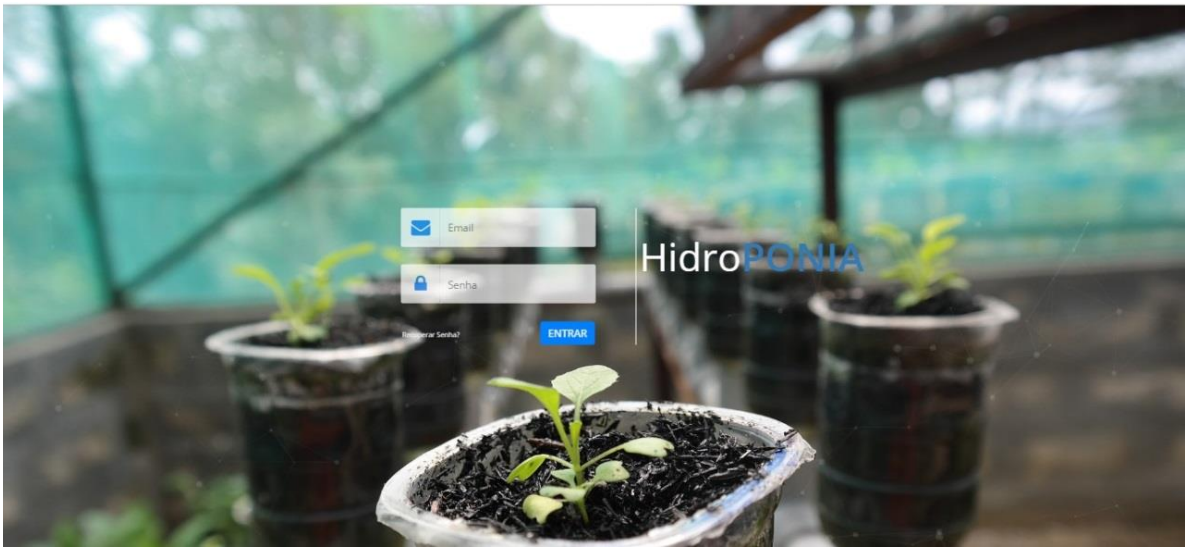


Fonte: (AUTORES, 2019).

4.2.5. Sistema web de monitoramento remoto

Na Figura 28, abaixo, encontra-se a tela de *login* do sistema web, onde são requisitados do usuário os dados de e-mail e senha, esses dados devem estar devidamente cadastrados no sistema.

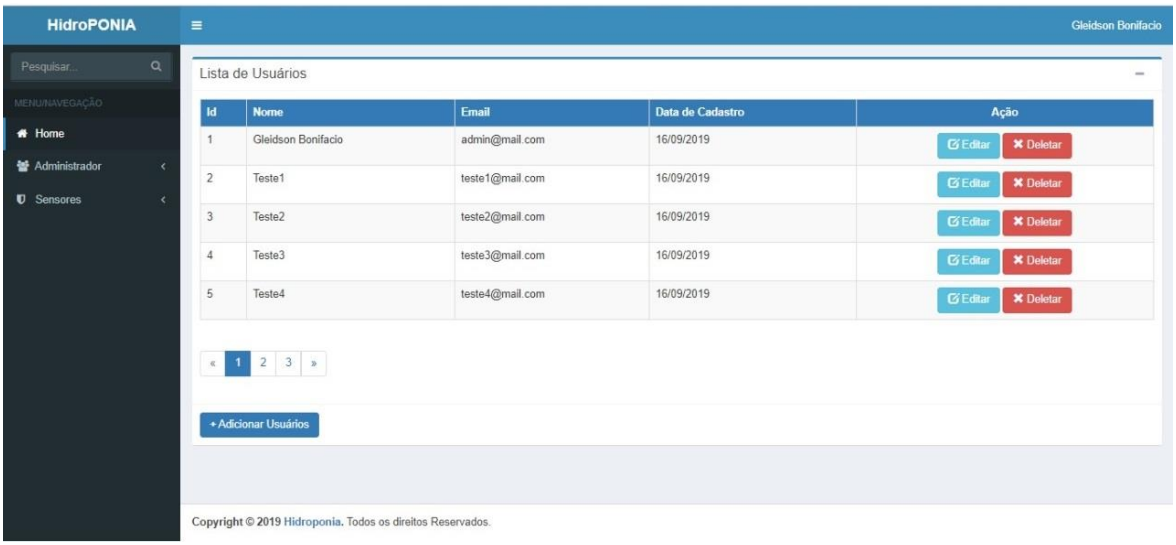
Figura 28 - Tela de *login*.



