

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE CIÊNCIAS - CAMPUS BAURU

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BEATRIZ AIKO HUKUCHIMA

**DESENVOLVIMENTO DE UM SISTEMA DE IOT APLICADO NA
ÁREA AGRÍCOLA**

BAURU

Dezembro/2020

BEATRIZ AIKO HUKUCHIMA

DESENVOLVIMENTO DE UM SISTEMA DE IOT APLICADO NA ÁREA AGRÍCOLA

Trabalho de Conclusão de Curso do Curso
de Ciência da Computação da Universidade
Estadual Paulista “Júlio de Mesquita Filho”,
Faculdade de Ciências, Campus Bauru.
Orientador: Prof. Dr. João Eduardo Machado
Perea Martins

BAURU
Dezembro/2020

Beatriz Aiko Hukuchima Desenvolvimento de um sistema de IoT aplicado na área agrícola/ Beatriz Aiko Hukuchima. – Bauru, Dezembro/2020- 50 p. : il. (algumas color.) ; 30 cm.

Trabalho de Conclusão de Curso – Universidade Estadual Paulista “Júlio de Mesquita Filho”

Faculdade de Ciências

Ciência da Computação, Dezembro/2020.

1. IoT 2. Agricultura 3. Sensores 4. Monitoramento 5. Internet

Beatriz Aiko Hukuchima

Desenvolvimento de um sistema de IoT aplicado na área agrícola

Trabalho de Conclusão de Curso do Curso de Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.

Banca Examinadora

**Prof. Dr. João Eduardo Machado Perea
Martins**

Orientador

Departamento de Ciência da Computação

Faculdade de Ciências

Universidade Estadual Paulista "Júlio de
Mesquita Filho"

**Profa Dra Simone das Graças Domingues
Prado**

Departamento de Ciência da Computação

Faculdade de Ciências

Universidade Estadual Paulista "Júlio de
Mesquita Filho"

Prof. Dr. Rogério Zanarde Barbosa
Faculdade de Agronomia e Engenharia Florestal

Bauru, 14 de Dezembro de 2020.

*Dedico este trabalho primeiramente à Deus. Dedico-o também a todos que me acompanharam
nesta jornada.*

Agradecimentos

Agradeço primeiramente à Deus, que me permitiu viver a graduação e realizar mais um sonho nessa vida. Agradeço ao meu avô, que tem cuidado tão bem de tudo e de todos, mesmo que em outro plano.

Agradeço meus professores por todo o conhecimento transmitido, pela dedicação, pelo amor ao ensino e pela minha transformação. Um obrigada especialmente à professora Andrea - que jamais mediu esforços para me ajudar e me permitiu viver dois lindos intercâmbios - e ao professor João Perea, pela orientação e auxílio na elaboração deste trabalho.

Obrigada aos meus familiares e aos meus amigos, que sempre estiveram presentes, dando força, conselhos e compartilhando tristezas e gargalhadas.

Agradeço ao Junior, meu namorado, por estar sempre tão disposto a me ouvir, aconselhar e apoiar. Obrigada pela parceria e por não me deixar desistir.

Por fim, gostaria de agradecer a todos os envolvidos durante todos estes anos de graduação. Obrigada por compartilharem comigo essa fase tão importante da minha vida.

Resumo

Para uma plantação render boas colheitas, sabe-se que são necessárias condições favoráveis de solo, ar e água. Atualmente, e de forma ascendente, o uso de sistemas de monitoramentos com aplicação da tecnologia *IoT* (*Internet of Things*) vem sendo utilizados pelos agricultores para maior controle e tomada de decisões em plantações, objetivando maior eficiência no campo. Com o apoio da *Internet*, sensores e *softwares*, grandes propriedades rurais são facilmente mapeadas e monitoradas com auxílio de tecnologias agrícolas de ponta, com soluções robustas e *softwares* custosos. Todavia, devido ao seu alto custo, não são acessíveis a muitos pequenos e médios agricultores. Tendo ciência de que a problemática deste assunto está na falta de recursos financeiros e não tecnológicos, este projeto objetiva a implementação de um sistema que utiliza a *IoT* aplicada na agricultura, visando a fácil implementação e baixo custo que, através da elaboração de um protótipo com utilização de sensores e desenvolvimento de uma página *web* que permite visualização de medições em tempo real, geração de gráficos e armazenamento dos dados, contribuem com o monitoramento de plantações de pequeno e médio porte. Os resultados obtidos com este projeto são satisfatórios, uma vez que atingem seus objetivos e mostram que mesmo a baixo custo, é possível realizar uma boa gestão de propriedade rural.

Palavras-chave: *IoT*, agronomia, plantação, *internet*, sensores, monitoramento agrícola.

Abstract

For a plantation to yield good harvests, it is known that favorable ground, air and water conditions are necessary. Nowadays and in an increasing way, the use of monitoring systems with the application of IoT (Internet of Things) technology has been used by farmers for greater control and decision making in plantations, aiming at greater efficiency in the field. With the support of the Internet, sensors and software, large rural properties are easily mapped and monitored with the help of cutting-edge agricultural technologies, with robust solutions and costly software. However, due to their high cost, they are not approachable to many small and medium rural producers. Being aware that the problem of this issue is the lack of financial and non-technological resources, this project aims to implement a system that uses IoT applied in agriculture, aiming at easy implementation and low cost that, through the elaboration of a prototype with use of sensors and development of a web page that allows visualization of measurements in real time, generation of graphics and data storage, contribute to the monitoring of small and medium-sized plantations. The results obtained with this project are satisfactory, once they reach their objectives and show that even at low cost, it is possible to carry out a good management of rural property.

Keywords: IoT, agronomy, plantation, internet, sensors, agricultural monitoring.

Lista de figuras

Figura 1 – Exemplo de JavaScript aplicado ao documento HTML.	19
Figura 2 – Placa Arduíno Uno.	24
Figura 3 – Módulo NodeMCU ESP8266.	25
Figura 4 – Sensor DHT11.	25
Figura 5 – Sensor LDR.	27
Figura 6 – Sensor de umidade de solo - Higrômetro.	28
Figura 7 – Demonstração da ferramenta para <i>upload</i> de arquivos na memória <i>flash</i> do microcontrolador.	32
Figura 8 – Diagrama representativo da organização dos arquivos do projeto.	33
Figura 9 – Protótipo do <i>hardware</i> com sensores e microcontrolador.	33
Figura 10 – Formulário Google - "Intermediária".	35
Figura 11 – Armazenamento de dados realizados na Planilha Google.	36
Figura 12 – Síntese da explicação referente à conexão da intermediária em conjunto com a planilha Google	36
Figura 13 – Exemplo de código HTML utilizado no projeto.	38
Figura 14 – Exemplo de código CSS utilizado no projeto.	38
Figura 15 – Ilustração do visor das variáveis de temperatura e umidade do ar em tempo x.	39
Figura 16 – Ilustração do visor das variáveis de temperatura e umidade do ar em tempo y.	40
Figura 17 – Diagrama - Integração final	43
Figura 18 – Exemplo de código desenvolvido na Arduíno IDE para subida dos arquivos no servidor <i>web</i>	44
Figura 19 – Ilustração da página <i>web</i> inicial contendo resumos referentes aos sensores utilizados no projeto.	44
Figura 20 – Ilustração da página <i>web</i> inicial contendo as medições realizadas pelos sensores de temperatura e umidade do ar em tempo real.	45
Figura 21 – Ilustração da página <i>web</i> inicial contendo as medições realizadas pelos sensores de umidade de solo e luminosidade em tempo real.	45
Figura 22 – Ilustração do gráfico gerado a partir do botão "Visualizar gráfico"do visor de umidade do ar.	46

Lista de abreviaturas e siglas

ADC	<i>Analog-to-Digital Converter</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
HTML	<i>Hyper Text Markup Language</i>
HTTP	<i>HyperText Transfer Protocol</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of Things</i>
IP	<i>Internet Protocol</i>
JSON	<i>JavaScript Object Notation</i>
LDR	<i>Light Dependent Resistor</i>
MCU	<i>Micro Controller Unit</i>
MIT	<i>Massachusetts Institute of Technology</i>
SPIFFS	<i>SPI Flash File System</i>
URL	<i>Uniform Resource Locator</i>
XML	<i>Extensible Markup Language</i>
XSLT	<i>Extensible Stylesheet Language Transformations</i>

Sumário

1	INTRODUÇÃO	12
1.1	Problemática	12
1.2	Justificativa	13
1.3	Objetivos	13
1.3.1	Objetivo Geral	13
1.3.2	Objetivos Específicos	14
1.4	Metodologia	14
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Microcontroladores	15
2.2	Banco de Dados	15
2.3	HTML	16
2.4	CSS	17
2.5	Javascript	18
2.5.1	Ajax	19
2.6	Ferramentas Google	20
2.6.1	Google Forms	21
2.6.2	Google Sheets	21
2.6.3	Google Charts	21
3	DISPOSITIVOS	23
3.1	Escolha da placa	23
3.1.1	Arduíno Uno	23
3.1.2	NodeMCU ESP8266	24
3.2	Sensores	25
3.2.1	Sensor de umidade e temperatura - DHT11	25
3.2.2	Sensor de Luminosidade - LDR	26
3.2.3	Sensor de Umidade de Solo - Higrômetro	27
4	DESENVOLVIMENTO	29
4.1	Estudo de <i>hardwares</i> e <i>webservers</i>	29
4.1.1	NodeMCU ESP8266 como <i>WebServer</i> x Hospedagem em <i>WebServers</i>	29
4.2	Testes unitários dos sensores	31
4.3	Utilização do Sistema de Arquivos do NodeMCU - <i>SPIFFS</i>	31
4.4	Elaboração do protótipo	33
4.5	Armazenamento de Dados	34

4.5.1	Concepção e execução	35
5	INTERFACE DESENVOLVIDA	37
5.1	Página web - HTML	37
5.2	Representação visual - Arquivo CSS	38
5.3	Funções em JavaScript	39
5.3.1	Atualização automática de valores através do <i>Ajax</i>	39
5.3.2	Geração de gráficos com <i>Google Charts</i>	40
5.4	Problemáticas do desenvolvimento da interface	41
5.4.1	Limitações de <i>hardware</i>	41
5.4.2	Geração de gráficos	41
6	INTEGRAÇÃO FINAL	43
7	CONCLUSÃO	47
	REFERÊNCIAS	48

1 Introdução

Esta pesquisa aborda a implementação de um sistema que utiliza a IoT (*Internet of Things* ou Internet das Coisas) aplicada na agricultura, objetivando sistemas de fácil implementação e baixo custo, que possam também ser usados por pequenos e médios produtores rurais.

A IoT é uma área da tecnologia que possibilita a conexão de objetos físicos e dispositivos com a *Internet* através da utilização de sensores inteligentes, possibilitando seu monitoramento através da rede (ABREU, 2019). Embora seja uma tecnologia emergente que se popularizou devido aos avanços tecnológicos recentes, a mesma surgiu no início dos anos 2000, quando Kevin Ashton estudava maneiras de melhoria de transações comerciais por meio da RFID conectada à *internet* em um laboratório do Massachusetts Institute of Technology (MIT) (RESEARCH, 2013).

Com a IoT, proporcionou-se relacionamentos não somente pessoas-pessoas, pessoas-coisas, mas também o relacionamento coisas-coisas, de modo com que objetos físicos pudessem trocar informações entre si, sem a necessidade de intervenção humana e realizassem ações de acordo com os dados recebidos. Deste modo, oportunidades e conexões se tornaram praticamente infinitas, segundo Roger (2017). Uma dessas oportunidades se deu no meio agrícola, por meio da possibilidade de utilização de sensores para medição de temperatura, umidade do ar, pH e umidade do solo, luminosidade, dentre outros, necessários para determinados tipos de plantações, possibilitando o acompanhamento e monitoramento em tempo real de cargas e lavouras, por exemplo.

O sistema foi desenvolvido com a utilização do microcontrolador NodeMCU ESP8266 e de sensores conectados ao mesmo (para medição de temperatura, umidade do ar, umidade do solo e luminosidade), que realizam o monitoramento dos dados em tempo real e enviam as informações para um servidor *web*. Deste modo, o servidor envia as informações para um sistema que realiza a gestão e o controle dos dados. O projeto, portanto, utiliza da tecnologia IoT, juntamente com o conhecimento das disciplinas de redes, microcontroladores e arquitetura de computadores, além das disciplinas voltadas para o desenvolvimento de *software*.

