UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

SISTEMA DE AUTOMAÇÃO, MONITORAMENTO E CONTROLE DE HORTA PELA *WEB*

HENRIQUE LARONSO MACEDO CARDEAL

PICOS - PI
2016

HENRIQUE LARONSO MACEDO CARDEAL

SISTEMA DE AUTOMAÇÃO, MONITORAMENTO E CONTROLE DE HORTA PELA *WEB*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros (CSHNB) da Universidade Federal do Piauí (UFPI) como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação do Professor Ivenilton Alexandre de Souza Moura.

PICOS – PI
2016

FICHA CATALOGRÁFICA

Serviço de Processamento Técnico da Universidade Federal do Piauí

Biblioteca José Albano de Macêdo

C266s Cardeal, Henrique Laronso Macedo.

Sistemas de automação, monitoramento e controle de horta pela web / Henrique Laronso Macedo Cardeal . – 2016.

CD-ROM : il.; 4 ¾ pol. (52 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.

Orientador(A): Prof°. Ivenilton Alexandre de Souza Moura

1. Automação-Agricultura. 2. Sistema de Monitoramento-Web. 3. Arduino. I. Título.

CDD 005

SISTEMA DE AUTOMAÇÃO, MONITORAMENTO E CONTROLE DE HORTA PELA *WEB*

HENRIQUE LARONSO MACEDO CARDEAL

Monografia aprovada como exigência parcial para obtenção do grau d
Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 19 de fevereiro de 2016

Profª. Esp. Ivenilton Alexandre de Souza Moura
Orientador

Frank César Lopes Véras
Prof. Me. Frank César Lopes Véras
Membro

Thiago José Barbosa Lima
Prof. Esp. Thiago José Barbosa Lima
Membro

Dedico este trabalho a toda minha família, em especial aos meus pais, Ivonildo e Humberta, e meu irmão Lariel por serem, a baixo de Deus, a minha base, o meu motivo para continuar seguindo em frente.

AGRADECIMENTOS

Agradeço primeiramente à Deus, princípio e fim de todas as coisas, por tantas bênçãos, copiosamente, derramadas sobre mim durante o período da graduação e por, mesmo eu não sendo merecedor, jamais ter me abandonado.

Agradeço à meus pais e avós, pela força dada em momentos difíceis e por terem me proporcionado uma educação humilde, mas grandiosa em valores e princípios de dignidade e moral.

Agradeço a meus professores, pelo conhecimento passado e experiências vividas, em especial a meu orientador e amigo, Ivenilton Alexandre, pela disponibilidade e os inúmeros conselhos dados, relacionados ao curso e à vida.

Agradeço ao pessoal que aturou minhas manias, passou por momentos de lutas, companheirismo e alegrias dividindo moradia comigo, como o pessoal da deprensão, em especial Alex, Andressa, Edivania, Celma, Layza, Roniel, Taís, Rangel e Fátima.

Agradeço aos amigos, na verdade mais que isso, verdadeiros anjos que tive a oportunidade de conhecer e a honra de conviver durante o decorrer do curso, dentro e fora da universidade. Sou grato eternamente à Deus, por ter colocado em minha vida, pessoas tão singulares como Abimael, Bell, Gilberlon, Givanaldo, Kaio, Lívia, Viviane, Manú, Renan, Raí e outros. Foram muitos momentos difíceis, e só consegui superá-los graças ao apoio de vocês.

Agradeço à meu irmão Lariel e meu primo/irmão Sebastião, pela força dada em momentos de desânimo e compreensão na ausência em momentos importantes.

Enfim, quero agradecer à toda minha família e todos aqueles que fizeram parte dessa conquista. Desde pessoas que me deram carona entre Padre Marcos e Picos, à todos os meus irmãos na fé que intercederam por mim em suas orações.

"Debaixo do céu e sobre a terra, existe tempo certo para tudo e momento certo para cada coisa."
(Eclesiastes: 3,1)

"Que eu não perca a vontade de ter grandes amigos, mesmo sabendo que, com as voltas do mundo, eles acabam indo embora de nossas vidas."
(Ariano Suassuna)

"A amizade, cuja fonte é Deus, não se esgota nunca."
(Santa Catarina e Sena)

RESUMO

Este trabalho, tem por finalidade integrar tecnologias no desenvolvimento de um protótipo capaz de auxiliar em tarefas básicas do cultivo, promovendo maior controle e uso eficiente de recursos utilizados para tal. O projeto foi feito pelo uso da plataforma de prototipagem *Arduino*, seus sensores e atuadores, os sistemas *web* e a linguagem de programação *Ruby* no *Framework Rails*. O protótipo, depois de implementado, automatizou o processo irrigatório de maneira eficiente, proporcionando a coleta de dados sobre o cultivo, com disponibilidade para diversas culturas. Ele também possibilitou acesso aos dados coletados remotamente por meio da *internet*, permitindo também ao usuário, realizar interferências no processo, sem a necessidade da presença física.

Palavras chave: *Arduino*, Automação, Sistema *web*, Agricultura.

ABSTRACT

This work aims to integrate technology in the development of a prototype able to assist with basic tasks of cultivation, promoting greater control and efficient use of resources used to do so. The project was done by the use of Arduino prototyping platform, its sensors and actuators, web systems and the Ruby programming language on Rails Framework. The prototype, after implemented, automated the process irrigatório efficiently, providing data collection on the cultivation, with availability for different cultures. He also allowed access to the data collected remotely via the Internet, also allowing the user, perform interference in the process, without the need for physical presence.

Keywords: Arduino, Automation, Web System, Agriculture.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CSS	*Cascading Style Sheets*
DRY	*Don't Repeat Yourself*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICSP	*In-Circuit Serial Programming*
MVC	*Model-View-Controller*
PWM	*Pulse Width Modulation*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
USB	*Universal Serial Bus*
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de sistemas de informação computacionais vem ajudando tanto na prática quanto na segurança, rapidez e maior eficiência de tarefas as quais eram inteiramente manuais. Embora estes sistemas tenham tido um grande avanço nos últimos tempos, deixam a desejar em algumas áreas onde obtiveram pouco foco e, consequentemente, pouco desenvolvimento. Este é o caso da produção agrária na região do semiárido piauiense, um setor que vem ganhando novas melhorias com a evolução da engenharia agronômica e suas técnicas que auxiliam no plantio de várias culturas. Contudo, essa é uma área de enorme carência, no que diz respeito a sistemas de informação computacionais. A utilização destes sistemas tem potencial para maximizar tal produção, tendo em vista os dispositivos disponíveis no mercado podem ser utilizados nessa área, oferecendo um maior aproveitamento e controle da mesma.

A conservação da água, bem como o seu manejo inteligente, é algo de extrema importância, principalmente em regiões secas como o Nordeste do Brasil. A escassez de chuvas e o mau uso das fontes naturais e artificiais da região, causam, entre outros transtornos, baixas na produção agrícola local. Para suprir tais necessidades hídricas agrárias, a maior parte dos agricultores tem visto a solução na irrigação.

Segundo Ferreira (2011, p 2), “Irrigação é um método artificial pelo qual se calcula a quantidade de água aplicada na planta, com o objetivo de suprir as necessidades hídricas totais ou suplementares da planta na falta de chuva.”. Embora a agricultura irrigada seja associada a um elevado nível tecnológico, é notório que a irrigação no Brasil ainda seja praticada de forma inadequada, com grande desperdício de água. O que, além de aumentar os custos da produção, gera enormes custos ambientais (MANTOVANI *et al*., 2006). Muitas vezes, mesmo quando a irrigação é realizada em excesso, é comum que a planta seja exposta a situações de déficit hídrico, pois o solo não tem capacidade para armazenar água durante o intervalo de irrigação.

Alguns equipamentos eletrônicos já foram desenvolvidos para o meio agrícola, capazes da captação de informações e atuação no meio ambiente. A utilidade de tais componentes para a automação do processo irrigatório de maneira inteligente, já foi vislumbrada em trabalhos acadêmicos, como o Sistema Automatizado de Irrigação Para Culturas Específicas (ALTOÉ, 2012), mostrando o potencial da utilização da automação nesse processo.

Tendo em vista o potencial e a necessidade do desenvolvimento de um sistema que atenda ao controle básico na área agronômica, este trabalho discorre sobre a elaboração e construção de um protótipo que auxilia na automação, monitoramento e controle eficiente de

horta, fazendo assim dos sistemas de informação uma ferramenta útil também às ciências agrárias.

1.1 Objetivo

Este projeto tem por objetivo, desenvolver um sistema que auxilie na automação, bem como, no monitoramento agrometeorológico e controle da irrigação pela *web*.

1.1.1 Objetivos específicos

- Automatizar o processo irrigatório de maneira eficiente;
- Captar e armazenar dados durante a execução de um cultivo, a fim de gerar informações sobre o mesmo;
- Possibilitar o monitoramento e a intervenção em tempo real, pelo usuário na execução do cultivo.

1.2 Organização do trabalho

O presente trabalho está organizado em 5 capítulos.

O segundo capítulo disserta sobre a fundamentação teórica utilizada para embasar o desenvolvimento do projeto, proporcionando uma melhor compreensão das tecnologias utilizadas.

O terceiro capítulo aborda sobre o desenvolvimento da aplicação, desde a escolha das tecnologias até a sua implementação.

O quarto capítulo descreve sobre os testes realizados durante e após o desenvolvimento do projeto.

A conclusão, dá-se no quinto e último capítulo, apresentando uma síntese sobre as considerações finais e indicações de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata dos conceitos básicos de tecnologias necessárias para entendimento do desenvolvimento do projeto.

2.1 Automação

Considera-se automação qualquer sistema de equipamentos eletrônicos e/ou mecânicos que controlam seu próprio funcionamento, quase sem a intervenção do homem (CASTRUCCI; BOTTURA, 2006). Assim possibilita fazer um trabalho por meio de máquinas controladas automaticamente, sendo capazes de se regularem sozinhas. Diferenciando-se da mecanização, que consiste na simples utilização de máquinas para auxiliar a realização de um trabalho, substituindo um esforço físico humano.

A automação de serviços em busca da economia de esforços físicos, data da pré-história com invenções como a roda. Daí por diante, o homem prosseguiu buscando e implementando ideias que viessem a facilitar serviços, antes inteiramente manuais, tornando-os mais práticos e eficientes, dando-lhe um maior conforto na execução dos mesmos. Pode-se observar uma exemplificação da evolução da execução de uma tarefa, desde o processo manual, passando pela mecanização até culminar na automação na Figura 1.