Fonte: (AUTORES, 2019).

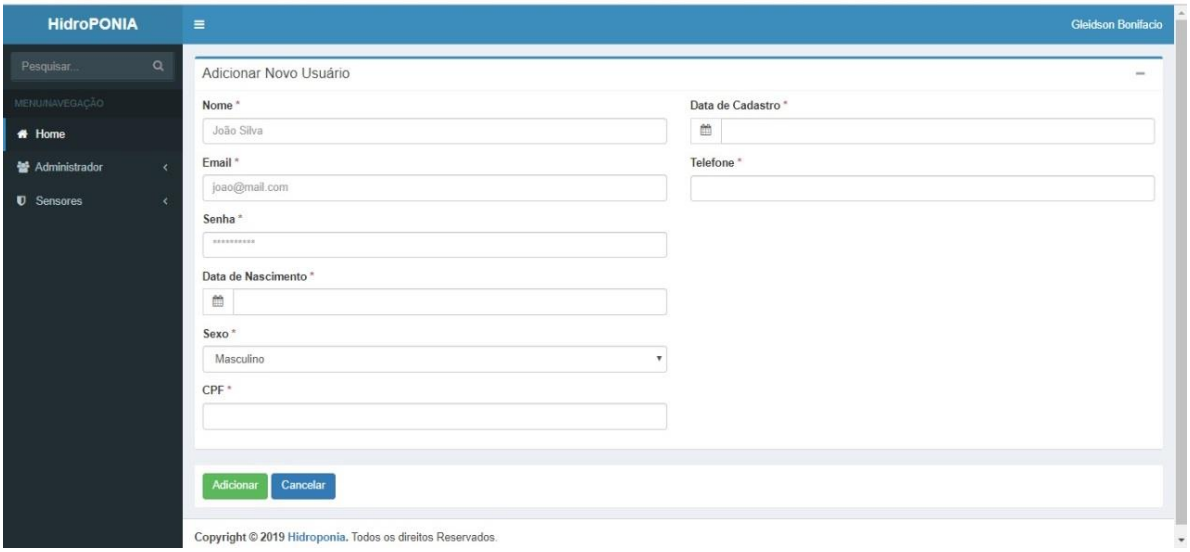
Um novo usuário poderá ser cadastrado somente pelo administrador, para isso é necessário que este vá até a lista de usuários (ilustrado na Figura 29) e escolha a opção adicionar usuário. Em seguida o administrador será redirecionado para a tela de cadastro, onde será necessário preencher todos os campos requisitados como está ilustrado na Figura 30, na página seguinte. Por se tratar de um protótipo ainda não existem níveis de usuário no sistema, portanto, todos os usuários que forem adicionados se tornarão administradores.

Figura 29 – Lista de usuários.



Fonte: (AUTORES, 2019).