1.1 Problemática

A principal problemática da tecnologia aplicada à agricultura não está na falta de recursos tecnológicos em si, mas sim na falta de recursos financeiros. Os custos de possuir máquinas inteligentes é elevado, suas possibilidades de vantagens são imaturas (SILVA; MUXITO, 2018) e agricultores de pequeno e médio porte muitas vezes não possuem acessibilidade aos recursos,

por exemplo. Embora a utilização de tecnologia traga resultados satisfatórios para aqueles que fazem a sua utilização na agricultura, ela ainda não está acessível para todos os tipos de produtores rurais. Com isso, produtores rurais de pequeno e médio porte não possuem a oportunidade de competição com gigantes do mercado e lucratividade com as grandes indústrias e a falta de inovação no campo dificulta a situação.

Tendo em vista a problemática, este trabalho visa facilitar e viabilizar a implementação de um sistema de monitoramento voltado para pequenos e médios produtores rurais, de modo com que os mesmos possam obter vantagens no campo sem a necessidade de um elevado custo nos investimentos com tecnologia.

1.2 Justificativa

A implementação deste projeto resolve o problema apresentado anteriormente pois a tecnologia aliada à produção agrícola pode aumentá-la e aprimorá-la, e seu monitoramento possibilita uma produção com menos impactos ambientais e menos desperdícios.

A tecnologia IoT utilizada neste projeto é relevante visto que é uma tecnologia de ponta, moderna e que pode ser aplicada em diversas áreas, causando impactos positivos se aplicadas adequadamente e neste caso, aprimorar o serviço manual do produtor rural. Com isso, os ganhos voltados para o aprimoramento tecnológico aplicados na agricultura são diversos, uma vez que são analisados vários aspectos de aplicabilidade da tecnologia (COSTA; OLIVEIRA; MOTA, 2018)

Por fim, pode-se concluir que a grande motivação deste projeto é a oportunidade de atuar em uma área de alta relevância econômica e social (agricultura) e principalmente de ter a chance de colaborar com os inúmeros agricultores de médio e pequeno porte existentes no Brasil.

1.3 Objetivos

O objetivo deste trabalho é o desenvolvimento de um sistema acessível que monitore, por meio da *Internet* e dados enviados em tempo real, através de um microcontrolador NodeMCU ESP8266 e sensores acoplados ao mesmo, uma propriedade rural de pequeno/médio porte. O trabalho visa o baixo custo, a fácil implementação em propriedades rurais e a utilização da tecnologia para monitoramento e melhoria de resultados no campo.

1.3.1 Objetivo Geral

Desenvolver um sistema utilizando microcontrolador e sensores para monitoramento e melhoria de resultados em propriedades rurais de pequeno/médio porte.

1.3.2 Objetivos Específicos

Dentre os objetivos específicos, destacam-se:

- Identificar dados relevantes ao produtor rural que possam impactar de maneira significativa a melhora na produção;
- Avaliar sensores e equipamentos de melhor custo/benefício para monitoramento da plantação;
- Desenvolver sistema de *hardware* utilizando o microcontrolador e sensores para monitoramento da produção/propriedade;
- Obter informações sobre serviços de rede para construção de página *web* hospedado em *webserver* para envio de dados em tempo real;
- Realizar o controle e a gestão de dados dos sensores através da *Internet*;
- Monitorar atividades e permitir o armazenamento de medições para consultas e comparações futuras.

1.4 Metodologia

A metodologia de pesquisa será baseada em dois pilares:

1. A pesquisa teórica, onde serão realizados os estudos bibliográficos para embasamento da teoria e elaboração da monografia;
2. A pesquisa prática será realizada através de experimentos utilizando o microcontrolador NodeMCU ESP8266 e sensores (DHT11, LDR e Higrômetro), envio de dados na rede, criação de protótipo e de uma interface de controle dos dados.

A coleta, análise de dados e envio dos dados serão realizados através de experimentos com o microcontrolador, por meio de testes práticos de sensores, envio e recebimento de informações na *Internet*, testes com o protótipo e armazenamento de dados para manter histórico de leituras.

Como o trabalho não envolve cálculos e resultados numéricos, não será realizada a tabulação e análise dos dados calculados. Deste modo, os dados analisados serão resultados de medições físicas, que serão verificados na parte do trabalho em que será feita a análise de sensores e aquisição de dados.

2 Fundamentação Teórica

Neste capítulo serão descritas, de maneira teórica, as principais ferramentas e principais conceitos utilizados para o desenvolvimento deste trabalho. Por meio de tais explicações, será possível melhor entendimento do projeto e tecnologias que o compõe.

2.1 Microcontroladores

Define-se microcontrolador como sendo um computador em um único *chip* (*single-chip computer*, em inglês), sendo um conjunto de *hardware* e *software* capaz de executar determinadas ações através da programação prévia do circuito via *software*. Este dispositivo é muito utilizado em aplicações da microeletrônica e possui diversas funcionalidades aplicadas inclusive em nosso cotidiano, como, por exemplo, em eletrodomésticos inteligentes e cartões de crédito utilizados pelos consumidores (PENIDO; TRINDADE, 2013).

Os microcontroladores são os sucessores dos circuitos digitais e são dispositivos que, além de menos complexos e mais compactos, conseguem ainda ser mais baratos que os circuitos digitais. Sendo assim, os microcontroladores vieram para substituir a lógica das portas digitais por uma junção de *softwares* e processadores (PENIDO; TRINDADE, 2013).

Neste trabalho também será encontrado o termo MCU - *Micro Controller Unit* - que designa igualmente o microcontrolador.

Apesar de seu tamanho reduzido, os dispositivos contam com um processador (núcleo), circuitos de memória, de dados, de *clock*, interfaces de comunicação serial, de entrada e saída, dentre outros, tudo isso em um único *chip*. Além disso, atualmente e cada vez com mais frequência, é possível encontrar placas de desenvolvimento vinculados a microcontroladores. Tais placas de desenvolvimento são placas de circuito em conjunto com periféricos, que facilitam a utilização de suas funções e sua programação (ELETRONJUN, 2019). Como exemplo, tem-se as famosas placas da Arduíno (Uno, Mega, Nano), placas MSP, ESP e placas NodeMCU.

Mais a frente, no capítulo 3, seção 3.1, serão abordados com mais detalhes dois microcontroladores: Arduíno Uno e NodeMCU ESP8266.

2.2 Banco de Dados

Os bancos de dados são utilizados para realizar o armazenamento de informações, podendo ser usados em inúmeros cenários (SOUZA, 2020), realizando o registro desde informações pessoais, imagens, datas até o registro de dados de sensores, como no caso deste projeto.

Um banco de dados armazena dados de maneira organizada e estruturada de maneira eficiente. São vários os motivos para realizar o armazenamento, como, por exemplo, para registro histórico, consulta futura, para segurança, geração de relatórios, gráficos, etc. Dentro de um sistema de gerenciamento de banco de dados (também conhecido como SGBD) é possível realizar pesquisa, manipulação, alteração, exclusão de dados específicos, selecionados pelo usuário administrador do banco de dados.

De acordo com o [Souza \(2020\)](#), dentre as vantagens da obtenção de um banco de dados, pode-se citar: a diminuição de riscos de operação devido à transparência dos dados, o fortalecimento da segurança, tendo em vista o acesso restrito e manipulação dos dados somente de pessoas autorizadas e (especialmente para o caso deste projeto), o auxílio nas tomadas de decisões, uma vez que com os dados em mãos, é possível tomar decisões com bases reais, com possibilidade de avaliação de cenários de forma transparente e alicerçado em fatos reais.

2.3 HTML

HTML significa Linguagem de Marcação de Hipertexto (em inglês, *HyperText Markup Language*) e, ao contrário do que pensam, não é uma linguagem de programação, mas sim de marcação, como seu próprio nome diz. Assim sendo, pode ser considerado o esqueleto da página *web*, entretanto, não é possível realizar a criação de funcionalidades dinâmicas através desta linguagem. Dentre as camadas de desenvolvimento envolvendo HTML, CSS e JavaScript, o HTML é o responsável pela exibição e organização da informação em páginas *web* e atribuição de significado ([TABLELESS, 201-](#)).

As informações contidas em um documento HTML são codificadas através de *tags*, que indicam o início e fim dos elementos. Como exemplo, tem-se as tags de parágrafos, títulos, cabeçalhos, quebra de linhas, botões, *scripts*, dentre outros ([TABLELESS, 201-](#)).

A linguagem foi inventada na Suíça, pelo físico Tim Berners-Lee, em meados de 1991. Desde então, novas versões do HTML são lançadas periodicamente, com adição de novos atributos e *tags*. Sua versão inicial continha 18 *tags* e devido a sua popularidade e acelerada ascensão, atualmente estima-se que haja cerca de 140 *tags*, com algumas delas, todavia, já não suportadas pelos navegadores ([LONGEN, 2019](#)).

Considerado o padrão oficial da *web*, possui uma vasta comunidade de desenvolvedores e, além de rodar em qualquer navegador, seu aprendizado é considerado fácil, segundo [Longen \(2019\)](#) e seu código pode ser escrito em qualquer editor de texto. Sua versão mais recente é o HTML5, lançada em 2014 com vários novos recursos adicionados à linguagem.

2.4 CSS

Cascading Style Sheets, nome completo do termo conhecido como CSS, pode ser descrito, em tradução livre, como Folha de Estilo em Cascatas. Comumente utilizado em conjunto com linguagens de marcação, HTML, por exemplo (GONÇALVES, 2019), trata-se, de maneira básica, da aparência e estilo das páginas *web*, separando o conteúdo da representação visual do *site*. Com ele, é possível alterar cores de fundo, textos, botões, *links*, ajustar fontes, dentre outros. Além disso, segundo Pereira (2009), grande parte dos menus em cascata, estilos de cabeçalho e rodapé das páginas *web* são desenvolvidos com esta linguagem.

Segundo Gonçalves (2019), a linguagem CSS foi desenvolvida em 1996, pelo *World Wide Web Consortium*, mais conhecido com W3C. Como o HTML não foi concebido para ter as *tags* responsáveis pela formatação da página, o CSS nasceu para solucionar este problema. Devido a isso, HTML e CSS possuem uma forte relação, andando juntos na construção e elaboração do desenvolvimento *web*.

Dentre suas vantagens, destacam-se a separação entre o estilo e conteúdo de uma página *web* (PEREIRA, 2009), preservação limpa da marcação HTML, economia de tempo, de código e diminuição de erros. Além disso, com apenas um arquivo CSS, pode-se tratar de praticamente todos os estilos de todos os arquivos HTML do *site*, facilitando e simplificando a manutenção e evolução dos conteúdos de estilo (GONÇALVES, 2019).

A estrutura desta linguagem é bem simples: basicamente, cada declaração abrange um nome de propriedade CSS e um valor. As declarações são separadas por dois pontos, terminam com ponto-e-vírgula e seus blocos de declaração são delimitados por chaves.

Além de tudo, os estilos CSS podem ser separados em: internos, externos e *inlines*. Abaixo, serão retratados, de forma resumida, como cada estilo trabalha:

O estilo interno é aquele desenvolvido dentro do mesmo arquivo HTML, sendo então carregado a cada atualização da página *web*, o que acarreta, entretanto, maior tempo de carregamento. Com ele não é possível o reaproveitamento do estilo em outras páginas, uma vez que seu código está inserido em um arquivo HTML específico.

Segundo Gonçalves (2019), o estilo externo pode ser considerado o mais conveniente e o mais utilizado entre os desenvolvedores. O desenvolvimento do estilo da página é realizado em um arquivo CSS distinto do arquivo HTML e, com isso, pode ser aplicado a qualquer página almejada. Além disso, este tipo de desenvolvimento pode otimizar o tempo de carregamento da página. Neste estilo, assim que o código de CSS é desenvolvido, é necessária a integração deste arquivo ao documento, comumente realizado através de referência no código HTML.

Por último, há o estilo *inline*. Nesta modalidade, o estilo CSS é inserido em elementos específicos (por exemplo: parágrafos, cabeçalhos, *links*, *divs*, etc.) desde que associados à *tag* `<style>`. Sendo assim, somente um único elemento é estilizado por vez e pode ser

utilizado quando, por exemplo, o desenvolvedor HTML não possui acesso aos arquivos CSS (GONÇALVES, 2019).

Em adição, o CSS conta com alguns elementos básicos, como as *id's*, classes e *tags* do HTML. Segundo Vieira (2016), a principal diferença entre as classes e os *id's* é que a classe pode ser utilizada quando vários elementos tem a necessidade dos mesmos parâmetros, enquanto os *id's* só podem ser utilizados em elementos únicos, ou seja, somente um *id* pode ser utilizado por *tag* HTML.