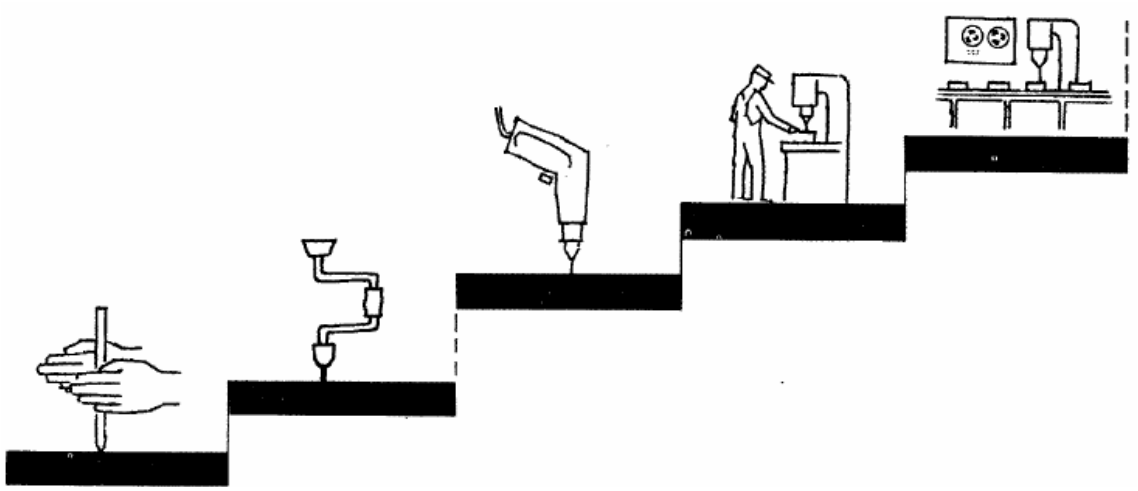

Figura 1: Evolução da automação ao longo dos tempos.
Fonte: PINTO (2005).

O grande destaque à automação, veio com a revolução industrial em meados do século XVIII, quando o sistema artesanal tornou-se industrial. Máquinas como o tear mecânico

tornaram um trabalho fatídico e demorado, em algo rápido e preciso. Com a contínua evolução da revolução industrial, as fábricas procuraram se equipar com máquinas que reproduzissem determinadas tarefas automaticamente (ROBÓTICA, 2009).

Em 1954, surgiu o primeiro robô automático, projetado por George Devol. Com base nesse projeto a empresa *Unimation Inc*. desenvolveu o *Unimate*. Ele começou a ser usado na linha de produção da General Motors em 1962, pegando pedaços quentes de metal e colando nos chassis dos carros. A partir do *Unimate* uma série de pesquisas visando a criação de novos robôs para a indústria surgiu.

Pinto (2005), afirma que a própria invenção do computador é relacionada à necessidade de automatizar cálculos, podendo ser observada desde a utilização de ábacos pelos babilônicos, passando pelas necessidades militares, chegando à grande gama de áreas de utilização atual.

Uma das grandes dificuldades em desenvolver protótipos na área da automação, é a montagem da complexa infraestrutura necessária. O desenvolvimento da infraestrutura demanda muito tempo, material específico e um conhecimento técnico avançado. Uma maneira de lidar com esse problema é a utilização de plataformas de desenvolvimento. Essas plataformas permitem uma prototipagem prática e rápida, possibilitando uma grande facilidade no seu manuseio (FONSECA; VEGA, 2011).

2.1.1 Plataforma de Prototipagem Eletrônica *Arduino*

As plataformas de desenvolvimento baseadas em microcontroladores podem ser utilizadas em várias áreas de conhecimento. Um microcontrolador, pode ser definido como a união de um microprocessador, de sistemas de temporização, de aquisição e comunicação em um mesmo circuito integrado (FONSECA; VEGA, 2011; *ARDUINO*, 2014). Um exemplo deste tipo de plataforma amplamente utilizado, é o *Arduino.*

Desenvolvido na Itália em 2005, o conceito *Arduino* consiste, por definição, em uma plataforma de computação física e lógica, onde circuitos digitais ligados a sensores e atuadores são capazes de captar situações no ambiente, realizar o processamento e responder com interferências no ambiente por meio dos atuadores (*ARDUINO*, 2014).

2.1.1.1 Sensores e atuadores

Conforme Wendling (2010), sensores são dispositivos sensíveis à alguma forma de energia do ambiente, relacionando informações sobre uma grandeza física que precisa ser mensurada, como: temperatura, pressão, corrente, entre outros. Os sensores se dividem em dois tipos, analógicos e digitais. Onde:

- Sensores analógicos: Assumem qualquer valor no seu sinal de saída, desde que dentro da sua faixa de operação. Baseando-se em sinais analógicos que, mesmo limitados entre dois valores de tensão, podem assumir infinitos valores de tensão intermediários. Isso significa que, teoricamente, para cada nível da condição medida, haverá um nível de tensão correspondente;
- Sensores Digitais: Podem assumir apenas dois valores no seu sinal de saída, sendo estes valores, 0 e 1. Baseando-se em níveis de tensão bem definidos, podendo ser descritos como alto (*high*) ou baixo (*low*), ou simplesmente 1 e 0. Esses sensores utilizam lógica binária, que é a base do funcionamento dos sistemas digitais.

Já os atuadores, são dispositivos que modificam uma variável controlada no ambiente. Eles recebem um sinal proveniente de um controlador e agem sobre o sistema controlado. Pode-se citar como exemplo relés, motores, válvulas, solenoides, entre outros (FELIZARDO; BRACARENSE, 2013).

No mercado atual, existe uma gama considerável de sensores e atuadores desenvolvidos para serem controlados por meio de plataformas de prototipagem, tais como *Arduino* e afins.

A grande vantagem da utilização do *Arduino* como plataforma de prototipagem, é a facilidade no manuseio do mesmo. O *Arduino* propicia ao desenvolvedor, por meio do seu ambiente de programação, a oportunidade de controlar uma grande variedade de dispositivos físicos. Ele conta com um ambiente de programação que trabalha com *Processing*, uma linguagem que se assemelha bastante ao C, e é multiplataforma, podendo ser usado no *Windows, Linux* ou *Mac*. Tudo isso permite que o usuário faça alterações físicas e lógicas no projeto, antes, durante e depois da sua implementação com muita praticidade.

Utilizar o *Arduino* é algo bem simples, tendo em vista, que não é necessário um conhecimento aprofundado em eletrônica, necessita-se apenas de um conhecimento básico em programação para desenvolver os primeiros projetos. Sua programação consiste em duas funções principais: a função *void setup()*, que só é executada no momento da inicialização da placa, onde normalmente se inicializa uma comunicação serial, ou mesmo, a definição de portas

de entrada e saída; já a função *void loop(),* é uma função que ficará executando um ciclo infinito de instruções, é nela que o tratamento de ações costuma ser feito.

Várias versões do *Arduino* surgiram, cada uma com aspectos voltados para diferentes perfis de desenvolvedores. O *Arduino* Uno por exemplo, é equipado com 14 pinos de entrada/saída digital, que vão do pino D0 ao pino D13 (dos quais 6 podem ser usados como saídas PWM - *Pulse Width Modulation*), 6 entradas analógicas, do pino A0 ao pino A5, um cristal oscilador de 16MHz, uma conexão USB (*Universal Serial Bus)*, uma entrada de alimentação, uma conexão ICSP (*In-Circuit Serial Programming*) e um botão de *reset*, como mostra a Figura 2.

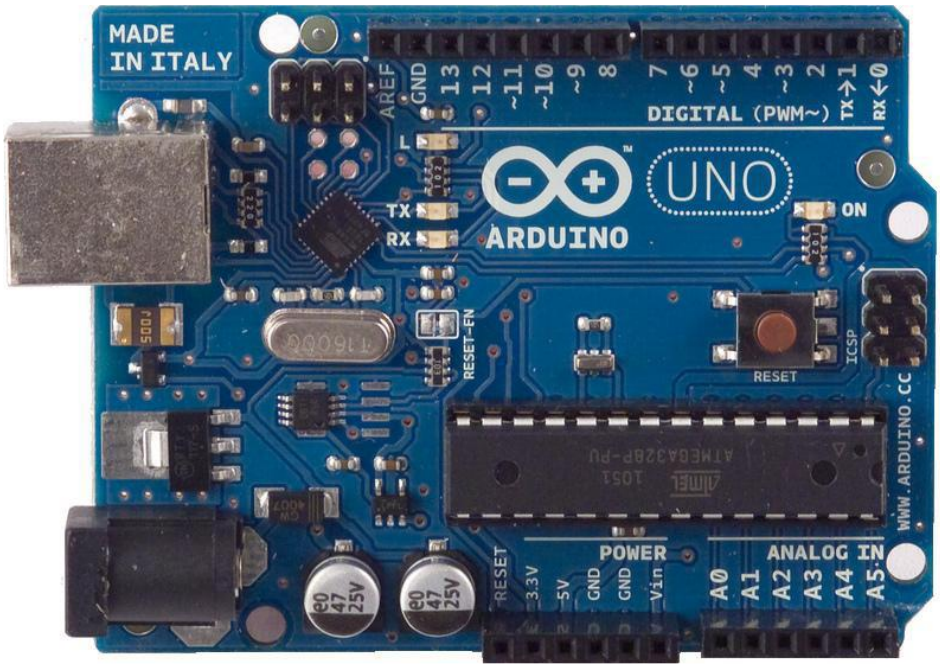

Figura 2. Placa *Arduino Uno*.
Fonte: *ARDUINO*, 2014.

Por ser uma plataforma de baixo custo e baseada no conceito de *hardware* e *software* livre, o *Arduino* tornou-se um grande aliado no avanço do desenvolvimento de sistemas de informação autônomos. Depois do seu lançamento inúmeros projetos surgiram, desde pequenos robôs a casas controladas a distância por sistemas *web*.

2.2 Sistemas *web*

Sistemas de informação computacionais, são sistemas compostos por *hardware*, *software*, banco de dados, pessoas e procedimentos configurados para receber, manipular,

armazenar e processar dados em informações (STAIR; REYNOLDS, 2006). Os sistemas de informação ainda se dividem em sistemas para *desktop* e *web.*

Os sistemas *web*, surgiram na década de 1990 com o objetivo de formar um repositório do conhecimento humano (BERNERS-LEE *et al.*, 1994). Baseando-se em mecanismos de armazenamento, recuperação e visualização de documentos eletrônicos.