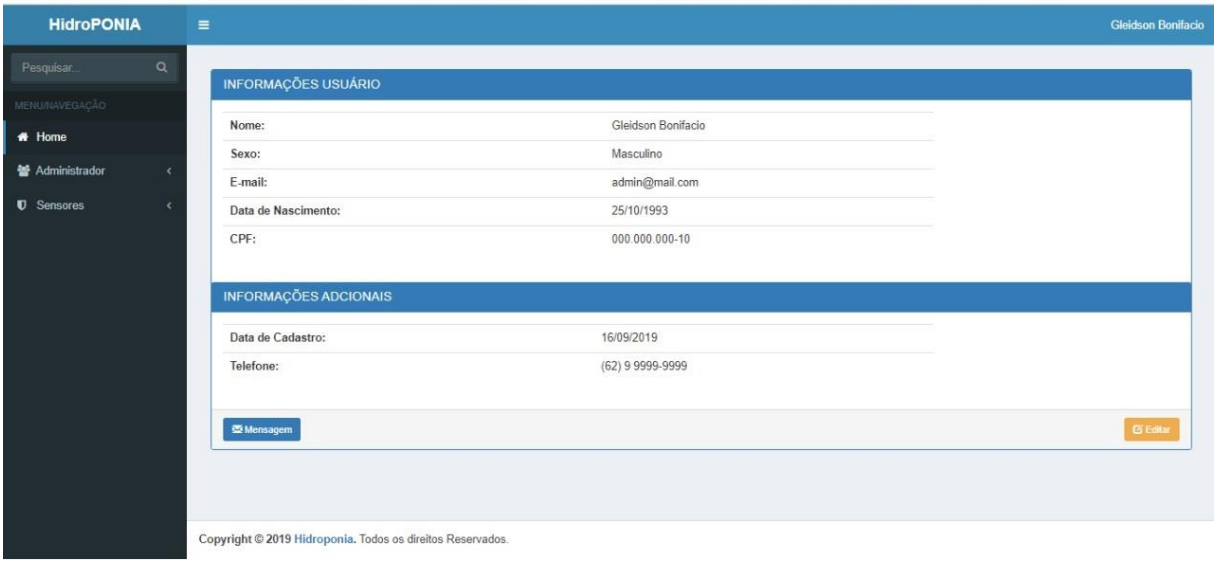
Figura 30 – Tela de cadastro de usuários.



Fonte: (AUTORES, 2019).

Após ser cadastrado, os dados do novo administrador ficarão salvos no sistema, como é mostrado na Figura 31, a baixo.

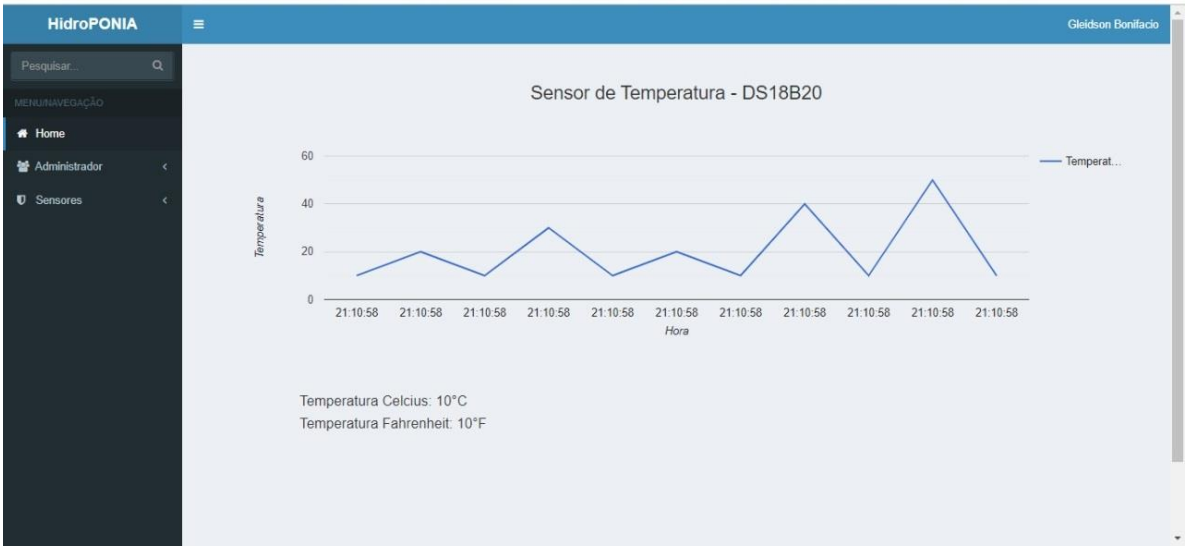
Figura 31 - Tela de dados do usuário cadastrado.



Fonte: (AUTORES, 2019).

Nas figuras 32 e 33, abaixo e na página seguinte, estão respectivamente as telas de monitoramento da temperatura e do PH da solução nutritiva. Onde são exibidos os valores de seus respectivos dados, assim como o horário e valores anteriores. Estes dados ficam salvos e suas variações são exemplificadas graficamente, como são mostradas nas figuras 32 e 33.

Figura 32 - Tela de monitoramento da temperatura.



Fonte: (AUTORES, 2019).

Figura 33 - Tela de monitoramento de PH da água.