A versão mais recente da linguagem é a CSS3 e foi lançada no ano de 2010. Dentre as melhorias desta versão, tem-se: possibilidade de animações, elaboração de cantos arredondados, efeitos de transição e gradientes, sombras, além da maior flexibilidade na criação de *layouts* e maior autonomia para os desenvolvedores e *webdesigners* (GONÇALVES, 2019).

2.5 Javascript

JavaScript é uma linguagem de programação muito utilizada no desenvolvimento *front-end* de páginas *web*, que, aplicada a documentos HTML, permite interações dinâmicas e flexíveis entre sites. Através dela, é possível realizar desde tarefas básicas e simples como ajuste de *layouts* e interações dinâmicas até a realização de desenvolvimento de jogos, plotagem de gráficos, criações de funções e animações gráficas (GUININ, 2019a).

Ela foi criada no ano de 1995, pelo co-fundador do projeto Mozilla, Brendan Eich, e seu propósito inicial era oferecer aos desenvolvedores soluções para tornar processos mais dinâmicos e agradáveis. Atualmente, é uma das tecnologias mais utilizadas e importantes no desenvolvimento *web*, com utilização em praticamente todo *site* existente. Além disso, a linguagem viabilizou a construção de aplicações mais dinâmicas e com conteúdos flexíveis (SILVA, 2015).

Por ser uma linguagem interpretada pelos navegadores e diversas plataformas, não necessita de compilador, uma vez que os navegadores *web* interpretam-a como HTML. Segundo Guinin (2019a), JavaScript pode ser conceituada como a terceira camada de desenvolvimento da *web*. Além de ser compacta e flexível, possui uma grande comunidade de desenvolvedores e apoiadores da linguagem, o que facilita seu aprendizado e evolução.

Em termos mais técnicos, tem-se que o JavaScript é uma linguagem de programação projetada para rodar do lado do navegador do usuário, diferentemente das linguagens de programação tradicionais que rodam majoritariamente em servidores. Seu funcionamento tem como característica rodar programas localmente, ou seja, do lado do cliente. Com isso, há vantagens como, por exemplo, a transformação e o processamento de dados enviados e recebidos, a atualização parcial de conteúdo sem a necessidade de carregamento completo da página (em conjunto com a utilização do Ajax), além de diversas possibilidades de construção

não somente de páginas *web* ou *softwares* funcionais, mas também de programas *desktop* e aplicativos *mobile* (SILVA, 2015).

Uma curiosidade ao que tange o nome da linguagem JavaScript é que esta não deve ser confundida ou associada à linguagem de programação Java. JavaScript sequer faz parte da plataforma Java. Dentre as principais diferenças tem-se que Java é uma linguagem de programação orientada a objetos, cria aplicações que podem rodar em *browsers* ou até mesmo em máquinas virtuais e além disso é uma linguagem compilada, ao passo que JavaScript é uma linguagem de *scripts* orientada a objetos, pode ser executada somente em *browsers* e seu código é interpretado, pois está sempre contido em documentos texto (HTML).

Abaixo, na figura 1, é apresentado um exemplo básico de utilização do JavaScript em um documento HTML.

Figura 1 – Exemplo de JavaScript aplicado ao documento HTML.

```

1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <h1> Exemplo de utilização de JavaScript </h1>
6
7  <p id="exemplo"> Hello Word! </p>
8
9  <button type="button" onclick="document.getElementById('exemplo').innerHTML='
10      AgronomIoT'"> Clique aqui </button>
11
12 </body>
13 </html>

```

Fonte: Elaborado pela autora.

2.5.1 Ajax

Ajax significa *Asynchronous JavaScript and XML* e, não sendo uma tecnologia e nem uma linguagem de programação, trata-se de um conjunto de técnicas de desenvolvimento *web* as quais possibilitam a criação de aplicações mais dinâmicas e interativas através de aplicações que trabalham de maneira assíncrona, onde requisições ao servidor podem rodar em segundo plano (MARQUES, 2019).

O termo foi introduzido em fevereiro de 2005, através do artigo “*Ajax: A New Approach to Web Applications*”, escrito por Jesse James Garrett, fundador e presidente da empresa Adaptive Path (CARVALHO, 2007).

Dentre um dos principais objetivos do Ajax, destacam-se a maior velocidade das respostas das páginas *web* devido ao câmbio de quantidades pequenas de dados com o servidor *web*, além de evitar o recarregamento inteiro da página *web* a cada vez que se faz necessária a consulta de alguma nova informação ao servidor, graças à sua principal característica: a comunicação assíncrona (CARVALHO, 2007).

Segundo [Marques \(2019\)](#), geralmente, um sistema que faz a utilização de Ajax é composto por :

- HTML para linguagem de marcação (principal) e CSS para a definição de estilos e apresentação da página *web*;
- *Document Object Model (DOM)* para exibição dos dados de maneira dinâmica e para interação com os elementos da página.
- *Extensible Markup Language (XML)* e *Extensible Stylesheet Language Transformations (XSLT)* para a troca de informações e para a manipulação dos dados, respectivamente.
- Objeto *XMLHttpRequest*, tecnologia que fica nos bastidores e torna possível a comunicação assíncrona. O *XMLHttpRequest* pode ser utilizado pela linguagem de programação JavaScript para fazer a transferência de XML e elementos textuais de um servidor *web* para uma página *web*, através do protocolo HTTP, permitindo controle maior na comunicação entre servidor e cliente devido a criação de canal de comunicação independente entre eles.

Por último, para abraçar todas as tecnologias citadas acima, utiliza-se a linguagem de programação JavaScript, que dentre suas características está o gerenciamento de conteúdos dinâmicos e possibilidade de interação dinâmica com o usuário, como abordado anteriormente. Em adição, vale ressaltar que ambos o JavaScript e o XML trabalham de maneira assíncrona no Ajax, de modo que qualquer aplicação que utilize este conjunto de técnicas de desenvolvimento *web* possa enviar e receber informações e dados sem a necessidade de recarregar a página *web* inteira ([MARQUES, 2019](#)).

O *Ajax* pode ser encontrado, por exemplo, em páginas como a da Google, onde sugestões de pesquisa vão aparecendo conforme o usuário digita o conteúdo a ser pesquisado, na tentativa de autocompletar a pesquisa. Um outro exemplo da utilização desta técnica é no *feed* do *Facebook*, onde, sem a necessidade de atualização da página e passando pelo *feed* de notícias, pode-se notar o surgimento de *pop-ups* de notificações automaticamente, sem que o usuário tenha que realizar qualquer ação.

2.6 Ferramentas Google

Nesta seção, serão abordadas as ferramentas da Google utilizadas no projeto. Estas ferramentas foram escolhidas devido à algumas limitações de *hardware* e, portanto, dão suporte ao projeto sem sobrecarregar o microcontrolador.

2.6.1 Google Forms

Iniciando com o *Google Forms* ou, em português, formulários Google, é um aplicativo gratuito o qual permite a criação de formulários *online* e acompanhamento de respostas (BIJORA, 2018), de modo que o usuário possa acessá-lo de qualquer lugar que esteja, desde que conectado a uma rede *Internet*.

Com este aplicativo, é possível realizar formulários com perguntas e definir a forma como a resposta será informada, como por exemplo, múltipla escolha, questão discursiva, selecionar caixas de seleção, definir uma lista suspensa, dentre outros. Além disso, é possível parametrizar o tipo de entrada da resposta, podendo defini-la como sendo número, texto, texto curto, data, hora, etc. É possível, inclusive, realizar o *upload* de arquivos, caso almejado pelo usuário.

Em adição, com o *Google Forms* é possível visualizar as perguntas e suas parametrizações, bem como acompanhar as respostas enviadas pelos usuários de maneira visual, através de gráficos gerados pelo próprio aplicativo. Também é possível realizar configurações gerais do questionário (como quantidade de respostas por usuário, coletar *e-mails* de respostas), configurações de apresentação (por exemplo: embaralhar ordem das perguntas, mostrar barra de progresso) e até criar testes com pontuações e correções automáticas das respostas.

2.6.2 Google Sheets

Google Sheets (em português Planilhas Google) é outro aplicativo gratuito da empresa que permite aos usuários criar, editar e compartilhar planilhas *onlines*. Além da vantagem do documento estar na nuvem e poder ser compartilhado com outras pessoas, seu salvamento é automático conforme digitação do usuário ou alteração da planilha, além de ser compatível com o *Excel* e não necessitar de conversões, segundo o próprio site da Google.

Além de todas suas vantagens, o Planilhas Google permite também integração com o Formulários Google. Deste modo, após recebimento das respostas no formulário e associação do formulário com a planilha desejada, os dados integram-se automaticamente. Sendo assim, todas as respostas são enviadas e salvas automaticamente na planilha, sem necessidade de intervenção do usuário.

2.6.3 Google Charts

Google Charts ou Gráficos Google, em português, é uma biblioteca da Google que permite a criação e interação com inúmeros tipos gráficos. Com ela, é possível desenhar gráficos como os de linha, pizza, histogramas, Gannt, Gauge, coluna, dentre muitos outros que são oferecidos pela biblioteca.

Como se não bastasse a variedade de tipos de gráficos possíveis, a *Google Charts Tools*

também permite customizar os gráficos, customizar opções de eixos, interagir com os gráficos (por meio de eventos, animações, *toolbars*, etc) e também, em alguns casos, permite também a plotagem de múltiplos gráficos em uma página *web*.

Em um sistema que trabalha com dados, a geração de gráficos é uma rica e visual fonte de informação, tendo em vista que gráficos são mais atrativos que simples números ou dados. Todavia, Douglas (2012a) aborda que construção de gráficos usualmente oferece certa dificuldade em qualquer linguagem de programação. A biblioteca da Google é simples, todavia muito poderosa e versátil.

Apresenta-se a posição de Douglas (2012a) referente à biblioteca da Google:

"(...) a *Google Chart Tools* é tida por muitos como a melhor disponível hoje em dia por ser de uma clareza muito grande, ter uma curva de aprendizagem pequena e uma documentação muito boa."

É possível encontrar, na própria documentação da biblioteca, que a maneira mais comum de utilizar o *Google Charts* é em conjunto com JavaScript simples incorporado na página *web*. Em adição, os gráficos, por serem renderizados através da tecnologia HTML5/SVG, possuem compatibilidade entre diversos navegadores, além de possuírem também portabilidade entre plataformas como as da *Apple* (como *iPads* e *iPhones*, por exemplo) e *Android*.

Ao que tange a base de dados para plotagem do gráfico, é possível definí-la de algumas maneiras, como, por exemplo, definir via código a base de dados (por meio da digitação de valores) e *dataviews*, conectar *databases* próprios, inserir dados de outras fontes, integrar os dados com o *Google Sheets*, além da possibilidade de implementação de novos tipos de fonte de dados, segundo a documentação do *Google Charts Tools*.

3 Dispositivos

Neste capítulo são apresentados os dispositivos de *hardware* utilizados na construção do protótipo. Em suma, o protótipo utiliza uma placa composta de um microcontrolador e circuitos de entrada/saída para conexão de sensores digitais e analógicos, equipamentos de prototipagem (como *jumpers* e *protoboards*) e sensores, utilizados para medição do ambiente externo.

3.1 Escolha da placa

A escolha da placa a ser trabalhada leva em conta seu custo/benefício, funcionalidade, facilidade de programação e utilização e sinergia com a tecnologia IoT. A placa escolhida foi a NodeMCU ESP8266, uma placa recente no mercado e que possui características aderentes ao projeto. Entretanto, para decisão da placa, foi realizado um estudo e uma comparação que poderá ser vista nos tópicos [3.1.1](#) e [3.1.2](#).

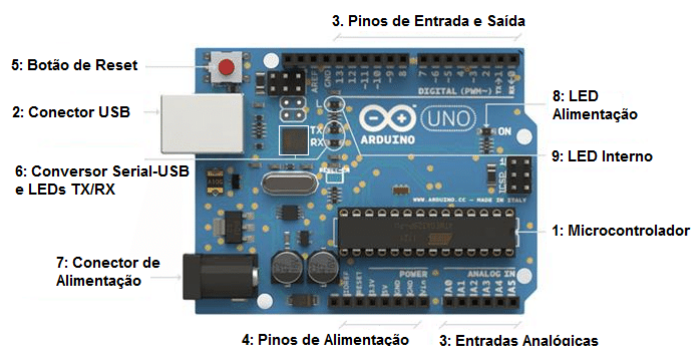
3.1.1 Arduíno Uno

O *Arduíno* é uma plataforma eletrônica de *hardware* e *software open-source*, ou seja, é uma plataforma de código aberto onde qualquer pessoa pode alterar, aperfeiçoar e personalizar a plataforma ([THOMSEN, 2014](#)). Seu *hardware* é extensível e é utilizado para construção de projetos e protótipos, enquanto o *software* é utilizado para programação da placa, que utiliza a linguagem de programação semelhante à C/C++, em conjunto com a *IDE Arduíno*, que pode ser baixada gratuitamente pelo *site* da *Arduíno*.