A tecnologia *web* tem seu funcionamento relativamente simples, uma aplicação desse tipo é projetada para ser executada em um servidor *web* utilizando o protocolo HTTP (*Hypertext Transfer Protocol*), para realizar o envio de requisições entre cliente e servidor (onde são executadas as tarefas). Estes sistemas são apresentados através de um navegador *web,* podendo assim, serem acessados a qualquer hora, de qualquer dispositivo computacional com recursos para esse tipo de acesso e em qualquer lugar onde se tenha uma conexão com a *internet*.

Com a ascensão da *internet* nos últimos anos, os sistemas *web* têm tido um enorme avanço, no que se diz respeito a popularidade e pluralidade de áreas, nas quais surgiram sistemas *web* específicos. Várias linguagens de programação voltadas para a área surgiram, proporcionando uma grande evolução ao incorporar novos recursos e novas funções a este tipo de sistema. Dentre estas linguagens, uma que vem ganhando grande destaque é o *Ruby*.

Na Figura 3 pode-se perceber o crescente aumento no número de programadores que utilizam o *Ruby* como linguagem de programação em seus projetos, demonstrado no *ranking* de linguagens de programação mais usadas no mundo, feito pelo site www.tiobe.com.

Jan 2016	Jan 2015	Change	Programming Language	Ratings	Change
1	2	↑	Java	21.465%	+5.94%
2	1	↓	C	16.036%	-0.67%
3	4	↑	C++	6.914%	+0.21%
4	5	↑	C#	4.707%	-0.34%
5	8	↑	Python	3.854%	+1.24%
6	6		PHP	2.706%	-1.08%
7	16	⇈	Visual Basic .NET	2.582%	+1.51%
8	7	↓	JavaScript	2.565%	-0.71%
9	14	⇈	Assembly language	2.095%	+0.92%
10	15	⇈	Ruby	2.047%	+0.92%
11	9	↓	Perl	1.841%	-0.42%
12	20	⇈	Delphi/Object Pascal	1.786%	+0.95%
13	17	⇈	Visual Basic	1.684%	+0.61%
14	25	⇈	Swift	1.363%	+0.62%
15	11	⇊	MATLAB	1.228%	-0.16%
16	30	⇈	Pascal	1.194%	+0.52%
17	82	⇈	Groovy	1.182%	+1.07%
18	3	⇊	Objective-C	1.074%	-5.88%
19	18	↓	R	1.054%	+0.01%
20	10	⇊	PL/SQL	1.016%	-1.00%

Figura 3: *Ranking* de linguagens de programação mais usadas. Janeiro - 2016.
Fonte: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.HTML.

2.2.1 A linguagem *Ruby*

Criada em 1995 no Japão por Yukihiro Matsumoto, a linguagem *Ruby* foi projetada tendo por inspiração, outras linguagens, como *Smalltak*, *Python*, *Lisp* e *Perl* (FLANAGAN; MATSUMOTO, 2008). Essa linguagem é dotada de uma grande simplicidade de sintaxe, é multiplataforma e multiparadigma, estando dentro do contexto de 4 paradigmas (Funcional, Imperativo, Reflexivo e Orientado a Objeto) que podem ser utilizados, pois possui interpretadores para vários sistemas operacionais, podendo ser executada nos mesmos.

Por ser uma linguagem interpretada, e não compilada, tem a necessidade da instalação prévia de um interpretador. O *Ruby* possui ainda, um gerenciador de pacotes e dependências: *Ruby*gems. Os *gems,* podem ser tidos como bibliotecas reutilizáveis de código *Ruby*. Portanto, para a utilização dos vários *gems* disponíveis, além do interpretador *Ruby*, é necessário também a instalação do *Rubygems*.

Segundo Caelum (2013), dentre as principais características do *Ruby*, destacam-se o fato da mesma ser puramente orientada a objeto, com uma tipagem forte e dinâmica, como também a sua expressividade, possibilitando uma grande facilidade na implementação e leitura de códigos. Um dos fatores que ajudou bastante a alavancar a utilização da linguagem de programação *Ruby* em aplicações *web*, foi a criação do *framework web Ruby on Rails.*

2.2.2 O *Framework Ruby on Rails*

Criado em 2003 por David Heinemeier Hansson na empresa *37signals*, *Rails* é um *framework web* que segue o padrão MVC (*Model-View-Controller*). Fuentes (2012) topifica este padrão da seguinte forma:

- Modelo: carregam duas responsabilidades: os dados, que são armazenados em um ou mais banco de dados, e as regras de negócio.
- Controle: é a camada de mediação entre o sistema e a *web.*
- Apresentação: meio pelo qual o sistema mostrará o resultado do processamento.

O *Rails* segue duas filosofias: convenções de configurações e DRY (*dont repeat yourself* – não se repita). Essas duas filosofias vêm para ajudar na diminuição do trabalho dos desenvolvedores. A primeira evita que o programador escreva códigos desnecessários. A segunda estimula o programador a fazer a reutilização de códigos os quais ele mesmo já escreveu.

Fuentes (2012) nos afirma que "O *Rails* é preparado para criar aplicações modernas e arrojadas". Além de arrojadas, o *Rails* proporciona uma grande agilidade no desenvolvimento

de aplicações, pois as mesmas são bem organizadas no modelo MVC, o que também facilita a sua manutenção.

Segundo Hartl (2012) *Ruby on Rails* faz uso na sua camada de apresentação de linguagens que a maioria dos navegadores compreende e que se tornaram padrões para o desenvolvimento *web*. São estas:

- HTML (*Hiper Text Markup Language*- Linguagem de Marcação de Hipertexto): É uma linguagem de marcação de texto, definindo o seu formato e significado. Trabalha também com hipertextos, que são ligações entre as páginas da *internet*, permitindo navegar entre elas;
- CSS (*Cascading Style Sheets*): Adiciona estilo à estrutura, formatando de maneira simples os elementos definidos no HTML;
- *JavaScript*: É uma linguagem de programação usada para desenvolver pequenos programas que realizam ações em uma página *web* e até mesmo em servidores desenvolvidos com *Node.js.*

2.2.3 *Node.js*

Com o passar dos anos, o *JavaScript* tornou-se um verdadeiro padrão para o desenvolvimento do lado cliente em aplicações *web*. Em contra peso, o âmbito do servidor, em sua maior parte, pertencia a linguagens como PHP e Java. Dois grandes obstáculos impediam o desenvolvimento do *JavaScript* no espaço do servidor. O primeiro foi a sua reputação, pois o mesmo era tido como linguagem de brincadeira, que convinha apenas para amadores. O segundo foi o seu precário desenvolvimento em comparação com as demais linguagens utilizadas em servidores *web* (IHRIG, 2014).

A grande alavancada do *JavaScript* deu-se com a ascensão da *internet*. Por se tratar da única linguagem suportada por todos os grandes navegadores, ganhou grande atenção de empresas como o *Google* e *Apple*, levando a imensas melhorias no seu desempenho.

Criado por Ryan Dahl em 2009, como uma estrutura primariamente usada para criar servidores altamente escalonáveis para aplicações *web*, o *Node.js* é escrito em C++ e em *JavaScript*. Visando impulsioná-lo, seu criador o conectou ao motor v8 de *JavaScript* do *Google*, presente no *Google Chrome*. Trata-se de uma plataforma cujo objetivo é a fácil construção de rápidas e escaláveis aplicações de rede. Para tal emprega um modelo baseado em eventos, e *non-blocking I/O* (JUNIOR, 2013).

Uma das partes fundamentais da arquitetura *Node.js* é o *Event Loop*. O *Event Loop* é o método que o *JavaScript* utiliza para lidar com os eventos de uma melhor maneira. Ele permite que o *framework*, ao invés do sistema operacional, gerencie a mudança entre as tarefas a serem executadas. Na prática ele é um *loop* infinito que a cada iteração verifica em sua fila de eventos se um determinado evento foi emitido, isso graças aos eventos do *JavaScript*.

Segundo Texeira (2013), o *JavaScript* contribui muito para o sucesso do *Node.js*. Dentre os motivos para o uso do *Node.js*, pode-se citar a facilidade de uso, já que o mesmo é de fácil instalação e configuração, o próprio fato de usar em sua programação, uma linguagem tão difundida no meio *web*, como o *JavaScript*, e o fato de existirem vários *frameworks* disponíveis para trabalhar com *Node*, tais como *Express JS*, *SAILS JS*, *TOTAL JS e LOCOMOTIVE JS.* Merecendo, este último *framework*, um destaque por proporcionar algumas características, como arquitetura MVC, encaminhamento de *Helpers*, convenção sobre configurações e conexão com qualquer banco de dados.

2.3 Banco de Dados

Os sistemas de informação computacionais, sejam eles para *desktop* ou *web*, têm por função básica a captação de dados e o seu tratamento para a geração de informações. Na maioria dos sistemas, existe a necessidade de armazenar os dados em uma base, e que a base gerencie estes dados. Para tal tarefa, surgiram os SGBD's (Sistemas Gerenciadores de Banco de Dados). O primeiro SGBD que se tem notícia, surgiu no final do ano de 1960 com base nos primitivos sistemas de arquivos disponíveis na época, os quais não controlavam o acesso concorrente por vários usuários ou processos (TAKAI *et al*, 2005).

Com o passar do tempo e a evolução dos sistemas de informação computacionais, os SGBD's também evoluíram. Atualmente, os seguintes modelos de dados são normalmente utilizados pelos SGBD's: modelo hierárquico, modelo em redes, modelo relacional, o modelo orientado a objetos e, há ainda, o modelo Objeto-Relacional.

> A área de atuação dos sistemas Objeto-Relacional tenta suprir a dificuldade dos sistemas relacionais convencionais, que é o de representar e manipular dados complexos, visando ser mais representativo em semântica e construção de modelagens. A solução proposta é a adição de facilidades para manusear tais dados utilizando-se das facilidades SQL (*Structured Query Language*) existentes. Para isso, foi necessário adicionar: extensões dos tipos básicos no contexto SQL; representações para objetos complexos no contexto SQL; herança no contexto SQL e sistema para produção de regras. (TAKAI *et al*, 2005, p 9)

Pode citar-se como exemplo de SGBD Objeto Relacional, o *PostgreSQL*.