Fonte: (AUTORES, 2019).

Na Figura 34, abaixo, está a tela de monitoramento de nível de água do reservatório, através desta é possível obter um controle preciso de quando será necessário repor a solução nutritiva. Evitando assim a ineficiência no cultivo hidropônico e a constante necessidade de monitoramento presencial.

Figura 34 - Tela de monitoramento de nível de água.

Fonte: (AUTORES, 2019).

5. CONSIDERACÕES FINAIS

O desenvolvimento de um protótipo do sistema de monitoramento remoto do cultivo hidropônico contribui para a modernização e melhoramento da agricultura no país. Este estudo colabora com a utilização de tecnologia no cultivo de alimentos, buscando incentivar a implementação do mesmo para ganhos de produtividade e qualidade do produto.

O protótipo de *hardware* e *software* desenvolvidos facilita a rotina do hidrocultor, monitorando o cultivo e possibilitando a visualização dos dados do mesmo de forma intuitiva. Logo o estudo desenvolvido auxilia no melhor acompanhamento da solução nutritiva e das condições de crescimento das plantas na hidroponia. Ao saber-se em tempo real o status do cultivo o hidrocultor poderá agir de maneira proativa em busca da otimização dos nutrientes que a planta necessita para o seu desenvolvimento, e consequentemente obterá melhores resultados em sua produção.

Espera-se que em estudos futuros a tecnologia desenvolvida possa ser implementada e escalada, e portanto, utilizada para gerar um melhor rendimento no cultivo hidropônico, beneficiando, assim, não somente o hidrocultor com uma melhoria na produtividade, mas também o consumidor com alimentos de qualidade.

6. REFERENCIAS BIBLIOGRAFICAS

ADRIEL, W. **Introdução ao *Framework* PHP Laravel**. 2015. Disponível em: <http://www.devmedia.com.br/introducao-ao-framework-php-laravel/33173>. Acesso em: 28 set. 2019.

ANDREI, L. **O que é Apache? Uma visão aprofundada do servidor Apache**. 2019. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-apache>. Acesso em: 28 set. 2019.

APRENDENDOFISICA. **Sensor ds18b20**. 2019. Disponível em: <http://aprendendofisica.pro.br/pmwiki.php/Main/DS18B20>. Acesso em: 7 mar. 2019.

ARDUINO-SANTA-EFIGENIA. **Sensor de pH Arduino com Módulo de Leitura**. 2019a. Disponível em: <http://www.arduinosantaefigenia.com.br/produto/189/sensores-e-modulos/modulos/sensor-de-ph-arduino-com-modulo-de-leitura>. Acesso em: 8 mar. 2019.

ARDUINO-SANTA-EFIGENIA. ***How to use a PH probe and sensor***. 2019b. Disponível em: <http://www.arduinosantaefigenia.com.br/wp/wp-content/uploads/2018/10/ph-sensor-ph-4502c.pdf>. Acesso em: 22 abr. 2019.

ARDUINO, T. et al. **Arduino Tutorial for *Ethernet Shield***. 2010.

ARROYO, A.; SANTOS, F. **Programação para web utilizando PHP**. Campinas SP: Centro de Computação da Unicamp, 2002.

BANZI, M.; SHILOH, M. **Primeiros Passos com o Arduino**. São Paulo SP: Novatec, 2015.

BEZERRA NETO, E.; BARRETO, L. . As técnicas de hidroponia. **Anais da Academia Pernambucana de Ciência Agronômica**, v. 8, n. 9, p. 107–137, 2012.

CHAVIER, L. F. **Programação para Arduino - Primeiros Passos**. 2019. Disponível em: <https://www.circuitar.com.br/tutoriais/programacao-para-arduino-primeiros-passos/>. Acesso em: 1 maio. 2019.

DANIEL. **Primeiros passos em PaaS com Heroku**. 2013. Disponível em: <https://www.devmedia.com.br/primeiros-passos-em-paas-com-heroku/29465>. Acesso em: 28 set. 2019.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA (EMBRAPA). **Automação e agricultura de precisão**. [s.d.]. Disponível em: <https://www.embrapa.br/tema-mecanizacao-e-agricultura-de-precisao/nota-tecnica>. Acesso em: 20 jun. 2019.