A mais famosa das placas é a *Arduíno Uno*, que possui uma boa configuração para abrangência de grande parte dos projetos, segundo [Mota \(2017b\)](#). Esta placa analisada é de fácil utilização e um bom custo/benefício, além de possuir diversos conteúdos na *Internet*. Entretanto, até a versão analisada (R3), foi visto que a placa não possui conectividade com a rede sem a utilização de módulos acoplados ou *shields*, sendo necessária a aquisição dos mesmos, o que elevaria não somente a complexidade do projeto, mas também seu custo.

Encontra-se na Figura 2, um exemplo da placa Uno, da Arduino.

Figura 2 – Placa Arduino Uno.



Fonte: [Mota \(2017b\)](#)

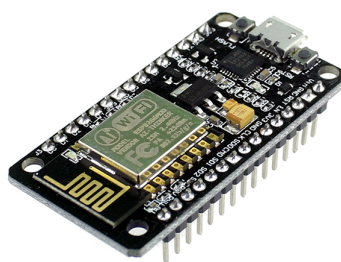
3.1.2 NodeMCU ESP8266

O módulo NodeMCU ESP8266 é uma placa de desenvolvimento que une o *chip* ESP8266 com a plataforma *open-source* NodeMCU e foi criado para o desenvolvimento de projetos IoT, segundo [Oliveira \(2016\)](#). Sua programação pode ser realizada usando tanto linguagem LUA quanto a própria IDE do Arduino, que possui linguagem de programação baseada no C/C++. Além disso, sua comunicação é realizada via cabo micro-USB, cabo comumente utilizado por celulares que utilizam o *Android* como sistema operacional.

Dentre as características relevantes sobre o módulo, pode-se destacar o *WiFi* nativo, baixo custo, baixo consumo de energia e seu tamanho reduzido. Todavia, uma grande desvantagem dá-se devido a pinagem reduzida da placa, contando com 13 pinos digitais e somente 1 pino dedicado à entrada/saída analógica. É possível fazer a expansão de tais pinos com a aquisição de circuitos integrados disponíveis no mercado ([OLIVEIRA, 2016](#)), entretanto a pinagem já existente na placa foi suficiente para suprir todas as necessidades do projeto presente e todos os sensores desejáveis foram conectados na placa sem necessidade de adição de circuitos integrados.

Abaixo, encontra-se um exemplo do módulo, como traz a Figura 3.

Figura 3 – Módulo NodeMCU ESP8266.



Fonte: [Oliveira \(2016\)](#)

3.2 Sensores

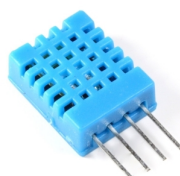
3.2.1 Sensor de umidade e temperatura - DHT11

O sensor DHT11 é capaz de realizar medições de temperatura e umidade graças ao termistor e ao sensor capacitivo que realizam a medição do ar ambiente, tudo em apenas um módulo. O DHT11 realiza medidas de temperaturas entre 0° a 50° Celsius e possui precisão de 2°C. Já na esfera da umidade, ele é capaz de medir entre 20 a 90% de umidade do ar, com uma precisão de 5% ([MOTA, 2017a](#)).

Este sensor foi escolhido para o projeto por ser um sensor simples (pinagem de fácil identificação e código não complexo), porém eficiente e devido ao seu baixo custo (custando em torno de quinze reais, aproximadamente), trazendo a possibilidade de medição de duas variáveis distintas em apenas um módulo. Além de ser acessível, o sensor também conta com diversas bibliotecas disponíveis na *internet* que facilitam a sua programação, fazendo com que seu manuseio possa ser realizado com apenas alguns comandos, sem a necessidade de aprofundamento técnico tanto na programação quanto nas especificações do sensor.

A figura 4 traz uma imagem deste sensor.

Figura 4 – Sensor DHT11.



Fonte: [Thomsen \(2013\)](#)

3.2.2 Sensor de Luminosidade - LDR

Para medição de luminosidade no ambiente é utilizado o sensor LDR, cuja sigla em inglês significa *Light Dependent Resistor*. Portanto, como já diz o próprio nome, é um resistor dependente de luz. Por ser constituído de um semicondutor de alta resistência, este sensor tem sua resistência menor quando há alta incidência de luz e alta resistência em ambientes com pouca iluminação (MOTA, 2017c).

Este sensor foi escolhido devido sua versatilidade podendo ser utilizado desde projetos complexos até projetos simples, como utilizar o sensor para ligar uma lâmpada sozinha em ambientes com pouca luminosidade ou ser utilizado para realizar automação residencial, por exemplo. Além disso, é um sensor simples (possuindo somente dois pinos para inserção na *proto board*) e barato, podendo ser encontrado com facilidade em lojas de eletrônica, além de possuir diversos tamanhos, que variam desde os 3mm até 12mm.

Neste projeto é utilizado um sensor LDR de 10mm conectado à única porta analógica presente no NodeMCU ESP8622, sendo, portanto, o único sensor analógico possível a ser utilizado no projeto. Devido ao conversor ADC (conversor analógico para digital), o sensor informa um valor entre 0 e 1024, que sem tratamento não informa ao usuário valores relevantes para o monitoramento. Portanto, este valor é convertido posteriormente em tensão, a partir da fórmula segundo Martins (2018):

$$V_s = \frac{V_{cc}}{R_a + R_{ldr}} \times (R_a) \quad (3.1)$$

onde,

V_s é tensão de saída

V_{cc} é tensão da fonte

R_a é resistência colocada em série com o sensor e

R_{ldr} é a resistência do sensor LDR.

Após cálculo da voltagem, pode-se calcular a variável Lux, que é a intensidade luminosa por unidade de área, através de uma regra de 3 simples, explicada detalhadamente abaixo:

Se 3,3v (tensão máxima permitida pelo NodeMCU ESP8266) é equivalente a 1024 (valor recebido do ADC), então a tensão obtida através da fórmula é igual x.

Com a variável x é possível determinar um limiar entre boa e baixa luminosidade, podendo, finalmente informar ao usuário se o ambiente possui boa ou baixa iluminação.

A figura 5 apresenta uma ilustração de um sensor LDR de tamanho 5mm.

Figura 5 – Sensor LDR.



Fonte: [Mota \(2017c\)](#)

3.2.3 Sensor de Umidade de Solo - Higrômetro

Por último, mas não menos importante, apresenta-se o sensor de umidade de solo, podendo ser chamado também de higrômetro.

Com este sensor é possível realizar a medição da umidade do solo, tanto de maneira digital quanto de maneira analógica (conectando-o a uma entrada ADC no microcontrolador). Esta é uma das principais vantagens deste sensor, pois permite sua utilização de acordo com a necessidade do usuário, sem limitar sua usabilidade. O higrômetro pode ser usado em projetos para realizar o controle da umidade do solo e informar ao produtor quando o solo está seco, por exemplo.

Seu funcionamento se dá pela aferição da condutividade do solo, medida por meio de uma haste que possui dois eletrodos. Com isso, através da condutividade é possível estimar quando e/ou quão úmido ou seco o solo está. Ou seja, quando o solo está úmido, há uma melhor condutividade em consequência da absorção da água, o que acarreta em um maior fluxo de corrente entre os dois eletrodos presentes no sensor. Por sua vez, quando o solo está seco, há pouca ou nenhuma corrente entre os eletrodos ([ALMEIDA, 2017](#)).

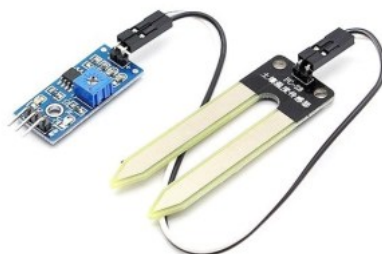
Como mencionado anteriormente, o sensor permite realizar a medição de umidade de duas maneiras e neste projeto o higrômetro é utilizado como um sensor digital. Sendo assim, o sensor informa somente se o solo está seco ou não a partir de uma calibração que é realizada manualmente em um potenciômetro embutido. Trabalhando como um sensor digital, sua precisão consequentemente é menor, entretanto, não necessita de um conversor ADC e pode ser conectado à porta digital no NodeMCU (lembrando que o microcontrolador utilizado possui somente uma porta analógica e a mesma já é utilizada pelo sensor de luminosidade LDR).

A única diferença e desvantagem deste sensor para os demais se dá devido a sua durabilidade. Por ser um sensor que possui hastes metálicas usadas para medição e por atuar em contato com o solo (terra + água), suas partes metálicas oxidam com mais facilidade, deteriorando o sensor. Por este motivo, os testes com este sensor foram realizados ao final do

projeto, afim de garantir sua integridade por mais tempo possível para fins de apresentação.

A figura 6 traz uma imagem deste sensor.

Figura 6 – Sensor de umidade de solo - Higrômetro.



Fonte: Almeida (2017)

4 Desenvolvimento

Neste capítulo é descrito como foi realizado o desenvolvimento do projeto. O desenvolvimento foi segregado em etapas de modo a facilitar e organizar a elaboração de todas as etapas, garantindo o melhor desempenho e otimização de tempo e esforço. Ao final de cada etapa, foram realizados testes de modo a garantir a integração de todos os processos (códigos no *software* Arduino IDE em conjunto com o desenvolvimento da página *web* e banco de dados) ao final do projeto.

4.1 Estudo de *hardwares* e *webservers*

A etapa de estudo de *hardware* foi composta com o estudo e análise dos sensores e do microcontrolador utilizado, além das possibilidades de servidores *web*.

O estudo do microcontrolador foi realizado inicialmente para avaliar os comportamentos e possibilidades do NodeMCU ESP8266. O estudo da placa permitiu o reconhecimento das portas digitais e da porta analógica, pinos de voltagem de 3,3v e pinos ligados ao *ground*, além de reconhecimento do módulo embutido de *WiFi* e botões de *reset*. Tal conhecimento foi necessário para início de trabalho em conjunto com os sensores, de modo a estabelecer as conexões corretas para que nenhum componente fosse danificado ou queimado durante a elaboração do projeto. Em adição, também foram realizados testes de conexão com a *Internet* através do módulo ESP8266 embutido e biblioteca respectiva, além de testes para utilização do NodeMCU como servidor *web* e conexões com o *SPIFFS*, que será abordado na seção 4.3.

O estudo de sensores foi realizado levando em conta suas funcionalidades, seu tipo de saída (analógico ou digital), seu custo benefício e suas diferenças de precisões.

Na subseção 4.1.1 será abordada a discussão referente à utilização de *webservers* para hospedagem do site *versus* a utilização do próprio NodeMCU como servidor *web*.

4.1.1 NodeMCU ESP8266 como *WebServer* x Hospedagem em *WebServers*

Para estabelecimento do servidor *web*, foram estudadas duas opções, as quais serão abordadas nos parágrafos seguintes. A primeira opção pensada e estudada foi a de hospedagem dos arquivos HTML e CSS em servidores gratuitos na *internet*. Após pesquisas, ponderações e descobertas, foi considerado também o estudo da utilização do NodeMCU ESP8266 como servidor *web*.

Para início da abordagem do assunto, tem-se a definição de servidor *web* (ou *webserver*, em inglês), que segundo a Redação (2017) é o responsável pelo atendimento das requisições

realizadas para um determinado endereço *web*. De forma mais lúdica, um servidor *web* pode ser definido como sendo um computador responsável pela hospedagem de uma ou mais aplicações na *internet* para que possam ser requisitados e acessados pelos clientes, mas também pode ser definido como um *software*, o qual atende às solicitações do tipo HTTP (em tradução ao português, é o protocolo de transferência de hiper-texto). De forma simplificada, um servidor HTTP realiza o envio e recebe respostas referentes aos arquivos que constituem um *site*, sendo o HTTP o padrão usado para comunicação cliente/servidor.

O site [Oficina \(2010\)](#) bem define um servidor *web* como sendo um programa de computador capaz de aceitar requisições HTTP de clientes (como navegadores, por exemplo), retornando os pedidos com respostas HTTP, com possibilidade de inclusão de páginas *web* compostas por arquivos HTML e objetos como imagens e vídeos.