2.3.1 *PostgreSQL*

Derivado do pacote *Postgres* desenvolvido na Universidade da Califórnia, o *PostgreSQL* é atualmente o mais avançado banco de dados de código aberto. Em 1994 Andrew Yu e Jolly Chen adicionaram um interpretador da linguagem SQL ao *POSTGRES* e puseram o nome de *Postgres*95. Em 1996, ficou claro que o nome "*Postgres*95" não resistiria ao teste do tempo, então foi escolhido o nome de *PostgreSQL*, para refletir o relacionamento entre o banco de dados em que ele foi baseado e as versões mais recentes com capacidade SQL. A sua ênfase também foi alterada, pois durante o desenvolvimento do seu predecessor, o foco era na identificação e compreensão de problemas existentes no servidor. Já o *PostgreSQL*, tem ênfase para o aumento das funcionalidades e recursos, mas ainda assim, o trabalho continuou em todas as áreas (*POSTGRE*SQL, 2005).

O *PostgreSQL* traz consigo várias funcionalidades modernas, tais como: comandos complexos, chaves estrangeiras, integridade transacional, entre outros. O mesmo, pode ser utilizado praticamente com qualquer linguagem de programação moderna. Por ainda ser compatível com várias plataformas, pode ser utilizado em projetos nos principais sistemas operacionais da atualidade.

O *PostgreSQL* possui, entre seus principais recursos, o *psql*. Ele é um cliente no modo terminal, que permite que o usuário entre com comandos para serem executados pelo *PostgreSQL*.

Por estas e outras características, é possível perceber que o *PostgreSQL* é um SGBD capaz de dar suporte a aplicações complexas e entender porque o mesmo é tão difundido atualmente.

2.4 Metodologia de desenvolvimento

Seja qual for a linguagem de programação que se trabalhe, ou o SGBD escolhido para armazenar os dados da aplicação, algo que não se pode deixar de observar na criação de aplicações, é a metodologia de desenvolvimento. Ter a organização que uma metodologia lhe proporciona, pode agilizar bastante o processo de desenvolvimento.

A produção de um *software* passa por várias etapas até que se chegue ao produto final. Pressman (2011) diz que as metodologias prescritivas foram propostas para trazer ordem a área de desenvolvimento de *software*, buscando sua estruturação, prescrevendo um conjunto de atividades e um fluxo para realizá-las. Como exemplo deste tipo de metodologia, temos o modelo cascata, espiral e evolucionário:

- Cascata: Propõe uma abordagem sequencial e sistemática, desde o levantamento de requisitos, até o suporte contínuo do *software* pronto (PRESSMAN, 2011);
- Espiral: O desenvolvimento acontece em meio a uma série de iterações, passando pelo planejamento, ativação, análise de riscos, avaliação e desenvolvimento. A cada nova iteração repete-se todas essas etapas (PÁDUA, 2000);
- Evolucionário: É interativo, apresentando características que dão suporte ao desenvolvimento de versões mais completas (PRESSMAN, 2011).

Um outro tipo de metodologia que chama bastante atenção, por focar mais no *software* do que na sua concepção e documentação, é a metodologia ágil. Segundo Sommerville (2011) as metodologias ágeis são adequadas no desenvolvimento de sistemas onde seus requisitos mudam rapidamente durante o desenvolvimento. Pode-se citar como exemplo de metodologia ágil o XP (*Extreme Programming*), Desenvolvimento de *software* adaptativo e *Scrum*:

- XP: "Os programadores trabalham em pares e desenvolvem testes para cada tarefa antes de escreverem o código." (SOMMERVILLE, 2011, p 71);
- Desenvolvimento de *software* adaptativo: Tem foco na colaboração humana e na auto-organização das equipes (SOMMERVILLE, 2011);
- *Scrum:* "É um *framework* Ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos." (SABBAGH, 2013, p 65). Tem por base o empirismo, utilizando uma abordagem interativa e incremental.

3 SISTEMA DE AUTOMAÇÃO, MONITORAMENTO E CONTROLE DE HORTA PELA *WEB*

Por perceber a atual ascensão dos sistemas *web* no mundo, inclusive àqueles que integram *web* e automação, e por notar o pouco foco que esse desenvolvimento dá à agronomia na região do semiárido piauiense, sentiu-se a necessidade de desenvolver uma aplicação *web*, voltada para tal área.

Ao analisar a ideia concebida e as tecnologias disponíveis para a sua realização, decidiu-se por desenvolver um protótipo de um sistema *web*, implementado com o *framework Ruby on Rails* com integração com a plataforma de prototipagem eletrônica *Arduino*. Visando que o mesmo, seja capaz de atender ao controle básico da área agronômica focando em três aspectos:

- Automação: Automatizar o processo irrigatório, de maneira que o usuário defina os níveis mínimos de umidade do solo para a cultura trabalhada. Assim fazendo o uso inteligente e eficiente da água nesse processo;
- Monitoramento: Permitir que o usuário tenha acesso à informações atuais, tais como, temperatura e umidade local da sua horta, podendo também interferir na mesma, a qualquer hora e em qualquer lugar por meio do acesso remoto ao sistema, possibilitado pela *internet*;
- Controle eficiente: Dar a opção do usuário entrar com determinados dados sobre o seu cultivo, para futura geração de informações que venham a auxiliá-lo no controle e otimização do processo produtivo.

Isso de maneira que seja um protótipo de um sistema simples e desenvolvido com periféricos de baixo custo, com o intuito de auxiliar tanto um pequeno agricultor que deseje automatizar o seu cultivo e obter um controle eficiente, quanto um pesquisador da área agronômica que, além de automatizar o cultivo e ter um controle eficiente, deseje obter um conjunto de informações diversas da execução do cultivo, desde o seu início ao seu fim, como também monitorar o cultivo a qualquer momento e de qualquer lugar por meio de um dispositivo conectado à *internet*.

Para o desenvolvimento objetivo do sistema, faz-se necessário um bom detalhamento das funções e necessidades que o mesmo deve ter e resolver. Algumas técnicas de engenharia, ajudam nesse processo, tais como o levantamento de requisitos do sistema e a diagramação do mesmo.

3.1 Requisitos do sistema

Os requisitos do sistema são uma forma de expressar as necessidades que ele deve resolver e funções que ele deve ter, para que possa ser construído de maneira que corresponda à sua função final. Estes requisitos ainda se dividem em funcionais (RF), não funcionais (NF) e regras de negócio (RN). Os requisitos percebidos para a elaboração do sistema, com base na ideia do autor, são demonstrados a seguir.

O Quadro 1, mostra os requisitos funcionais que o protótipo deve ter, para que atenda aos objetivos propostos pela ideia.

Quadro 1: Requisitos funcionais

Identificador	**Descrição**	**Depende de**
RF01	O sistema deverá pedir uma identificação do usuário.	
RF02	O sistema deverá possuir um cadastro de culturas.	
RF03	O sistema deverá possuir um cadastro de cultivos, vinculados à uma cultura.	RF02
RF04	O sistema deverá automatizar a captação de dados e a irrigação de uma horta.	RF03
RF05	O sistema deverá disponibilizar as informações captadas no decorrer da execução de um cultivo.	RF02, RF03
RF06	O sistema permitirá a verificação em tempo real das informações da horta.	RF02

O Quadro 2, mostra os requisitos não funcionais que o protótipo deve possuir, para que alcance os objetivos propostos pela ideia.

Quadro 2: Requisitos não funcionais.

Identificador	**Descrição**	**Categoria**	**Depende de**
RNF01	O sistema deve possuir *interface* intuitiva que permita ao usuário utilizar a plataforma e realizar suas atividades sem esforços.	Usabilidade	
RNF02	O sistema deve estar disponível pela *Internet*, e capaz de ser acessado por meio dos principais navegadores disponíveis no mercado.	Portabilidade	
RNF03	O sistema deve ser capaz de se comunicar com o *Arduino* para poder fazer as leituras da horta e irriga-la.	Integração	
RNF04	A persistência das informações deve ser implementada em um Sistema Gerenciador de Bancos de Dados Relacionais (SGBDR).	Manutenibilidade	

Definidos os requisitos não funcionais, o Quadro 3 mostra as regras de negócio que devem ser seguidas na execução do sistema.

Quadro 3: Regras de negócio.

Identificador	**Descrição**	**Depende de**
RN01	Usuários não identificados não podem realizar acesso ao sistema.	
RN02	Um cultivo deve, obrigatoriamente ser vinculado à uma cultura.	
RN03	Só pode haver a automação de um cultivo por vez.	

3.2 Principais diagramas do sistema

Após a especificação e análise de requisitos, as informações colhidas podem ser utilizadas para a diagramação do sistema. Logo a seguir, serão mostrados os principais diagramas do protótipo em questão.

3.2.1 Diagrama de casos de uso

Por meio do diagrama de casos de uso, apresentado na Figura 4, é possível se ter um melhor entendimento das atividades que o sistema deve realizar. No diagrama é mostrada a interação do usuário com o sistema, que se dá pela simples identificação do usuário. Daí por diante o usuário pode executar todas as ações, desde que respeite suas dependências, como por exemplo, o início de um cultivo, depende do prévio cadastro de uma cultura, já que o cultivo deve ter uma cultura vinculada à ele.