FURLANI, P. R. et al. **Cultivo Hidropônico de Plantas Parte 2 - Solução nutritiva**. 2009. Disponível em: <http://www.infobibos.com/Artigos/2009_2/hidroponiap2/index.htm>. Acesso em: 20 jun. 2019.

GROHO. **Métodos de oxigenação da solução nutritiva**. [s.d.]. Disponível em: <https://www.groho.pt/post/metodos-de-oxigenacao-da-solucao-nutritiva>. Acesso em: 11 abr. 2019.

GUEDES, G. T. A. **Uml - Uma Abordagem Prática**. São Paulo SP: Novatec, 2011.

HEROKU. **Heroku**. 2019. Disponível em: <https://devcenter.heroku.com/>. Acesso em: 28 set. 2019.

LONAX-ADMIN. **Entenda a importância da qualidade da água para a hidroponia**. 2019a. Disponível em: <https://www.lonax.com.br/entenda-a-importancia-da-qualidade-da-agua-para-a-hidroponia/ >. Acesso em: 11 mar. 2019.

LONAX-ADMIN. **Conheça 3 vantagens e 3 desvantagens da hidroponia**. 2018b. Disponível em: <https://www.lonax.com.br/conheca-3-vantagens-e-3-desvantagens-da-hidroponia/>. Acesso em: 2 out. 2019.

MELONIO, N. **Hidroponia: conheça os prós e contra nesse tipo de cultivo**. 2012. Disponível em: <https://www.oeco.org.br/noticias/25959-hidroponia-conheca-os-pros-e-contra-nesse-tipo-de-cultivo/>. Acesso em: 27 abr. 2019.

MOTA, A. **O que é Arduino e como funciona?.** 2017. Disponível em: <https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/>. Acesso em: 14 mar. 2019.

MULTIPEÇAS. **MINI BOMBA DE AGUA PARA IRRIGACAO 12V RS385**. 2019. Disponível em: <http://www.multipecas.curitiba.br/loja/produto/mini-bomba-de-agua-para-irrigacao-12v-rs385>. Acesso em: 12 abr. 2019.

OLARIA.UCPEL. **SENSOR DE TEMPERATURA 1-WIRE DS18B20**. 2019. Disponível em: <http://olaria.ucpel.tche.br/autubi/lib/exe/fetch.php?media=ds18b20.pdf>. Acesso em: 30 abr. 2019.

PRACIANO, E. **Os benefícios e as vantagens do PHP**. 2014. Disponível em: <https://elias.praciano.com/2014/02/15-beneficios-e-vantagens-do-php/>. Acesso em: 28 set. 2019.

PRESSMAN, R. **Engenharia de Software. Uma Abordagen Profissional**. 7. ed. Porto alegre RS: Ltda, AMGH Editora, 2011.

ROBOCORE. **BlackBoard UNO R3**. 2019a. Disponível em: <https://www.robocore.net/loja/arduino/arduino-blackboard>. Acesso em: 5 mar. 2019.

ROBOCORE. **BlackBoard UNO R3 - Inf. Técnicas**. 2019b. Disponível em: <https://www.robocore.net/loja/arduino/arduino-blackboard#inf_tecnicas>. Acesso em: 27 fev. 2019.

ROBOCORE. **Sensor de temperatura DS18B20 - À Prova de Água**. 2019c. Disponível em: <https://www.robocore.net/loja/sensores/sensor-de-temperatura-ds18b20-a-prova-de-agua#inf_tecnicas>. Acesso em: 25 abr. 2019.

RURALNEWS, R. **Hidroponia**. 2017. Disponível em: <http://www.ruralnews.com.br/visualiza.php?id=71>. Acesso em: 14 mar. 2019.

SANTOS, C. B. DA C. **Cultivo Hidropônico: uma prática eficiente e de alta rentabilidade**. 2015. Disponível em: <http://www.esalq.usp.br/cprural/boapratica/mostra/97/cultivo-hidroponico-uma-pratica-eficiente-e-de-alta-rentabilidade.html>. Acesso em: 20 jun. 2019.

SANTOS, O. S. et al. **Cultivo sem Solo - Hidroponia**. 2. ed. Santa Maria: Rurais, Universidade Federal de Santa Maria – Centro de Ciências, 2002.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice, 2011.