Tendo esta informação em vista, foi analisado primeiramente sobre a hospedagem do site (arquivos HTML e CSS) e seus componentes (imagens para ilustração da página *web*) em um servidor gratuito da *internet*. Entretanto, sua gratuidade possui consequências, como algumas abordadas pelo site [Oficina \(2010\)](#): limite de armazenamento, baixas velocidades, a necessidade de incluir diversas propagandas no *site*, baixa disponibilidade ou estabilidade ou em geral, um baixo desempenho. Além disso, muitos servidores *web* gratuitos comprometem a segurança do *site* devido aos seus *softwares* não atualizados, além de servidores de baixo nível ([DENIAL, 2020](#)). Além disso, para utilização de servidores gratuitos, é necessário hospedar os arquivos do projeto no local especificado pelo hospedeiro, sem garantia de segurança dos arquivos ou garantia de não manipulação das informações pelo próprio *host*.

Em alternativa à hospedagem gratuita em servidores *web* disponíveis na *Internet*, foi estudada a possibilidade do próprio NodeMCU ESP8266 funcionar como um servidor. Nesta abordagem, os arquivos não ficam expostos a um terceiro, pois localizam-se na memória do próprio microcontrolador. Além disso, através de testes realizados com a placa, notou-se uma alta disponibilidade do servidor, desde que o microcontrolador esteja conectado a uma fonte de energia. Como o desenvolvimento do código para criação do servidor (com criação dos objetos cliente/servidor) é realizado internamente, dentro do mesmo arquivo de programação para leitura dos sensores, conexão à *Internet* e envio das informações para o banco de dados, o mesmo permanece seguro, sem necessidade de exposição do mesmo para os *sites* de hospedagem gratuita. Além de tudo, o acesso ao *webserver* é muito simples, sendo necessária apenas digitação do IP disponibilizado pelo microcontrolador no navegador.

Em contrapartida a essa possibilidade, como bem citado pelo [Guimarães \(2018\)](#), há limitação do envio de dados na utilização do microcontrolador funcionando como servidor *web*. Segundo o autor, após alguns testes foram detectados, ao enviar códigos em HTML com cerca de 77 linhas, erros que acabavam por prejudicar a visualização perfeita da página. Caso semelhante aconteceu durante a execução de testes do projeto, como descritos na seção [5.4.1](#).

Com todas as vantagens já citadas acima, e tendo em vista o objetivo do projeto e a

necessidade de segurança dos dados, além da disponibilidade da página para monitoramento em tempo real da produção rural, optou-se pela hospedagem dos arquivos no próprio microcontrolador, utilizando, portanto, o NodeMCU ESP8266 como servidor *web* e realizando as programações necessárias para tal propósito.

4.2 Testes unitários dos sensores

Os testes unitários dos sensores deram início ao desenvolvimento do projeto após os testes com o microcontrolador, sendo a segunda etapa trabalhada. Nesta etapa, cada sensor foi testado individualmente afim de obter mais conhecimento do seu funcionamento e observar seu comportamento. Além disso, foram testados outros sensores além dos aderidos ao projeto, afim de obter mais informações sobre os mesmos e também para realizar a seleção de quais sensores seriam, de fato, utilizados no projeto.

Os sensores foram testados através do *Arduíno IDE* e cada sensor possuiu, inicialmente, um programa individual. Após a testagem de cada um, foram realizados testes incorporando cada vez mais sensores, de modo a garantir o funcionamento e a integração dos sensores em um único programa.

4.3 Utilização do Sistema de Arquivos do NodeMCU - *SPIFFS*

Para disponibilizar os arquivos ao *webserver* para serem consumidos pelos clientes (navegadores de *internet*, por exemplo), tais registros são armazenados no próprio microcontrolador, graças à memória *flash SPI* que está presente nos módulos ESP8266. Esta memória permite o armazenamento de arquivos e dados ([BAUERMEISTER, 2017](#)) através de seu sistema de arquivos, o SPIFFS (*SPI Flash File System*).

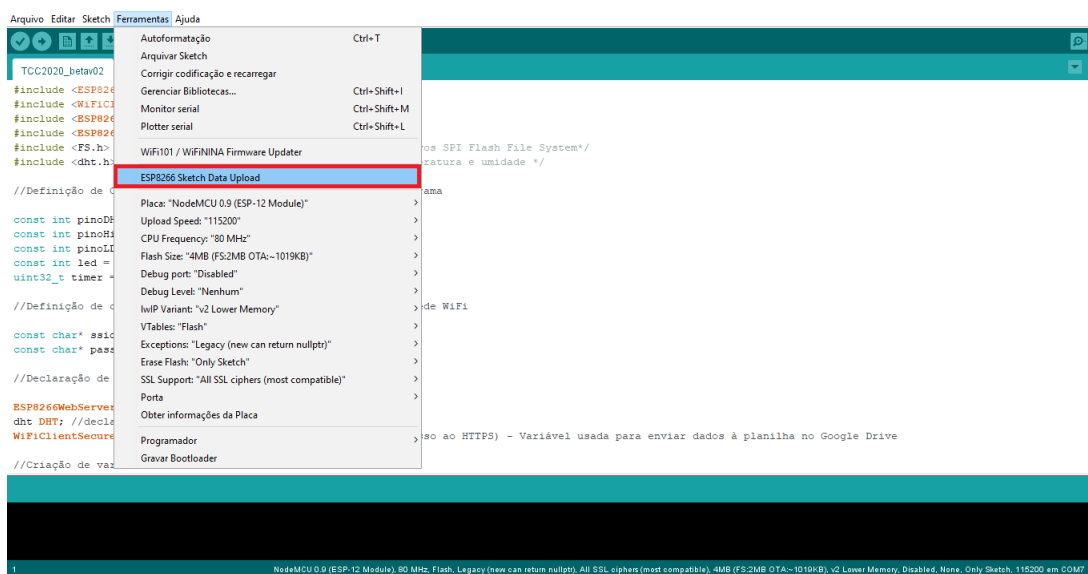
Com este recurso, é possível, por exemplo, registrar e armazenar dados de sensores em arquivo texto, salvar estados de sensores ou *LEDs* na memória e inclusive hospedar páginas *web* (arquivos HTML, CSS e JavaScript) e imagens para exibi-los em navegadores, como é o caso deste projeto. Uma outra vantagem é que os dados gravados mantêm-se mesmo na falta de energia.

Para tal, é necessária utilização de uma biblioteca denominada *FS.h*, que viabiliza básicas manipulações de arquivos, além de ser nativa da IDE Arduíno. Dentre as possibilidades de utilização de arquivos, destacam-se as funções básicas como *begin* para iniciar o sistema de arquivos, *open* para realizar a abertura do registro, *close* para fechar o arquivo manipulado, *write* para escrever em um arquivo determinado pelo usuário, *read* para realizar a leitura de um documento. Dentre outras funções, encontram-se funções de remover informações, renomear arquivos, verificar a existência de um registro, verificar seu tamanho, a posição de determinado dado, dentre outros.

Sua sintaxe é bem simples e didática, além de ser muito semelhante com a manipulação de arquivos da linguagem C.

Todavia, para salvamento do arquivo na memória *flash* do NodeMCU ESP8266, é necessária instalação de uma ferramenta na IDE Arduíno que permite o carregamento destes arquivos inteiros para a memória. Após o procedimento de *download* e instalação na IDE, é criada a nova ferramenta de *upload* no menu "Ferramentas", como pode ser observado na figura 7.

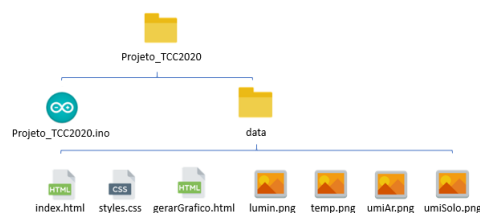
Figura 7 – Demonstração da ferramenta para *upload* de arquivos na memória *flash* do microcontrolador.



Fonte: Elaborado pela autora

Em adição, para que as informações sejam gravadas permanentemente, o arquivo deve estar inserido em uma pasta denominada "*data*". Vale ressaltar que esta pasta deve estar dentro da pasta do projeto. Para simplificar o entendimento deste parágrafo, será utilizada a figura 8, que resume, de maneira visual e verídica, o que foi tratado anteriormente. A imagem representa fielmente os arquivos deste projeto em questão.

Figura 8 – Diagrama representativo da organização dos arquivos do projeto.



Fonte: Elaborado pela autora

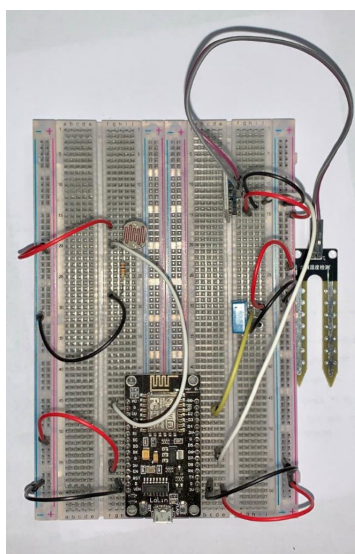
4.4 Elaboração do protótipo

Para elaboração do protótipo foram utilizadas duas *protoboards* de 830 pontos em conjunto e suas conexões com os sensores foram realizadas através de *jumpers*. Uma *protoboard* é uma placa com centenas de furos que permite a montagem de circuitos e realização de testes com componentes sem a necessidade de soldá-los. Os componentes (neste caso, os sensores e o microcontrolador) são encaixados através de pinos embutidos, o que permite flexibilidade na montagem e possibilidade de alteração e substituição de maneira simples e rápida.

Os *jumpers*, por sua vez, são fios metálicos, muitas vezes encapados de plástico e com uma ponta metálica que realizam a condução da eletricidade de um local para outro, permitindo assim a interação dos componentes.

No protótipo, retratado na figura 9, pode-se observar a conexão dos sensores DHT11, LDR e Higrômetro com o microcontrolador através de *jumpers* nas *protoboards*.

Figura 9 – Protótipo do *hardware* com sensores e microcontrolador.



Fonte: Elaborado pela autora

4.5 Armazenamento de Dados

Ao pensar em armazenamento de dados, a primeira ideia obtida foi a concepção de um banco de dados tradicional, hospedado inicialmente em uma máquina servidora. O ponto deste armazenamento é a inexistência do conceito de IoT, uma vez que os dados estariam acessíveis somente localmente, com a necessidade da máquina estar ligada e conectada à rede *Internet*. Em caso de desligamento ou queima do HD do computador, os dados não estariam disponíveis e poderiam inclusive serem perdidos.

A segunda concepção de armazenamento foi a ideia de instalação de um banco de dados em um servidor *on-line*. Na *internet* encontram-se diversos servidores *on-lines* e gratuitos, com armazenamento limitado, mas suficiente para o armazenamento das variáveis medidas pelos sensores. Para utilização destes tipos de servidores, é necessária a criação de conta nos *sites* de *webservers*, entretanto, não pode-se garantir total confidencialidade das informações, uma vez que não é sabida a precedência destes servidores *web* e nem sobre a possibilidade de compartilhamento das informações lá colocadas com outras organizações. *Webservers* conhecidos e que possuem certificado de segurança, maior estabilidade e disponibilidade como *HostGator*, *Locaweb* e *GoDaddy* são pagos e, portanto, suas aplicações não foram cogitadas de serem utilizadas neste projeto.

Por fim, a ideia que melhor se adequou no quesito de banco de dados, confidencialidade das informações e sinergia com o conceito IoT, foi o armazenamento das informações em uma planilha do *Google*. A planilha permite o recebimento das informações diretamente do programa no NodeMCU, além de permitir o acesso de qualquer lugar, desde que o usuário possua acesso à *internet*, devido ao conceito em nuvem utilizado pela empresa *Google*. Com isso, os dados estão sempre disponíveis para visualização e monitoramento, aderindo com excelência ao conceito IoT.

Em adição, para utilizar desta ferramenta (Planilhas do Google) não é necessária instalação de nenhum *software* nem no NodeMCU e em nenhum computador, além também da possibilidade de compartilhamento da planilha com outros usuários, caso seja desejável. Usando o conceito de armazenamento em uma planilha do Google é possível obter um banco de dados *on-line*, de simples e intuitiva visualização e manipulação e gratuito, não necessitando de um servidor dedicado para hospedar este serviço (MORAIS, 2017).

Além disso, por tratar-se da utilização de uma ferramenta da Google, tem-se ideia de alta disponibilidade e escalabilidade e por seu armazenamento ser na nuvem, seus dados podem ser acessados de qualquer lugar em que o produtor esteja, tornando possível manter o monitoramento à distância da propriedade, sem a necessidade de ter que manter um computador como um servidor ligado por 24 horas, 7 dias da semana.

4.5.1 Concepção e execução

Para criação de um banco de dados utilizando o Google Planilhas, foi necessária primeiramente a criação de uma conta no Google, de modo que as informações a serem armazenadas possuíssem uma conta Google específica para sua função.