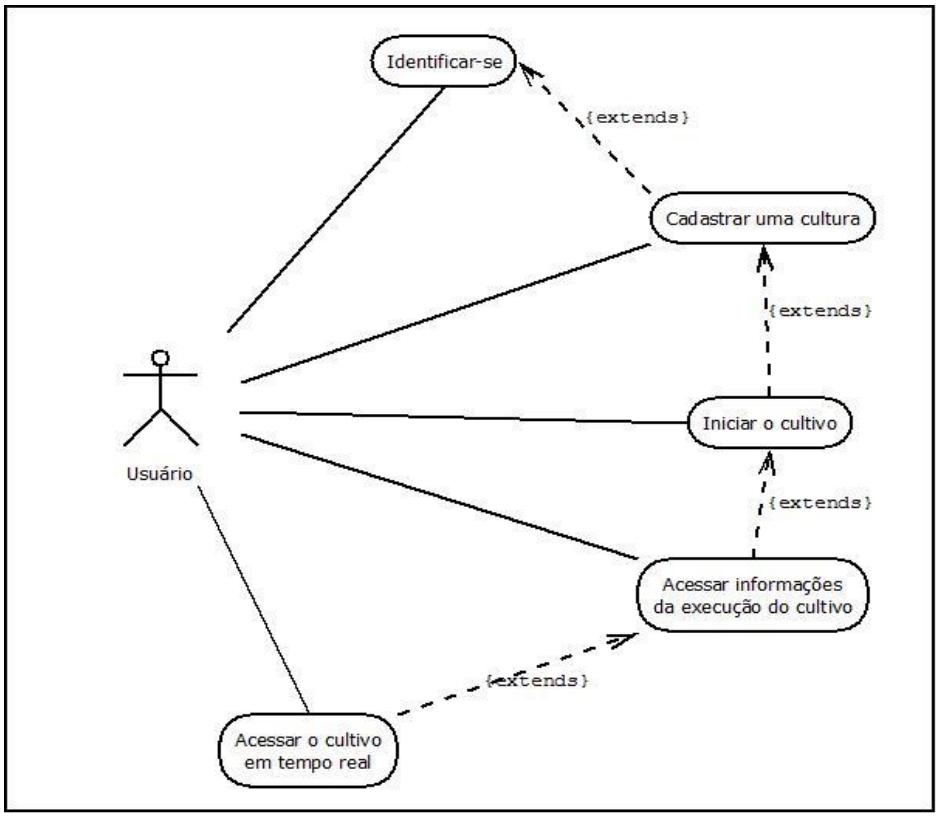

Figura 4: Diagrama de casos de uso.
Fonte: Produzido pelo autor.

3.2.2 Diagrama de classes

No diagrama de classes demonstrado na Figura 5, é apresentada toda a estrutura de classes do sistema, como elas se relacionam e as definições dos métodos e atributos de cada uma. Nele, as classes são representadas pelos retângulos que levam seu nome na parte superior, seus atributos na parte central e seus métodos na parte inferior.

Neste diagrama, pode ser observado que a classe “Cultura”, será responsável por manter os dados referentes às culturas inseridos pelo usuário. A mesma se relaciona com a classe “Cultivo”, pois um cultivo deve ser vinculado à uma cultura. Por esse mesmo motivo, a classe “Cultivo” também se relaciona com a classe “Dados_cultivo”, já que os dados mantidos por essa classe pertencem à um cultivo.

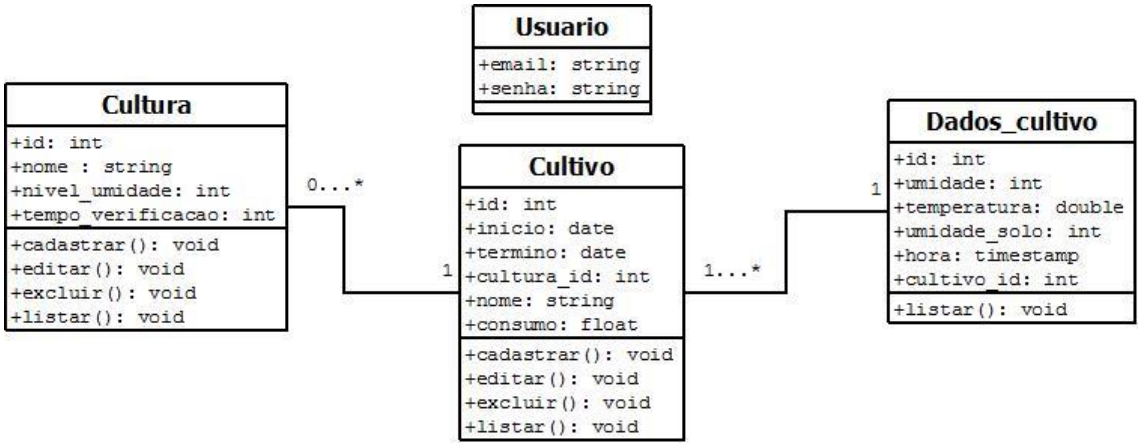

Figura 5: Diagrama de classes.
Fonte: Produzida pelo autor.

Pode-se perceber ainda, que a classe "usuário" se apresenta sem associações, já que a mesma serve apenas para identificação do usuário.

Com o sistema melhor compreendido e detalhado pelas tabelas de requisitos do sistema e os diagramas de casos de uso e de classes, pôde-se vislumbrar com maior atenção, as ferramentas e tecnologias que seriam necessárias para o seu desenvolvimento.

3.3 Método

Para realização da atividade planejada, foram necessários estudos e pesquisas bibliográficas acerca das tecnologias fundamentais para o desenvolvimento do protótipo, que culminaram na escolha das seguintes tecnologias:

3.3.1 DHT11

O sensor DHT11, foi escolhido a fim de captar a temperatura e a umidade ambiente durante o funcionamento do protótipo. Ele é um sensor de saída digital. O mesmo, possui uma interface física de 3 pinos, sendo eles, o pino +5V, o pino GND e o pino DATA, ilustrados na Figura 6.

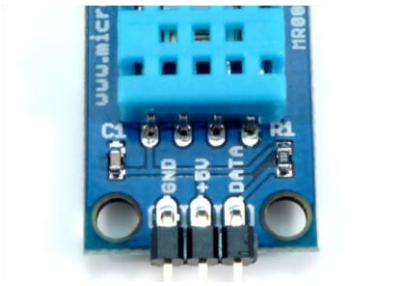

Figura 6: Sensor DHT11.
Fonte: http://www.*Arduino*eletronica.com.br/2014/03/tutoriais-sensores-xv-umidade.HTML#.Vp590CorLIU. Acessado em 19 de Jan de 2016.

Os dois primeiros pinos são, respectivamente, aterramento e alimentação. Já o terceiro pino, é o de envio e recebimento de dados.

Ele conta com a biblioteca *DHT.h*, que possui algumas funções nativas, tais como *DHT.humidity* e *DHT.temperature*, como também é capaz de fazer uma leitura rápida e precisa dos dados no ambiente, por meio de uma porta analógica.

3.3.2 Sensor de umidade do solo Higrômetro

Por se mostrar eficiente e possuir um custo acessível, o sensor de umidade do solo Higrômetro, foi escolhido com o intuito de fazer a leitura da umidade do solo durante o funcionamento do protótipo. Ele possui dois tipos de saída em sua interface: A0 e D0, onde A0 é o canal analógico, no qual podemos obter a sua precisão, pois recebemos valores reais, em números de 0 a 1023 e o canal D0, que é o canal digital, trabalhando somente com 0 ou 1 (ligado ou desligado). O mesmo possui ainda mais dois pinos de alimentação, sendo esses o +VCC e o GND, como podemos observar na Figura 7.

O sensor, retorna um inteiro que varia de acordo com a resistência à condução de uma corrente elétrica entre dois pontos, sendo este retorno o parâmetro para o cálculo da umidade do solo.

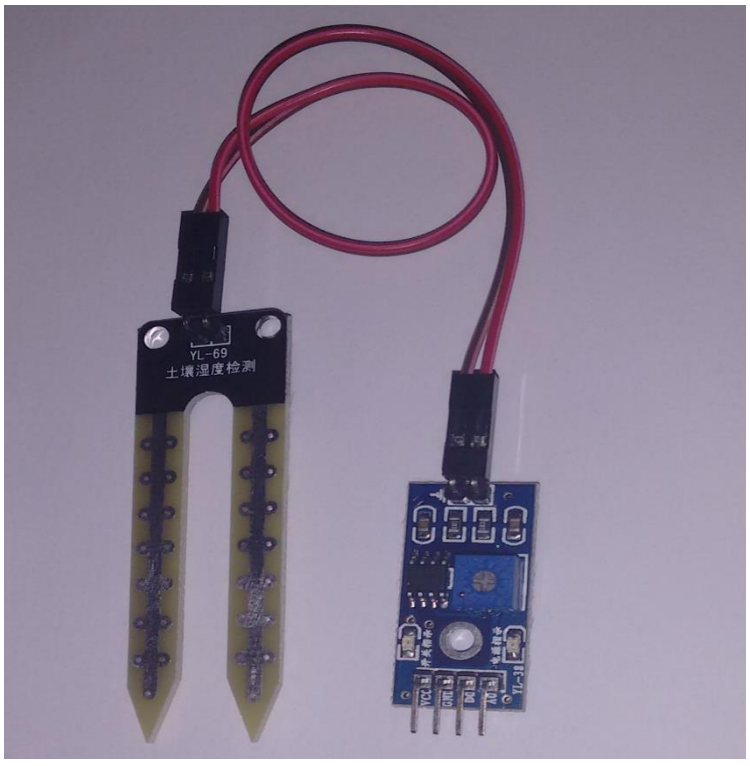

Figura 7: Sensor de umidade do solo Higrômetro.
Fonte: Produzida pelo autor.

3.3.3 Sensor de fluxo de fluido ½”

Por haver a necessidade de se mensurar o consumo de água durante as irrigações de um cultivo, o sensor de fluxo de fluido, foi selecionado para executar tal tarefa durante o funcionamento do protótipo, gerando uma informação de relevância significativa no gerenciamento do cultivo. Ele é um sensor de efeito *HALL*, possui um conector com três pinos, como podemos ver na Figura 8, sendo estes, o vermelho a alimentação, o preto o aterramento e o amarelo saída PWM. Conta com dois orifícios, um de saída e outro de entrada de ½” de diâmetro e atua com uma alimentação de 5V.

Figura 8: Sensor de Fluxo de fluido.
Fonte: http://www.filipeflop.com/pd-206c5b-sensor-de-fluxo-de-agua-1-2-yf-s201.html.

Em sua estrutura há um válvula em formato de cata-vento com um imã acoplado que trabalha em conjunto com um sensor *hall* para enviar o sinal PWM. Através destes pulsos é possível mensurar a vazão de água.

3.3.4 Válvula solenoide ½”

A Válvula solenoide ½”, foi escolhida para executar a ação de abrir e fechar o fluxo de água no processo de irrigação. A mesma é um atuador composto por um corpo de válvula acoplado a um solenoide.

No corpo da válvula existem dois orifícios, o de entrada e o de saída. Esses orifícios permitem a passagem de um fluído ou não quando a haste é acionada pela força da bobina. Esta força faz com que o pino seja puxado para o centro da bobina, permitindo a passagem do fluído. O processo de fechamento, ocorre graças a perda de energia, pois o pino exerce uma força graças ao seu peso e da mola que tem instalada. Os estados de desenergizada e energizada podem, respectivamente ser observado na Figura 9.