SOUZA, F. **Placa Arduino da Robocore - BlackBoard**. 2014a. Disponível em: <https://www.embarcados.com.br/placa-arduino-da-robocore-blackboard/>. Acesso em: 10 abr. 2019.

SOUZA, F. **Arduino MEGA 2560**. 2014b. Disponível em: <https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 25 set. 2019.

SPERAFICO, D. **Os benefícios da moderna tecnologia para o agronegócio**. 2018. Disponível em: <https://agencia.fpagropecuaria.org.br/2018/06/15/os-beneficios-da-moderna-tecnologia-para-o-agronegocio/>. Acesso em: 2 out. 2019.

TECNOVESTE. **O que é Open Source?**. 2016. Disponível em: <http://blogs.correiobraziliense.com.br/tecnoveste/o-que-e-open-source/>. Acesso em: 14 mar. 2019.

THOMSEN, A. **O que é Arduino?**. 2014. Disponível em: <https://www.filipeflop.com/blog/o-que-e-arduino/>. Acesso em: 1 maio. 2019.

UZINAINFO. **Mini Bomba de Água (d'água) para Arduino RS-385 - Alto Fluxo**. 2019a. Disponível em: <https://www.usinainfo.com.br/mini-bombas-de-agua-e-ar/mini-bomba-de-agua-dagua-para-arduino-rs-385-alto-fluxo-2814.html>. Acesso em: 11 abr. 2019.

UZINAINFO. **Sensor de Temperatura DS18B20 à Prova DÁgua**. 2019b. Disponível em: <https://www.usinainfo.com.br/sensor-de-temperatura-arduino/sensor-de-temperatura-ds18b20-a-prova-d-agua-2645.html>. Acesso em: 10 abr. 2019.

VIDA-DE-SILICIO. **Sensor de Nível de Água tipo Boia vertical**. 2019. Disponível em: <https://www.vidadesilicio.com.br/sensor-de-nivel-de-agua-tipo-boia>. Acesso em: 30 abr. 2019.

WELLING, L.; THOMSON, L. **PHP e MySQL - Desenvolvimento Web**. São Paulo: Elsevier, 2005.

WENDELL. **Laravel Tutorial**. 2015. Disponível em: <https://www.devmedia.com.br/laravel-tutorial/33173>. Acesso em: 28 set. 2019.

ANEXOS

ANEXO A – Código do Portal Vida de Silício: DS18B20 - Sensor de temperatura inteligente

```
#include <DallasTemperature.h>
#include <OneWire.h>
OneWire pino(10);
DallasTemperature barramento(&pino);
DeviceAddress sensor;

void setup(void)
{
  Serial.begin(9600);
  barramento.begin();
  barramento.getAddress(sensor, 0);
}
void loop()
{
  barramento.requestTemperatures();
  float temperatura = barramento.getTempC(sensor);
    Serial.print("Temperatura: ");
  Serial.print(temperatura);
    Serial.print(" Cº");
    Serial.print("\n");
  delay(500);
}
```

ANEXO B – Código de Gidahatari: *Cómo medir PH desde tu laptop en tiempo real con arduino*? (Como medir PH em seu laptop em tempo real com *arduino*)

```
const byte pHpin = A0;
float Po;
void setup()
{
Serial.begin(9600);
}
void loop()
{
Po = (1023 - analogRead(pHpin)) / 73.07;
Serial.println(Po, 2);
delay(1000);
}
```