Em seguida, foi realizada a criação de uma planilha na ferramenta, onde lhe foi atribuída um nome de acordo com sua função. Após a criação da planilha, foi realizada a criação de um formulário Google, que funciona como uma intermediária entre os dados enviados do programa da IDE Arduíno e entre a planilha armazenadora dos dados em si. No formulário intermediário, foram configuradas "perguntas", que na planilha dão lugares às colunas e as respostas, que na planilha dão lugares às linhas, com os dados dos sensores enviados pelo microcontrolador.

Após este procedimento, foi necessário vincular o formulário à planilha já criada, de modo que os dados a serem enviados para o formulário possuíssem como destino final a planilha, onde os dados seriam, de fato, armazenados.

Em seguida, foi necessário obter o *link* do formulário e seus respectivos campos de entradas das respostas (cada campo de *input text* do formulário possui um código específico, no qual os dados são enviados através desta associação), para que pudesse haver associação do envio de dados através da programação (via código).

Uma informação muito interessante e de grande valia é que o próprio *Google* já fornece na planilha uma coluna inicial contendo o campo "Carimbo de data/hora", não sendo então necessário o registro desta informação via programação, mas sendo um registro *default* da própria ferramenta (Google Planilhas).

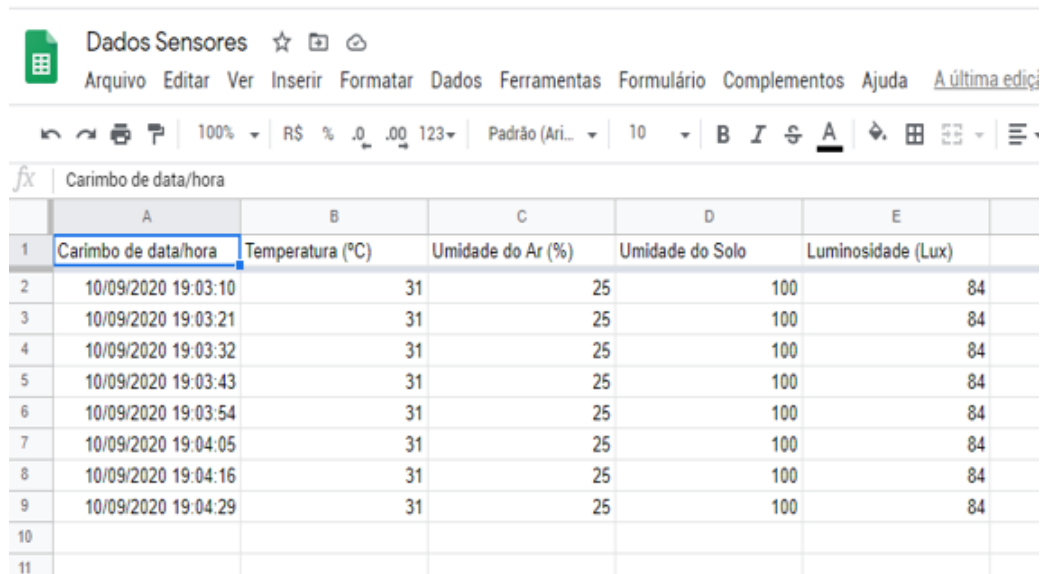
A seguir, as figuras 10 e 11 ilustram o formulário e a planilha Google utilizadas no projeto.

Figura 10 – Formulário Google - "Intermediária".

A imagem mostra a interface de um formulário Google no navegador. No topo, há uma barra de título com o nome 'Intermediária - Dados Sensores' e ícones de compartilhamento, privacidade, configurações, envio e perfil. Abaixo, há uma aba 'Perguntas' selecionada, com uma contagem de 'Respostas: 1/227'. O formulário principal contém o título 'Intermediária - Dados Sensores' e uma seção 'Descrição do formulário'. Seguem três perguntas de texto curto: 'Temperatura' (com o código '111' ao lado), 'Umidade do Ar' e 'Umidade do Solo'. À direita, há uma barra lateral com ícones para adicionar perguntas, templates, respostas, comentários e uma aba de perguntas. No canto inferior direito, há um ícone de ajuda.

Fonte: Elaborado pela autora

Figura 11 – Armazenamento de dados realizados na Planilha Google.

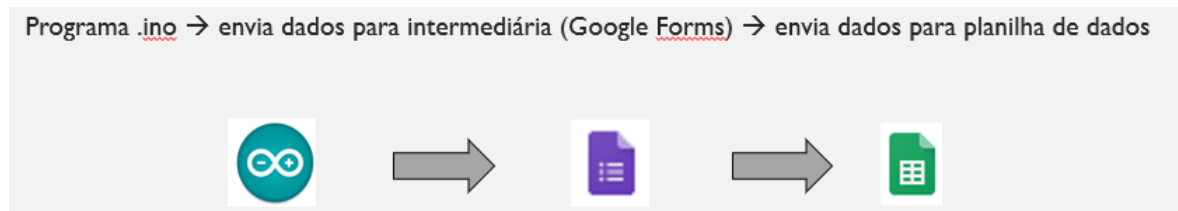


	A	B	C	D	E
1	Carimbo de data/hora	Temperatura (°C)	Umidade do Ar (%)	Umidade do Solo	Luminosidade (Lux)
2	10/09/2020 19:03:10	31	25	100	84
3	10/09/2020 19:03:21	31	25	100	84
4	10/09/2020 19:03:32	31	25	100	84
5	10/09/2020 19:03:43	31	25	100	84
6	10/09/2020 19:03:54	31	25	100	84
7	10/09/2020 19:04:05	31	25	100	84
8	10/09/2020 19:04:16	31	25	100	84
9	10/09/2020 19:04:29	31	25	100	84
10					
11					

Fonte: Elaborado pela autora

A figura 12 sintetiza a explicação realizada sobre o funcionamento da intermediária (formulário Google) e sua interação com a planilha.

Figura 12 – Síntese da explicação referente à conexão da intermediária em conjunto com a planilha Google



Fonte: Elaborado pela autora

5 Interface Desenvolvida

Após medição dos valores dos sensores e armazenamento das informações em um banco de dados *on-line*, se fez necessário o desenvolvimento de uma interface ao usuário afim de conter a consolidação das informações obtidas dos sensores. Em consequência se fez necessária uma interface de simples e intuitivo manuseio, *user friendly* e com os dados relevantes ao usuário, de modo a garantir a visualização das informações e o monitoramento da plantação.

Para desenvolvimento do *front-end*, foi utilizada a linguagem de marcação *HTML*, combinada com um arquivo *CSS* para separar o conteúdo da página de sua representação visual e a linguagem de programação *JavaScript*, usada afim de permitir a criação de funções e conteúdos dinâmicos, permitindo interatividade com a página.

5.1 Página web - HTML

Para construção da página *web* foi utilizada a linguagem de marcação *HTML*. O objetivo do *site* é informar ao usuário os valores obtidos das medições em tempo real, com possibilidade de visualização de um gráfico contendo o histórico das últimas medições.

Para visualização do gráfico, o usuário precisa apertar o botão de "Visualizar gráfico" e então é encaminhado a uma outra página *web* responsável pela plotagem do gráfico utilizando ferramentas *JavaScript* e com conteúdo gerado a partir da biblioteca *Google Charts*.

Portanto, o projeto detém dois arquivos *HTML*: um com objetivo de mostrar as informações enviadas pelos sensores e outro com objetivo de mostrar o histórico de medições de determinada variável escolhida (temperatura, umidade do ar, umidade do solo ou luminosidade) de maneira visual, através da geração de gráficos.

As páginas permitem uma navegação intuitiva e rápida e possuem somente os elementos relevantes e conteúdo essencial para monitoramento de uma produção pelos pequenos/médios produtores rurais. Devido aos problemas encontrados durante o desenvolvimento do código, abordados na seção 5.4, não foi possível a adição de outros elementos visuais na página *web*.

A figura 13 apresenta um exemplo de código da linguagem *HTML* desenvolvido no projeto.

Figura 13 – Exemplo de código HTML utilizado no projeto.

```
</script>
<a name="linkInicio"> </a> <div id="inicio">
<h1> AgronomIoT </h1>
<div id="rowImagem">
  <div id="colunaImagem">
    
  </div>
  <div id="colunaImagem">
    
  </div>
  <div id="colunaImagem">
    
  </div>
  <div id="colunaImagem">
    
  </div>
</div>
```

Fonte: Elaborado pela autora

5.2 Representação visual - Arquivo CSS

Como usualmente utilizado em projetos *web*, o arquivo CSS foi utilizado para estilizar a página HTML, de modo que os arquivos (de estruturação da página - HTML e estilo - CSS) ficassem separados e, conseqüentemente, mais organizados e estruturados.

O arquivo CSS desenvolvido para o projeto contém as características visuais de títulos, botões, corpo, *header* e tipo de rolagem da página, além de aperfeiçoar a apresentação dos visores das medições de temperatura, umidade do ar, do solo e da luminosidade, trazendo personalidade e identidade própria para o *site*.

Assim como os demais arquivos do projeto, o arquivo CSS também é disponibilizado pelo servidor para que possa ser requisitado pelos *browsers* que solicitam a página *web*, em conjunto com o arquivo HTML.

Na figura 14 encontra-se um exemplo de código CSS desenvolvido e utilizado no projeto.

Figura 14 – Exemplo de código CSS utilizado no projeto.

```
#divTemp_UmidAr
{
  height: 560px;
  position: relative;
  top: 50px;
  background: linear-gradient(to right, #ada996, #f2f2f2, #dbdbdb, #eaeaea);
}
#visor
{
  background: linear-gradient(to bottom, #dae2f8, #d6a4a4);
  width: 500px;
  height: 250px;
  position: relative;
  top: 100px;
}
```

Fonte: Elaborado pela autora

5.3 Funções em JavaScript

Uma das linguagens mais utilizadas no desenvolvimento web é a Javascript que, por ser uma linguagem flexível e interpretada pelos navegadores, pode ser utilizada para construções de aplicações de alto nível de complexidade (GUININ, 2019b). Devido às características citadas anteriormente e por possuir uma vasta comunidade de desenvolvedores, seu aprendizado é mais fácil e pode-se encontrar muito material de apoio na *internet*.

Neste projeto, o Javascript foi utilizado para permitir a criação de funções que permitem a atualização de valores dos sensores nos visores correspondentes sem a necessidade de *refresh* total da página e também a geração de gráficos através da biblioteca do *Google Chart*. Tais funções serão abordadas nas próximas seções 5.3.1 e 5.3.2.

5.3.1 Atualização automática de valores através do Ajax

Uma questão que surgiu durante o desenvolvimento do projeto foi a questão de atualizações automáticas dos valores dos sensores no *site*. Inicialmente, foram pensadas em soluções onde o próprio usuário realizasse o *refresh* da página para *update* dos valores ou definição de tempo para que, via programação, houvesse a atualização da página inteira.

Após estudos sobre tais possibilidades, foram descobertas as técnicas do *Ajax - Asynchronous JavaScript and XML* -, onde tanto o JavaScript quanto o XML trabalham de maneira assíncrona. Deste modo, toda aplicação que faça a utilização do Ajax pode tanto enviar quanto receber dados do servidor (neste caso do próprio NodeMCU ESP8266) sem a necessidade de recarregar a página inteira, segundo Marques (2019).

No caso deste projeto, as técnicas do *Ajax* foram utilizadas para realizar a atualização automática dos valores dos sensores na página *web*. Desta maneira, somente os textos contendo as medições dos sensores são atualizados, sem a necessidade de recarregamento inteiro da página.

As figuras 15 e 16 ilustram as alterações de valores dos sensores de temperatura e umidade do ar em tempos distintos através das técnicas do Ajax.

Figura 15 – Ilustração do visor das variáveis de temperatura e umidade do ar em tempo x.



Fonte: Elaborado pela autora

Figura 16 – Ilustração do visor das variáveis de temperatura e umidade do ar em tempo y.



Fonte: Elaborado pela autora

5.3.2 Geração de gráficos com *Google Charts*

O *Google Charts* é uma biblioteca disponibilizada pela Google que permite o desenvolvimento, interação e plotagem de gráficos em uma página web. A biblioteca pode ser utilizada por desenvolvedores *front-end* tanto em formatos como *json* quanto *JavaScript* (formato adotado neste projeto).

O *Google Charts* foi escolhido devido a sua curva de aprendizagem pequena, grande clareza e uma documentação muito boa (DOUGLAS, 2012b). Além disso, devido ao armazenamento dos dados estar sendo realizado no Google Planilhas, houve grande sinergia entre as ferramentas, tendo em vista que pertencem a mesma corporação Google e tais aplicações estão integradas entre si. Com isso, foi possível utilizar dados armazenados na planilha em gráficos gerados em uma página *web* customizada.