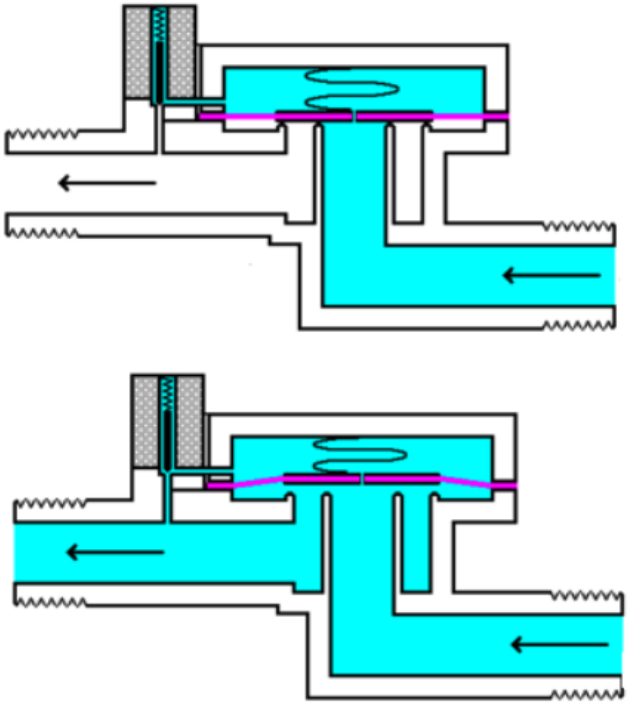

Figura 9: Estados da válvula solenoide.

Fonte: http://acquaticos.blogspot.com.br/2010/10/valvula-solenoide.html. Acessado em 19 Jan de 2016.

3.3.5 *Arduino Uno*

Por ser uma plataforma de prototipagem de baixo custo, possuir integração com várias outras tecnologias, ser capaz de controlar vários periféricos, e ter uma interface simples de manusear, o *Arduino Uno* foi escolhido para controlar os sensores e atuadores, e enviar os dados percebidos no ambiente para a aplicação.

3.3.6 O *Framework Ruby on Rails*

A escolha do *framework Ruby on Rails* para o desenvolvimento da aplicação *web*, deu-se pelo fato do mesmo utilizar uma linguagem de tipagem forte e dinâmica, como o *Ruby*, seguir o modelo MVC, facilitando bastante a manutenção, como também organiza de maneira significativa o código, e a filosofia DRY, que possibilita e incentiva a reutilização de código, agilizando bastante no processo de desenvolvimento. Ele ainda possibilita integração com a plataforma *Node.js*, o *framework Bootstrap* e a linguagem de programação *JavaScript*.

3.3.7 *Node.js*

Pela necessidade de uma rotina que se repetisse dentro de um determinado intervalo, optou-se pela utilização de um servidor *Node.js* para realizar a tarefa da comunicação com o *Arduino*, ficando então responsável pela tarefa de automatizar o ciclo de verificação de dados do ambiente no qual o protótipo fosse instalado, como também, a verificação instantânea, já que o *Node.js* é orientado a evento.

3.3.8 *PostgreSQL*

Por ser compatível com quase todas as linguagens de programação, inclusive *Ruby* e *JavaScript*, o *PostgreSQL* foi escolhido como banco de dados da aplicação, ficando incumbido de armazenar dados inseridos pelo usuário, ou captados pelo protótipo.

3.3.9 Metodologia de desenvolvimento

A metodologia de desenvolvimento escolhida, foi o *framework Scrum*, por ser ágil e simples, sendo adaptado ao projeto.

3.4 Implementação do sistema *web*

O sistema *web* do protótipo foi desenvolvido no *Framework Ruby on Rails* e tem como servidores locais o *webric*, que é nativo do *framework*, trabalhando em conjunto com um servidor *Node.js*. Conta com um repositório de dados composto por quatro tabelas. São estas, a tabela "*users"*, "culturas", "cultivos" e "dados_cultivos", como demonstra a Figura 10. Onde:

- Tabela "*users"*: Serve para guardar as informações de identificação do usuário. Tais como *email* e senha;
- Tabela "culturas": É responsável por guardar informações referentes a cada cultura. É ela quem armazena informações como o nome da cultura, o nível de umidade mínimo que o solo deve permanecer para o desenvolvimento ideal da cultura e o intervalo entre as verificações automáticas;
- Tabela "cultivos": Armazena informações do cultivo em atividade, tais como a data de início e de término, o nome do cultivo, a cultura a qual esteja vinculado e o consumo de água gasto pela irrigação até então;

- Tabela “dados_cultivos”: Armazena os dados captados automaticamente pelos sensores entre o intervalo definido na tabela cultura, como a temperatura ambiente, a umidade ambiente e do solo, a hora em que foi feita a verificação e o cultivo em vigor.

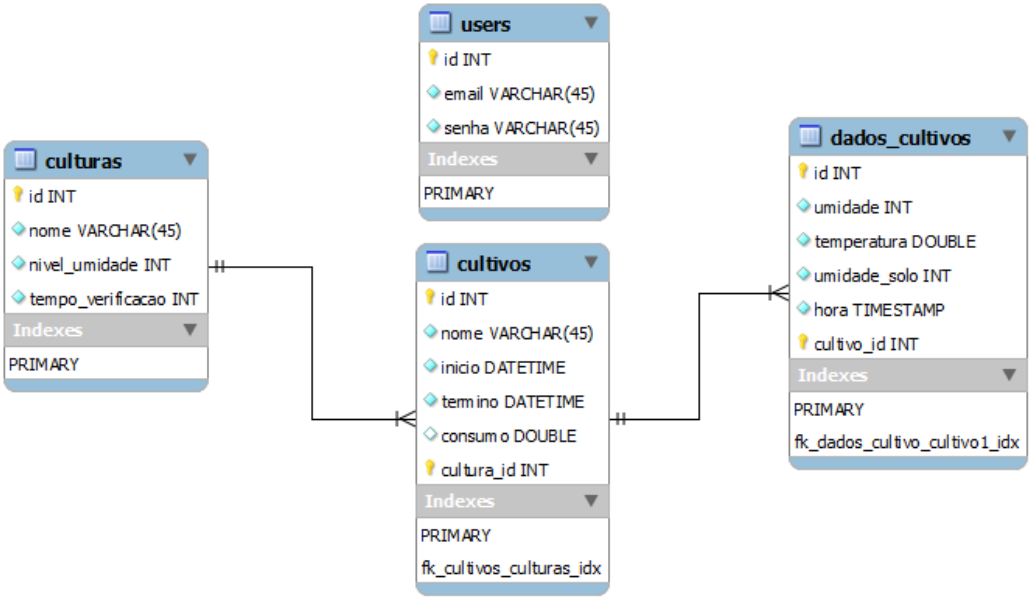

Figura 10: Modelagem do banco de dados.
Fonte: Produzida pelo autor.

Com o intuito de possibilitar a execução eficiente de um plantio, focado na cultura cultivada, a primeira ação que o sistema executa para iniciar um cultivo, é solicitar que o usuário cadastre uma cultura, definindo em seus parâmetros, entre outros, o nível de umidade mínimo e o intervalo em minutos entre as verificações da referida cultura, como é mostrado na Figura 11. Esses dados são salvos na tabela “culturas”.

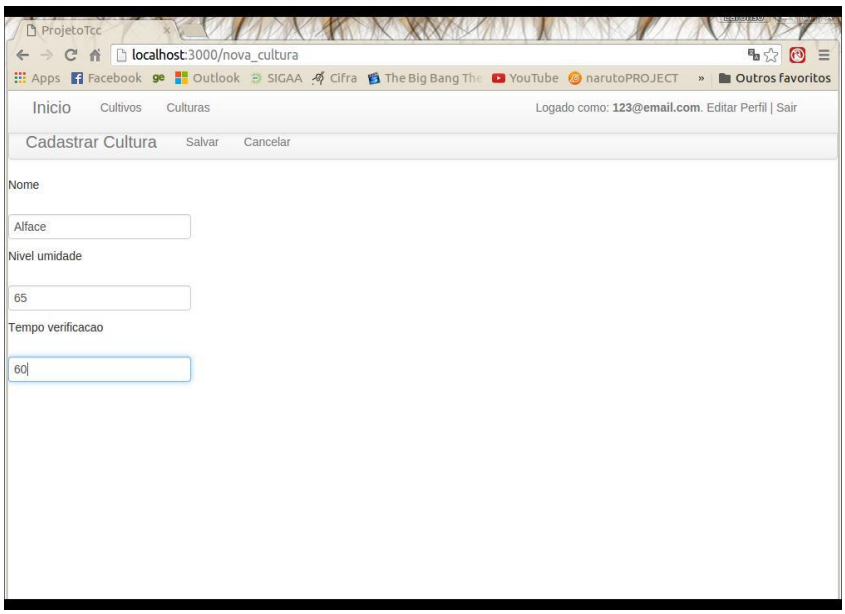

Figura 11: Tela de cadastro de cultura.
Fonte: Produzida pelo autor.

Posteriormente é solicitado o cadastro de um cultivo, demonstrado na Figura 12. Nesse, são inseridas informações como a data de início e de término e a escolha da cultura a ser trabalhada no cultivo em questão, e o *controller* responsável faz a inserção desses dados na tabela “cultivos”.

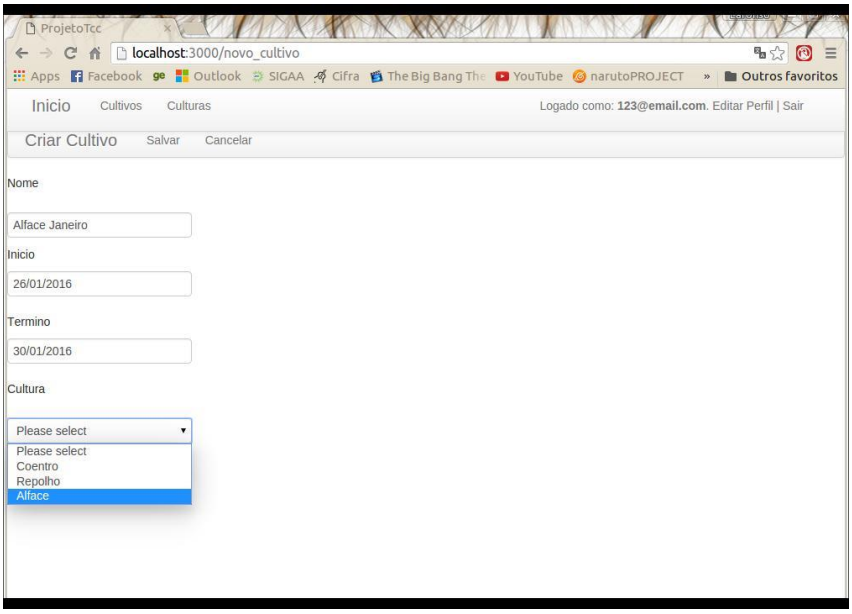

Figura 12: Tela de cadastro de cultivo.
Fonte: Produzida pelo autor.