APÊNDICES

APÊNDICE A - Código *arduino* dos Sensores do tipo Boia

```
DeviceAddress sensor;
// Pino ligado ao sensor de nivel de liquido
// Led verde
int pinosensor_1 = 3;
int pino_led_verde_cheio = 2;
int pino_led_verde_vazio = 0;
// Led amarelo
int pinosensor_2 = 5;
int pino_led_amarelo_cheio = 4;
int pino_led_amarelo_vazio = 13;
// Led vermelho
int pinosensor_3 = 7;
int pino_led_vermelho_cheio = 6;
int pino_led_vermelho_vazio = 0;

void setup()
{
  Serial.begin(9600);
  barramento.begin();
  barramento.getAddress(sensor, 0);
  //verde
  pinMode(pinosensor_1, INPUT);
  pinMode(pino_led_verde_cheio, OUTPUT);
  pinMode(pino_led_verde_vazio, OUTPUT);
  //amarelo
  pinMode(pinosensor_2, INPUT);
  pinMode(pino_led_amarelo_cheio, OUTPUT);
  pinMode(pino_led_amarelo_vazio, OUTPUT);
  //vermelho
```

```
  pinMode(pinosensor_3, INPUT);
  pinMode(pino_led_vermelho_cheio, OUTPUT);
  pinMode(pino_led_vermelho_vazio, OUTPUT);
}

void loop()
{
  barramento.requestTemperatures();
  float temperatura = barramento.getTempC(sensor);
    Serial.print("Temperatura: ");
  Serial.print(temperatura);
    Serial.print(" Cº");
    Serial.print("\n");

  int estado_verde = digitalRead(pinosensor_1);
  int estado_amarelo = digitalRead(pinosensor_2);
  int estado_vermelho = digitalRead(pinosensor_3);

  if(estado_verde == 1)
  {
    digitalWrite(pino_led_verde_cheio, LOW);
    digitalWrite(pino_led_verde_vazio, HIGH);
  }
  else
  {
    digitalWrite(pino_led_verde_cheio, HIGH);
    digitalWrite(pino_led_verde_vazio, LOW);
  }

  if(estado_amarelo == 1)
  {
    digitalWrite(pino_led_amarelo_cheio, LOW);
    digitalWrite(pino_led_amarelo_vazio, HIGH);
```

```
  }
  else
  {
    digitalWrite(pino_led_amarelo_cheio, HIGH);
    digitalWrite(pino_led_amarelo_vazio, LOW);
  }

  if(estado_vermelho == 1)
  {
    digitalWrite(pino_led_vermelho_cheio, LOW);
    digitalWrite(pino_led_vermelho_vazio, HIGH);
  }
  else
  {
    digitalWrite(pino_led_vermelho_cheio, HIGH);
    digitalWrite(pino_led_vermelho_vazio, LOW);
  }
  delay(100);
}
```

APÊNDICE B - Código *arduino* com todos os Sensores Integrados

```
#include <DallasTemperature.h>
#include <OneWire.h>

OneWire pino(10);
DallasTemperature barramento(&pino);
DeviceAddress sensor;
const byte pHpin = A0;
float Po;
// Led Green
int zp4510_1 = 7; // Boia 1
int pin_led_green_full = 6;
int pin_led_green_empty = 0;
// Led Yellow
int zp4510_2 = 5; // Boia 2
int pin_led_yellow_full = 4;
int pin_led_yellow_empty = 0;
// Led Red
int zp4510_3 = 3; // Boia 3
int pin_led_red_full = 2;
int pin_led_red_empty = 0;
void setup()
{
  Serial.begin(9600);
  // Green
  pinMode(zp4510_1, INPUT);
  pinMode(pin_led_green_full, OUTPUT);
  pinMode(pin_led_green_empty, OUTPUT);
  // Yellow
  pinMode(zp4510_2, INPUT);
  pinMode(pin_led_yellow_full, OUTPUT);
  pinMode(pin_led_yellow_empty, OUTPUT);
```

```
  // Red
  pinMode(zp4510_3, INPUT);
  pinMode(pin_led_red_full, OUTPUT);
  pinMode(pin_led_red_empty, OUTPUT);

  barramento.getAddress(sensor, 0);
 }
 void loop()
 {
  barramento.requestTemperatures();
  float temperatura = barramento.getTempC(sensor);
  Serial.print("####################");
  Serial.print("\n");
  Serial.print("Temperatura: ");
  Serial.print(temperatura);
  Serial.print(" Cº");
  Serial.print("\n");
  Serial.print("####################");
  Serial.print("\n");
  Serial.print("\n\n");
  Po = (1023 - analogRead(pHpin)) / 73.07; // Leia e inverta o valor da entrada analógica do
sensor de pH e, em seguida, dimensione 0-14.
  Serial.print("####################");
  Serial.print("\n");
  Serial.print("PH: ");
  Serial.println(Po, 2);
  Serial.print("####################");
  Serial.print("\n");
  int state_green = digitalRead(zp4510_1);
  int state_yellow = digitalRead(zp4510_2);
  int state_red = digitalRead(zp4510_3);

  if (state_green == 1) // Condição Boia
```