Para geração de gráficos foi utilizado um arquivo HTML distinto do arquivo da página inicial (com resumo e medições dos sensores) para segregação de código e também devido à dificuldade encontrada na plotagem do gráfico em uma única página, que será abordada na subseção 5.4.2. Este arquivo é responsável, basicamente, por gerar o gráfico de acordo com a solicitação que é enviada pelo usuário. Ou seja, caso o usuário aperte o botão de "Gerar Gráfico" no visor da unidade de temperatura, a página deve gerar o gráfico com as informações referentes à variável temperatura, e assim sucessivamente para as demais variáveis de medição do projeto.

Na programação, inicialmente, foram definidos alguns critérios da biblioteca a ser carregada pelo *Google Charts*. Portanto, houve a definição de carregamento da biblioteca atual (pode-se optar pelo carregamento de bibliotecas de versões anteriores ou versões *beta*), o pacote de gráficos a ser utilizado (neste caso, o pacote com os principais tipos de gráficos) e por fim o idioma das informações a serem exibidas.

Em seguida, houveram comparações com o parâmetro enviado da página principal, de qual unidade de medida deveria ser plotada no gráfico, passado via *URL*, com uma variável denominada "*param*". De acordo com o valor recebido de "*param*" são solicitadas,

via programação e requisição, às linhas da tabela correspondente àquela variável no Google Planilhas. Desta maneira, com a informação de quais linhas devem ser buscadas (denominado *database*, ou base de dados) e qual o código de identificação da planilha (também enviado via programação), a biblioteca consegue realizar a busca dos dados e trazê-los de forma gráfica e eficaz ao usuário.

Além da plotagem das informações, o Google Charts também permite adicionar customizações aos gráficos, permitindo melhor detalhamento dos dados exibidos. No projeto, foram aplicadas customizações de tamanho (altura e largura), título, parâmetro de título no eixo x (horizontal) e parametrização de tipo de curva a ser plotada.

Após definição da base de dados, características desejadas do gráfico e definição de local de plotagem do mesmo, o gráfico é desenhado pela biblioteca através da função *draw*.

5.4 Problemáticas do desenvolvimento da interface

5.4.1 Limitações de *hardware*

Como já informado anteriormente, para o *hardware* foi utilizado o microcontrolador NodeMCU ESP8266. O microcontrolador foi escolhido devido a suas inúmeras vantagens comparativas com relação ao Arduino Uno, todavia, ainda possui suas limitações como todo dispositivo.

A restrição encontrada neste microcontrolador que dificultou inicialmente o desenvolvimento deste projeto foi a sua pequena capacidade de memória. O NodeMCU conta com uma memória *flash* de 4 MB, sendo capaz de realizar o armazenamento de arquivos em sua memória interna. O impasse, entretanto, se deu durante a leitura dos arquivos armazenados em sua memória interna para transmiti-los ao servidor *web*. Como os arquivos HTML e CSS passaram a crescer conforme elaboração da página *web* e adição de funções e estilo, sua memória e sua pilha passaram a estourar conforme haviam outras execuções a serem realizadas pela programação. Com isso, o microcontrolador passou a se resetar com frequência e, conseqüentemente, não conseguia realizar ações como manter o servidor *web* funcionando (disponível) e enviar os dados para o Google Planilhas.

Para resolver o problema encontrado de limitação de *hardware* foi decidido construir páginas *web* concisas, porém com todas as informações necessárias para monitoramento das unidades medidas.

5.4.2 Geração de gráficos

Durante a fase de estudos e testes para criação do código em JavaScript para geração dos gráficos, foi notada uma certa limitação na plotagem de mais de um gráfico em uma mesma página *web*/arquivo HTML.

Ao buscar pela documentação da biblioteca Google Charts, a mesma indicava a possibilidade de geração de mais de um gráfico na mesma página, todavia, foi observado que tais gráficos eram gerados a partir de dados inseridos pelo próprio usuário, ou seja, gráficos eram normalmente gerados a partir do preenchimento de uma tabela de dados e não obtidos através de requisições ao Google Planilhas.

Com isso, a necessidade de buscar dados em outra aplicação inviabilizou as múltiplas plotagens de gráficos em uma mesma página, uma vez que a função responsável pela consulta em outra fonte de dados (variável de *query*) admite somente uma URL como fonte de dados. No caso deste projeto, a ideia de plotagem de quatro gráficos em uma mesma página teve de ser adaptada, uma vez que seriam necessárias buscas de quatro fontes de dados distintas, caso não suportado pela biblioteca. Todavia, mesmo com as mudanças realizadas, houve somente alteração e adaptação na maneira de execução e exibição dos gráficos, não havendo impactos no propósito final da geração dos mesmos.

Para resolução do problema foram necessários diversos testes que buscavam soluções desde alternativas dadas pela própria documentação do Google Charts, como tentativas de plotagem de gráficos com base de dados inserida manualmente, até a tentativa de criação de diversos arquivos HTML, cada qual correspondente a uma base de dados, responsável pela geração do gráfico. Contudo, muitas das ideias propostas de solução acabaram sendo inviabilizadas devido às limitações do *hardware* como abordado no tópico antecedente a este.

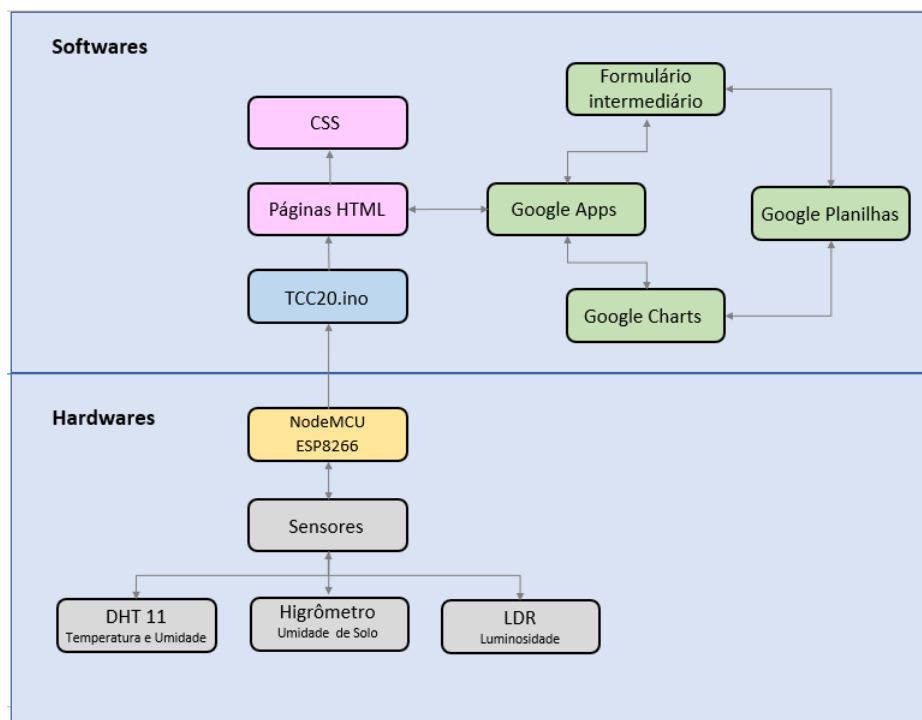
Por fim, o problema foi solucionado com a criação de apenas um novo arquivo HTML, com o código recebendo, via URL, somente um parâmetro que corresponde à base de dados que será utilizada para plotagem do gráfico. Por consequência, não houve a necessidade do servidor disponibilizar diversos arquivos em HTML e com a comparação dos parâmetros foi possível delimitar unicamente uma base de dados para plotagem do gráfico.

6 Integração final

Neste penúltimo capítulo será abordado a integração dos *softwares* e *hardwares* para funcionamento completo do sistema proposto.

Na figura 17 é apresentado o diagrama contendo todas as integrações e ferramentas utilizadas para funcionamento do projeto, com utilização do protótipo contendo os sensores, a página *web* desenvolvida e demais ferramentas utilizadas para armazenamento de informações e plotagem dos gráficos.

Figura 17 – Diagrama - Integração final



Fonte: Elaborado pela autora

Iniciando com o código na Arduíno IDE, tem-se na figura 18 um exemplo do código fonte desenvolvido ao longo do projeto. Com a programação do código, tem-se o microcontrolador conectado à uma rede *WiFi* e funcionando como um servidor *web*, ao passo em que espera uma requisição do cliente em IP específico obtido através da conexão *WiFi*. Em paralelo, há a leitura dos sensores e envio das respectivas leituras para o formulário Google que, por sua vez, realiza a integração e envio dos dados para o Google Planilhas, onde os valores são então armazenados permanentemente.

Figura 18 – Exemplo de código desenvolvido na Arduino IDE para subida dos arquivos no servidor *web*.

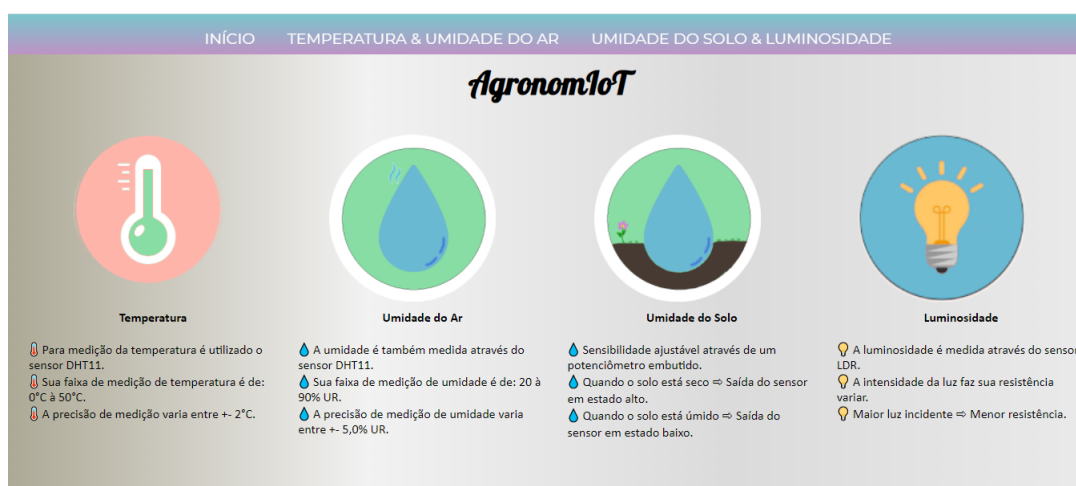
```
server.on("/", handleRoot);
server.on("/styles.css", handleCSS);
server.on("/gerarGrafico.html", handleTemp);
server.on("/icone_temperatura.png", []() { carregaImg("/icone_temperatura.png", "image/png"); });
server.on("/icone_umidade_ar.png", []() { carregaImg("/icone_umidade_ar.png", "image/png"); });
server.on("/icone_umidade_solo.png", []() { carregaImg("/icone_umidade_solo.png", "image/png"); });
server.on("/icone_luminosidade.png", []() { carregaImg("/icone_luminosidade.png", "image/png"); });
//server.on("/numIP", HTTP_GET, lerIP);
server.on("/temperatura", HTTP_GET, lerTemperatura);
server.on("/umid_ar", HTTP_GET, lerUmidAr);
server.on("/luminosidade", HTTP_GET, lerLuminosidade);
server.on("/umid_solo", HTTP_GET, lerUmidSolo);

server.on("/inline", []() {
server.send(200, "text/plain", "this works as well");
});
```

Fonte: Elaborado pela autora

Finalmente, quando a página *web* é solicitada pelo usuário (ou seja, quando o IP especificado é digitado pelo usuário na URL de seu navegador), o servidor atende à requisição através da leitura e envio dos arquivos HTML e CSS, em conjunto com o envio das imagens do *site*. Com isso, por meio da página *web* é possível realizar o monitoramento em tempo real das variáveis medidas, bem como visualizar os gráficos com histórico das medições, caso desejado. As figuras 19, 20, 21 e 22 representam visualmente tanto a página inicial do projeto, com resumos e medições quanto a geração do gráfico da variável de umidade do ar. Vale ressaltar que, caso almejado pelo usuário, é possível a visualização dos registros dos dados no Google Planilhas, através de URL específica.

Figura 19 – Ilustração da página *web* inicial contendo resumos referentes aos sensores utilizados no projeto.



Fonte: Elaborado pela autora

Figura 20 – Ilustração da página *web* inicial contendo as medições realizadas pelos sensores de temperatura e umidade do ar em tempo real.