Uma vez que o cultivo é criado, inicia-se a automação da horta. Esta automação dá-se com um servidor *Node.js* que é executado em segundo plano e, por meio da função *setInterval()*, faz as verificações das informações a serem salvas na tabela “dados_cultivo” após o tempo de intervalo definido na cultura vinculada ao cultivo. Ainda durante essa rotina de verificações, compara-se o valor da umidade do solo com o nível de umidade mínima da cultura em questão. Caso o valor da umidade do solo seja menor que o mínimo aceitável, é enviado pelo servidor um comando para que o sistema físico irrigue o solo até que se atinja o nível ideal, logo em seguida é verificada a quantidade de água gasta na irrigação e esta quantidade é salva no atributo “consumo” da tabela “cultivo”.

O conjunto de informações geradas a partir dos dados captados pelas verificações automáticas, pode ser visto e até impresso, ao se selecionar um cultivo na página de listagem dos cultivos. Além das verificações automatizadas, é possível também monitorar e interferir na horta em tempo real. Pois, por meio de um *Socket*, ao se clicar em um botão da página *web* de consulta, é enviado um comando ao servidor *Node.js*, que envia um comando de ação/consulta ao sistema físico do protótipo e retorna a informação para a página *web*. Assim é possível que se faça um monitoramento dos dados da horta em tempo real, como também interferir nela enviando um comando momentâneo de irrigação.

Toda a comunicação com o sistema físico do protótipo, dá-se pelo servidor *Node.js* de maneira bidirecional, por meio da biblioteca *SerialPort*. Esta biblioteca, se conecta à porta serial utilizada, a uma taxa de transferência de 9600b/s. Possibilitando que o servidor possa escrever caracteres na porta serial e que o *Arduino* leia esses caracteres. E é basicamente assim que o servidor se comunica com o sistema físico. Ele envia um caractere específico, de acordo com a ação que se queira executar, o *Arduino* lê esse caractere, executa a ação referente à leitura e dá uma resposta ao servidor também por meio de uma escrita na porta serial. Os caracteres utilizados na comunicação, estão exemplificados no Quadro 4, juntamente com suas respectivas ações e respostas.

Quadro 4: comandos, ações e respostas da comunicação *Node.js*/*Arduino.*

Caractere escrito pelo servidor	**Ação no *Arduino***	**Resposta escrita na porta serial**
“A”	Leitura da umidade do solo.	Umidade do solo.
“E”	Leitura da umidade ambiente.	Umidade ambiente.
“T”	Leitura da temperatura ambiente.	Temperatura ambiente.
“I”	Irriga por 10segundos, leitura da umidade do solo e leitura do consumo de água	Umidade do solo.
“L”		Quantidade de água gasta na irrigação.

3.5 Implementação do código *Arduino*

Para o código embarcado no *Arduino* deste projeto, houve a necessidade da inclusão da biblioteca “*dht.h”*, a fim de possibilitar a comunicação com o sensor DHT11. Depois disso, foram declaradas algumas variáveis necessárias ao código, tais como as que recebem os pinos que são utilizados pra leitura de sensores, ou ativação de atuadores, e essas variáveis são inicializadas na função *void setup()*. Os pinos definidos foram:

- O pino digital 2: Por ser um dos dois pinos no *Arduino* Uno capaz de ler pulsos, ficou responsável por fazer a leitura do sensor de fluxo de fluido para calcular a quantidade de água gasta a cada irrigação;
- Pino analógico A0: responsável por receber os dados de leitura do sensor de humidade do solo Higrômetro;

- O pino analógico A1: responsável por receber os dados de leitura do sensor DHT11;
- O pino digital 13: responsável pela ativação da válvula solenoide, abrindo ou fechando o fluxo de água;

Ainda dentro da função *void setup()*, é feita a chamada da comunicação serial da seguinte forma: *Serial.begin(9600);*. Passa-se por parâmetro a definição da taxa de bits da comunicação com a porta serial em 9600b/s, sendo esta taxa, igual a definida no servidor responsável pela comunicação com o *Arduino*.

Na função *void loop()*, que executa um loop infinito, fica a parte da interação com o servidor *Node.js*. Por meio de um desvio condicional é verificado se algo foi escrito na porta serial. Caso algo tenha sido escrito, ela verifica se é um comando válido por meio de outro desvio condicional e, caso seja, executa a ação. Os comandos e suas respectivas ações foram demonstrados no tópico anterior no Quadro 4. O código *Arduino* para a leitura do sensor de umidade é semelhante ao exemplificado na Figura 13:

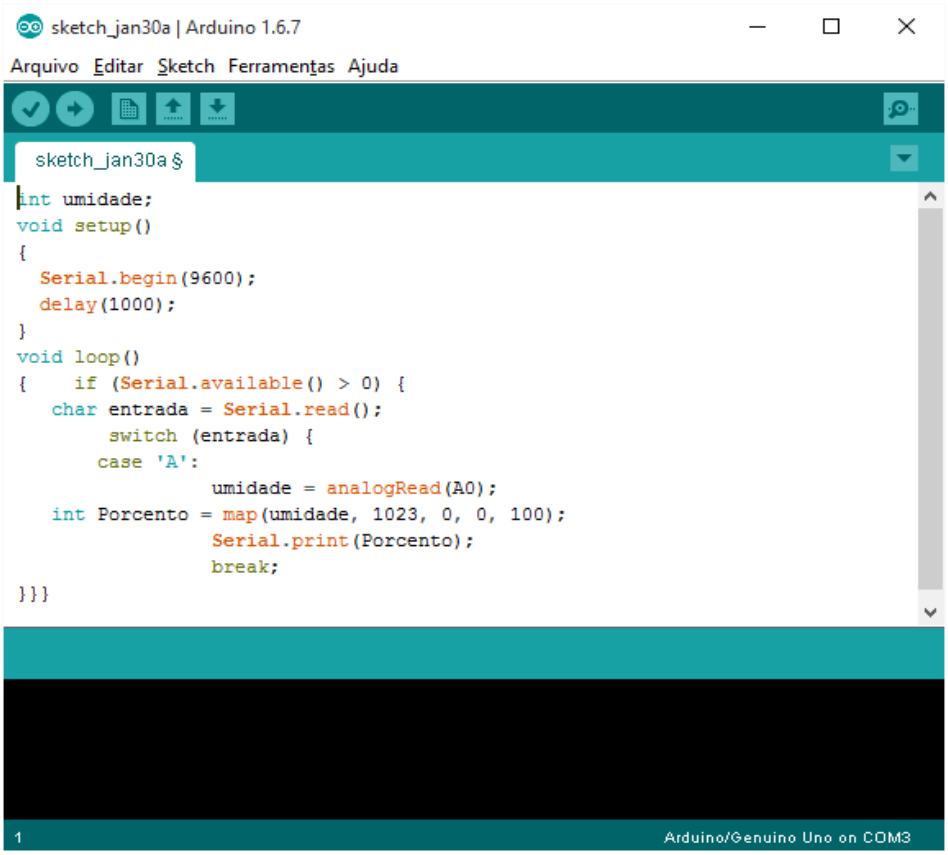

Figura 13: Código de leitura de umidade do solo.
Fonte: Produzida pelo autor.

Com este código embarcado no *Arduino*, basta conectar o sensor de umidade, sendo o pino de saída analógico do sensor ligado na porta analógica A0 do *Arduino* e a devida alimentação, que já será possível fazer a leitura da umidade do solo.

No tópico seguinte discorre-se sobre a montagem do protótipo físico no *Arduino*, bem como o ambiente em que o mesmo foi instalado.

3.6 Implementação do sistema físico

Inicialmente foi construído um balcão, semelhante ao modelo de balcão utilizado por cultivadores de algumas culturas de hortaliças, onde o protótipo físico foi implementado. O balcão mencionado pode ser visto na Figura 14.

Figura 14: Balcão de cultivo.
Fonte: Produzida pelo autor.

A válvula solenoide e o sensor de fluxo de água, foram acoplados entre a tubulação e o cano com furos para irrigação, de maneira que sempre que a válvula solenoide abrir o fluxo de água para irrigação, o sensor de fluxo possa fazer a leitura para cálculo do consumo de água gasto. A implementação destes dispositivos pode ser vista na Figura 15.

Figura 15: Disposição da válvula solenoide e do sensor de fluxo.
Fonte: Produzida pelo autor.

Ainda na Figura 15, Pode ser observado que os cabos que conectam os dispositivos ao *Arduino* foram prolongados, a fim de manter o *Arduino* e o servidor que executa aplicação *web*, à uma determinada distancia da horta para evitar a exposição a fatores que coloquem sua integridade física em risco, a mesma ação foi tomada para os demais dispositivos.

Na Figura 16, é apresentada a implantação do sensor de umidade do solo, que foi introduzido no solo, para que possa fazer as medições dos níveis de umidade do mesmo.