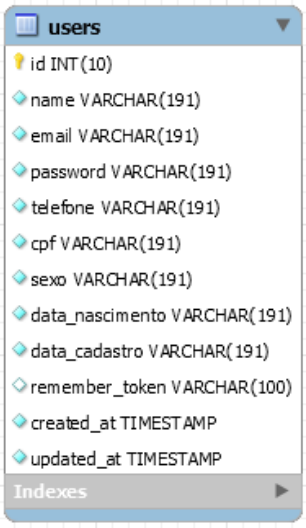
```
 {
  digitalWrite(pin_led_green_full, LOW);
  digitalWrite(pin_led_green_empty, HIGH);
 }
 else
 {
  digitalWrite(pin_led_green_full, HIGH);
  digitalWrite(pin_led_green_empty, LOW);
 }

 if (state_yellow == 1)
 {
  digitalWrite(pin_led_yellow_full, HIGH);
  digitalWrite(pin_led_yellow_empty, LOW);
 }
 else
 {
  digitalWrite(pin_led_yellow_full, LOW);
  digitalWrite(pin_led_yellow_empty, HIGH);
 }

 if (state_red == 1)
 {
  digitalWrite(pin_led_red_full, HIGH);
  digitalWrite(pin_led_red_empty, LOW);
 }
 else
 {
  digitalWrite(pin_led_red_full, LOW);
  digitalWrite(pin_led_red_empty, HIGH);
 }
 delay(1000);
}
```

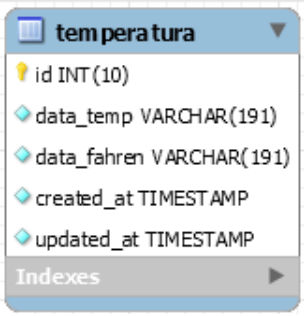
APÊNDICE C – Tabelas geradas

>>Tabela Users<<



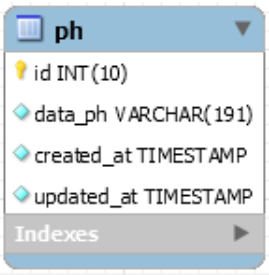

```
CREATE TABLE `users` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `telefone` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `cpf` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `sexo` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `data_nascimento` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `data_cadastro` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

>> Tabela temperatura<<



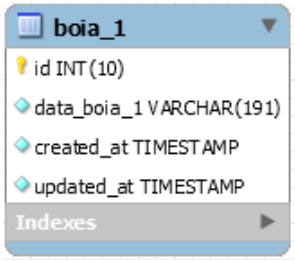

```
CREATE TABLE `temperatura` (
  `id` int(10) UNSIGNED NOT NULL,
  `data_temp` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `data_fahren` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

>> Tabela pH<<



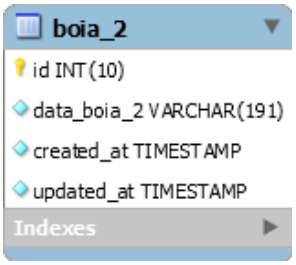

```
CREATE TABLE `ph` (
  `id` int(10) UNSIGNED NOT NULL,
  `data_ph` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

>> **Tabela boia 1**<<



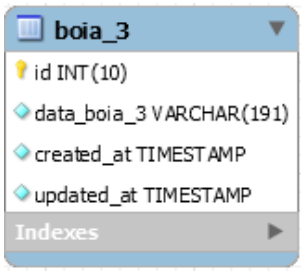

```
CREATE TABLE `boia_1` (
  `id` int(10) UNSIGNED NOT NULL,
  `data_boia_1` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

>> **Tabela boia 2**<<




```
CREATE TABLE `boia_2` (
  `id` int(10) UNSIGNED NOT NULL,
  `data_boia_2` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

>> **Tabela boia 3**<<




```
CREATE TABLE `boia_3` (
  `id` int(10) UNSIGNED NOT NULL,
  `data_boia_3` varchar(191) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT current_timestamp(),
  `updated_at` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

APÊNDICE D – Artefatos gerados durante a etapa de desenvolvimento.

https://github.com/gleidson-bs