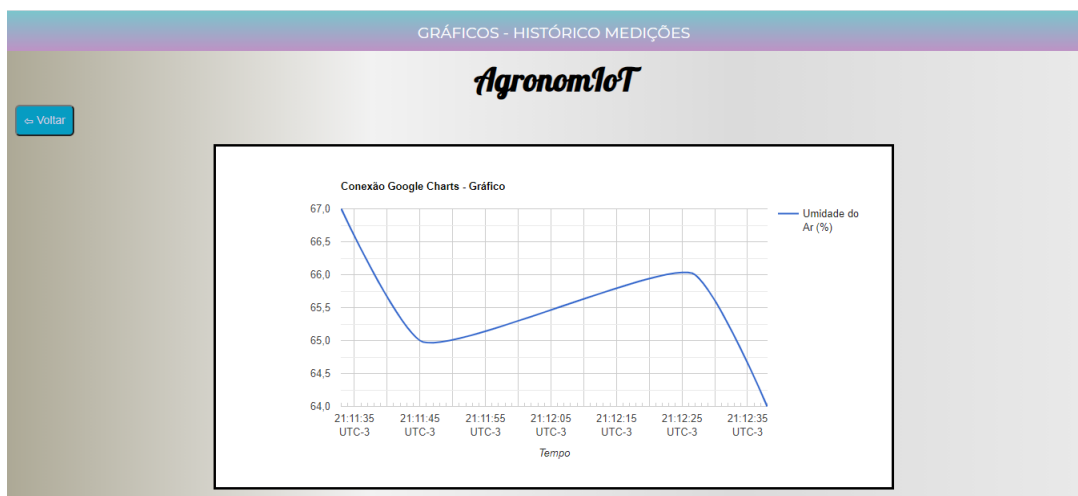
Fonte: Elaborado pela autora

Figura 21 – Ilustração da página *web* inicial contendo as medições realizadas pelos sensores de umidade de solo e luminosidade em tempo real.



Fonte: Elaborado pela autora

Figura 22 – Ilustração do gráfico gerado a partir do botão "Visualizar gráfico" do visor de umidade do ar.



Fonte: Elaborado pela autora

Portanto, para integração final foi necessária junção das partes do projeto levando em conta seu protótipo (microcontrolador e sensores acoplados na *proto board*), o código de programação realizado na Arduino IDE, o desenvolvimento das páginas *web* (arquivos HTML e CSS) e suas respectivas conexões com ferramentas da Google (Google Formulários, Google Planilhas e Google Charts).

7 Conclusão

Neste projeto de conclusão de curso foi apresentado um sistema utilizando a tecnologia IoT para monitoramento de produções agrícolas através de sensores e da *internet*, visando a fácil implementação e o baixo custo, tornando-o acessível para pequenos e médios produtores rurais, uma vez que as tecnologias presentes na agricultura possuem alto custo e exigem elevados investimentos por parte do agricultor.

Tendo em vista o propósito de baixo custo, foram selecionados sensores de acordo com seus custos/benefícios, e que agregassem informações relevantes referentes à plantação e ao seu ambiente aos seus respectivos produtores.

Além da possibilidade de acompanhamento das medições de temperatura, umidade do solo, umidade do ar e luminosidade em tempo real, o usuário também tem a capacidade de gerar gráficos com valores históricos e visualizar informações passadas em um banco de dados em *cloud*, através da utilização da ferramenta *Google Sheets*. Com isso, conclui-se que o projeto atingiu seu objetivo, mostrando resultados satisfatórios, tendo em vista a problemática abordada. Embora a precisão dos sensores não seja 100% assertiva, entende-se que o projeto busca o desenvolvimento de um sistema que possa auxiliar o monitoramento de uma propriedade rural de pequeno e médio porte, não tendo como objetivo a agricultura de precisão.

Com o sistema, é possível realizar as medições de variáveis relevantes na agricultura. Além disso, a geração de gráficos pode auxiliar o agricultor na tomada de decisões para melhoria de sua plantação.

Como possibilidade de melhoria do sistema para trabalhos futuros ou ainda como próximos passos, pode-se pensar na elaboração de um aplicativo para *smartphones*, de modo a facilitar ainda mais o acesso à informação. Em adição, poderia ser realizado o estudo de demais sensores relevantes à área agrícola, tornando o monitoramento cada vez mais completo. Além disso, poderia-se realizar a troca dos sensores atuais por sensores de melhor precisão, visando, caso desejado, a agricultura de precisão.

Referências

- ABREU, N. *IoT – Internet of Things: um caminho para a sustentabilidade*. 2019. Disponível em: <<https://autossustentavel.com/2019/08/iot-internet-of-things-um-caminho-para-a-sustentabilidade.html>>. Acesso em 03 de Março de 2020.
- ALMEIDA, D. *Sensor de umidade do solo com Arduino – Higrômetro*. 2017. Disponível em: <<https://portal.vidadesilicio.com.br/sensor-de-umidade-do-solo-higrometro/>>. Acesso em 17 de Outubro de 2020.
- BAUERMEISTER, G. *ESP8266: Gravando dados permanentes na memória Flash*. 2017. Disponível em: <<https://www.filipeflop.com/blog/esp8266-gravando-dados-memoria-flash/>>. Acesso em 01 de Maio de 2020.
- BIJORA, H. *Google Forms: o que é e como usar o app de formulários online*. 2018. Disponível em: <<https://www.techtudo.com.br/dicas-e-tutoriais/2018/07/google-forms-o-que-e-e-como-usar-o-app-de-formularios-online.html>>. Acesso em 22 de Novembro de 2020.
- CARVALHO, R. M. de. *O que é o AJAX*. 2007. Disponível em: <<https://www.devmedia.com.br/o-que-e-o-ajax/6702>>. Acesso em 02 de Novembro de 2020.
- COSTA, C. L.; OLIVEIRA, L.; MÓTA, L. M. S. Internet das coisas (iot): um estudo exploratório em agronegócios. In: . [S.l.]: VI Simpósio da Ciência do Agronegócio, 2018. p. 1,2.
- DENIAL, A. *Os 5 melhores serviços de hospedagem Web gratuita: estes são realmente bons em 2020*. 2020. Disponível em: <<https://www.websiteplanet.com/pt-br/blog/melhores-e-realmente-gratuitos-servicos-de-hospedagem-de-sites/>>. Acesso em 10 de Maio de 2020.
- DOUGLAS, A. *Introdução à Google Chart Tools*. 2012. Disponível em: <<https://www.devmedia.com.br/introducao-a-google-chart-tools/26453>>. Acesso em 22 de Novembro de 2020.
- DOUGLAS, A. *Introdução à Google Chart Tools*. 2012. Disponível em: <<https://www.devmedia.com.br/introducao-a-google-chart-tools/26453>>. Acesso em 08 de Agosto de 2020.
- ELETRONJUN. *O que são microcontroladores? Descubra suas aplicações!* 2019. Disponível em: <<https://www.eletronjun.com.br/post/o-que-sao-microcontroladores>>. Acesso em 15 de Novembro de 2020.
- GONÇALVES, A. *O que é CSS? Guia Básico para Iniciantes*. 2019. Disponível em: <<https://www.hostinger.com.br/tutoriais/o-que-e-css-guia-basico-de-css/>>. Acesso em 20 de Novembro de 2020.
- GUIMARÃES, F. *Como criar um Web Server com NodeMcu – Aula 6 – NB*. 2018. Disponível em: <<http://mundoprojetado.com.br/como-criar-um-web-server-com-nodemcu-aula-6-nb/>>. Acesso em 10 de Maio de 2020.

GUININ, D. *Saiba tudo sobre a linguagem JavaScript*. 2019. Disponível em: <<https://www.segredosdatecnologia.com/tudo-sobre-o-javascript/>>. Acesso em 02 de Novembro de 2020.

GUININ, D. *Saiba tudo sobre a linguagem JavaScript*. 2019. Disponível em: <<https://www.segredosdatecnologia.com/tudo-sobre-o-javascript/>>. Acesso em 01 de Agosto de 2020.

LONGEN, A. *O Que é HTML? Guia Básico Para Iniciantes*. 2019. Disponível em: <<https://www.hostinger.com.br/tutoriais/o-que-e-html-conceitos-basicos/>>. Acesso em 15 de Novembro de 2020.

MARQUES, B. *O Que é AJAX e Como Funciona?* 2019. Disponível em: <<https://www.hostinger.com.br/tutoriais/o-que-e-ajax/>>. Acesso em 20 de Julho de 2020.

MARTINS, J. E. M. P. Automation experiments in physics laboratories. *Phys. Educ.* 53 (2018) 055009, p. 3, 2018.

MORAIS, J. *Banco de Dados com Google Planilhas com ESP*. 2017. Disponível em: <<https://portal.vidadesilicio.com.br/banco-de-dados-com-google-planilhas-com-esp/>>. Acesso em 29 de Junho de 2020.

MOTA, A. *DHT11 e DHT22, Sensor de umidade e Temperatura com Arduino*. 2017. Disponível em: <<https://portal.vidadesilicio.com.br/dht11-dht22-sensor-de-umidade-e-temperatura>>. Acesso em 25 de Abril de 2020.

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 20 de Abril de 2020.

MOTA, A. *Sensor de luz com LDR*. 2017. Disponível em: <<https://portal.vidadesilicio.com.br/sensor-de-luz-com-ldr/>>. Acesso em 26 de Abril de 2020.

OFICINA, R. *O que é um Servidor web?* 2010. Disponível em: <https://www.oficinadanet.com.br/artigo/servidores/o_que_e_um_servidor_web>. Acesso em 08 de Maio de 2020.

OLIVEIRA, G. *NodeMCU – Uma plataforma com características singulares para o seu projeto IoT*. 2016. Disponível em: <<https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-uma-plataforma-com-caracteristicas-singulares-para-o-seu-projeto-iot/>>. Acesso em 21 de Abril de 2020.

PENIDO Édilus de C. C.; TRINDADE, R. S. Microcontroladores. *e-Tec Brasil*, p. 9,15,16, 2013.

PEREIRA, A. P. *O que é CSS?* 2009. Disponível em: <<https://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>>. Acesso em 20 de Novembro de 2020.

REDAÇÃO. *O que é um servidor web (web server)*. 2017. Disponível em: <<https://tudosobrehospedagemdesites.com.br/servidor-web/>>. Acesso em 08 de Maio de 2020.

RESEARCH, L. *Uma introdução à Internet da Coisas (IoT)*. 2013. Disponível em: <https://www.cisco.com/c/dam/global/pt_br/assets/brand/iot/iot/pdfs/lopez_research_an_introduction_to_iiot_102413_final_portuguese.pdf>. Acesso em 03 de Março de 2020.

ROGER, M. *Afinal o que é Internet das Coisas? Entenda de uma vez por todas*. 2017. Disponível em: <<https://www.canalmaxdicas.com.br/2017/06/o-que-e-internet-das-coisasentenda-de.html>>. Acesso em 13 de Março de 2020.

SILVA, A. M.; MUXITO, E. Agricultura inteligente – proposta de automação de pivôs e canais de irrigação com prototipação por arduino e webservice. In: . [S.l.]: III Congresso internacional Adventista de Tecnologia (CIAT), At Centro Universitário Adventista de São Paulo, 2018. p. 1,2.

SILVA, G. *O que é e como funciona a linguagem JavaScript?* 2015. Disponível em: <<https://canaltech.com.br/internet/O-que-e-e-como-funciona-a-linguagem-JavaScript/>>. Acesso em 02 de Novembro de 2020.

SOUZA, I. de. *Banco de dados: saiba o que é, os tipos e a importância para o site da sua empresa*. 2020. Disponível em: <<https://rockcontent.com/br/blog/banco-de-dados/>>. Acesso em 20 de Novembro de 2020.

TABLELESS. *O Que é HTML? Guia Básico Para Iniciantes*. 201–. Disponível em: <<https://tableless.github.io/iniciantes/manual/html/>>. Acesso em 15 de Novembro de 2020.

THOMSEN, A. *Monitorando Temperatura e Umidade com o sensor DHT11*. 2013. Disponível em: <<https://www.filipeflop.com/blog/monitorando-temperatura-e-umidade-com-o-sensor-dht11/>>. Acesso em 25 de Abril de 2020.

THOMSEN, A. *O que é Arduino?* 2014. Disponível em: <<https://www.filipeflop.com/blog/o-que-e-arduino/>>. Acesso em 20 de Abril de 2020.

VIEIRA, K. *O que é CSS e como ele funciona*. 2016. Disponível em: <<https://www.hostgator.com.br/blog/o-que-e-css/>>. Acesso em 20 de Novembro de 2020.