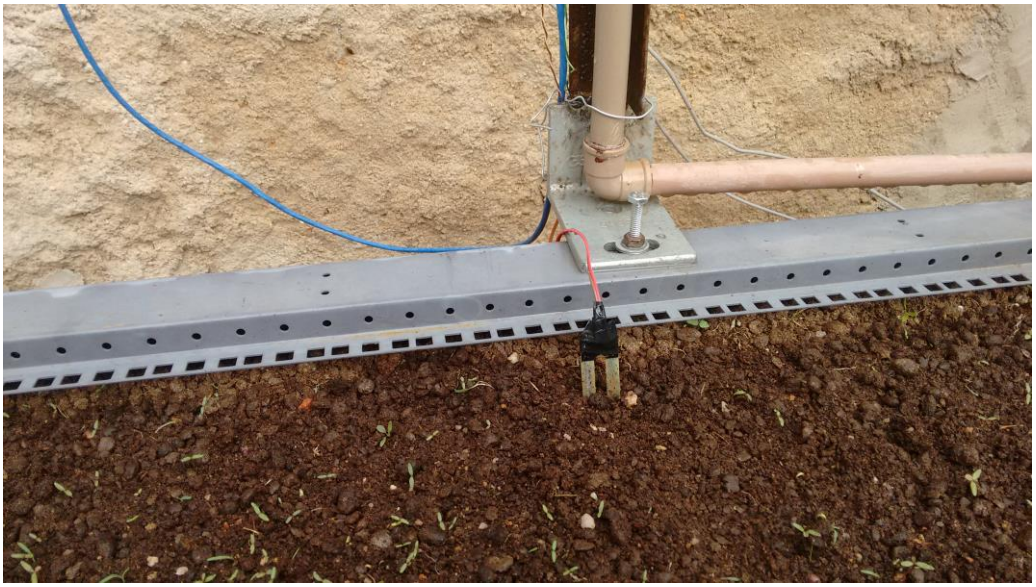

Figura 16: Implantação do sensor de umidade do solo.
Fonte: Produzida pelo autor.

O sensor de umidade e temperatura ambiente (DHT11), foi instalado acima do dispositivo de irrigação, a fim de que permaneça próximo à horta para fazer as leituras dos dados, mas que não seja molhado pela aspersão de água, de forma que tenha sua integridade física preservada para poder medir as variáveis do ambiente. Esta instalação pode ser observada na Figura 17.

Figura 17: Implantação do sensor de umidade e temperatura (DHT11).
Fonte: Produzida pelo autor.

A comunicação da placa com a aplicação *web*, bem como sua alimentação, dá-se por meio de um cabo USB que conecta o *Arduino* ao servidor que hospeda a aplicação.

A montagem do circuito do *Arduino* pode ser vista na Figura 18, nela um esquema de montagem mostra a conexão dos sensores ao *Arduino*, é realizada com o auxílio de uma *protoboard*, onde todos os sensores têm sua alimentação ligada ao pino de 5V e o GND do *Arduino* e fazem sua comunicação por meio dos pinos A0, A1 e D2. Sendo, respectivamente, pinos comunicação dos sensores de umidade do solo, DHT11(de umidade e temperatura ambiente) e sensor de fluxo de água. Já a ligação da válvula solenoide, responsável pela abertura e fechamento do fluxo de água para a irrigação, ocorre por meio de um módulo relé interligado à uma fonte externa, pois a válvula funciona em uma voltagem maior do que o *Arduino* pode fornecer. O disparo deste módulo relé, acontece por meio da porta D13.

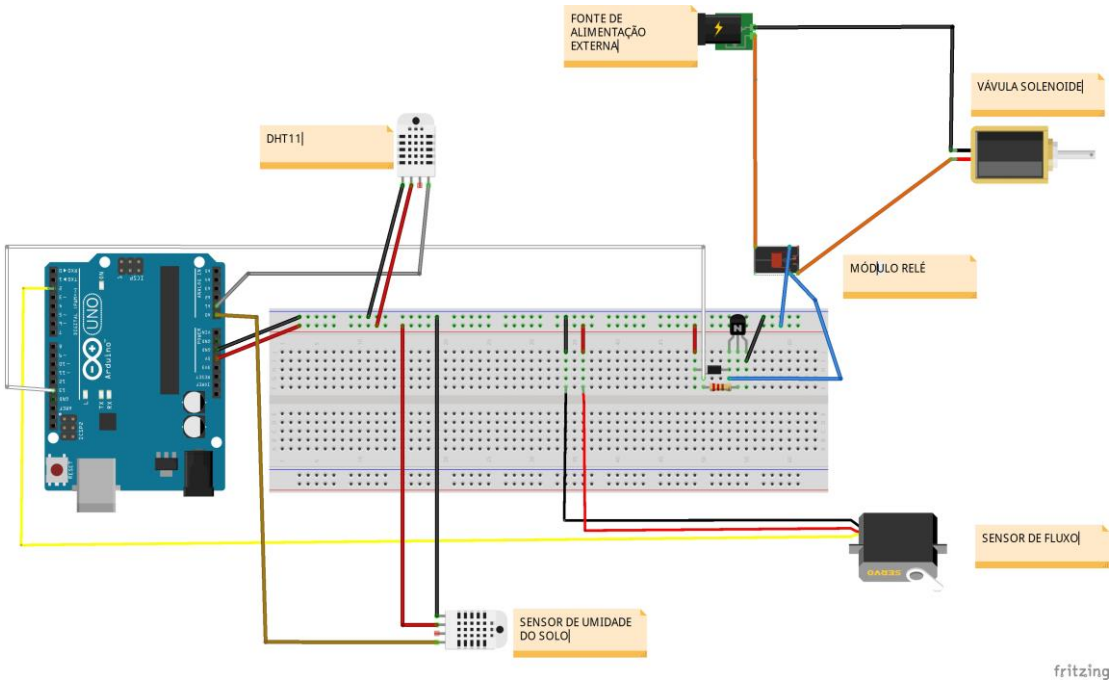

Figura 18: Circuito do *Arduino*.
Fonte: Produzida pelo autor.

4 TESTES

Este capítulo apresenta os principais testes realizados durante e depois do desenvolvimento do protótipo.

Os primeiros testes, foram realizados ainda no início do desenvolvimento do protótipo, com os testes individuais dos sensores e atuadores posteriormente utilizados na implementação do projeto.

4.1 Testes dos sensores e atuadores

Os testes dos sensores e atuadores utilizados no projeto, foram realizados de maneira isolada com o *Arduino* e, em sua maioria, utilizando códigos prontos disponibilizados pelos fabricantes dos dispositivos. Isso foi feito, com o intuito de observar o comportamento dos mesmos.

Durante a realização dos testes com os dispositivos isolados, todos se comportaram como o esperado, exceto o sensor de fluxo de água. Por meio de testes com uma mangueira e um balde de água, percebeu-se que os valores medidos pelo sensor não condiziam com os reais.

Após serem feitas algumas alterações no valor do fator de multiplicação de pulsos utilizado no código, seguidas de testes, ficou observado que o valor atribuído para o fator, que mais se aproximava de um resultado exato era de 4.5, e não 5.5 como constava no código disponibilizado pelo fabricante.

Posteriormente aos testes com os dispositivos isolados, realizou-se alguns testes, já com todos os periféricos conectados ao *Arduino* e embarcando nele o código desenvolvido para ser utilizado na aplicação, testando por meio do monitor serial. Este teste, ocorreu sem problemas, pois todos os dispositivos se comportaram como o esperado.

4.2 Testes de comunicação

Durante a fase de desenvolvimento da aplicação, alguns testes de comunicação entre o servidor *Node.js* (responsável pela comunicação com o protótipo físico) e o *Arduino*, foram executados. Nesta execução, algumas inconformidades foram percebidas no modelo de comunicação bidirecional. De maneira que, ao ser enviado um comando pelo servidor, o *Arduino* recebia e reconhecia este comando, executava a ação correspondente, respondia para o servidor, porém o servidor recebia valores incorretos.

Após algumas revisões e alterações na sintaxe de algumas funções no servidor, foram realizados novos testes de comunicação. E esses, demonstraram sucesso em sua execução.

4.3 Testando o sistema completo

Terminado o desenvolvimento da aplicação *web* e a implantação do sistema físico, decidiu-se testar o sistema por completo, realizando o cultivo da cultura do coentro. Sendo essa cultura, cadastrada com um nível de umidade do solo mínimo de 55% e intervalo de verificação de 120 minutos, como pode ser observado na Figura 19.

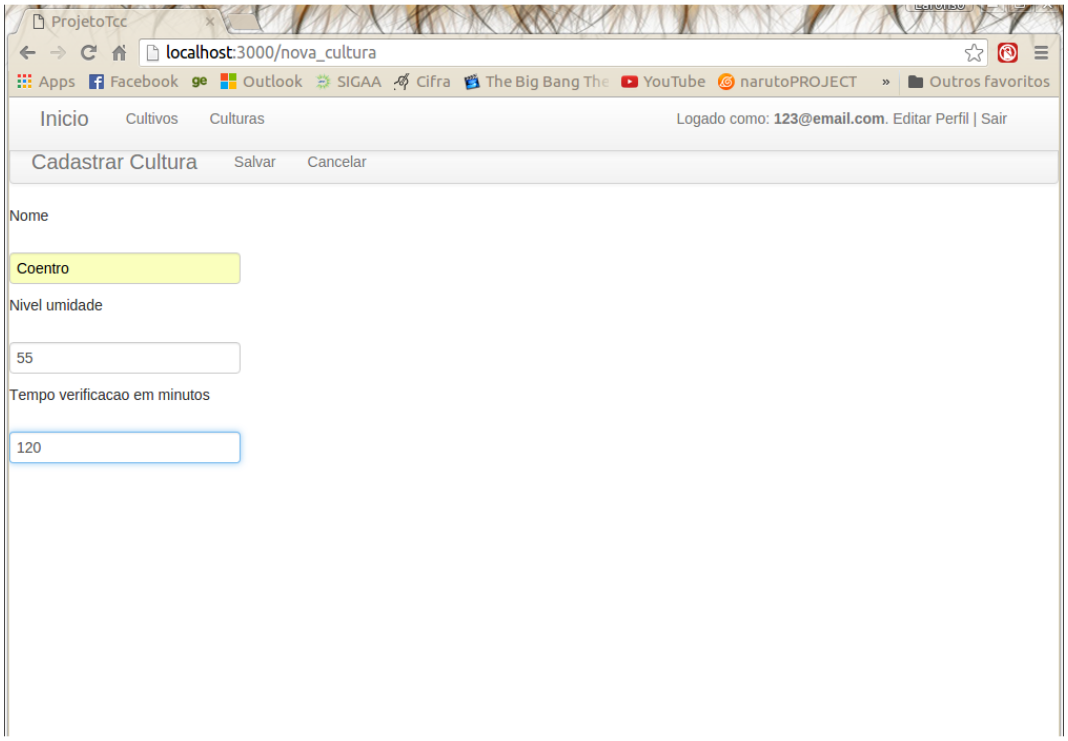

Figura 19: Cadastro de cultura.
Fonte: Produzida pelo autor.

O cultivo criado para o teste, teve seu início em 02/02/2016 e término datado para 29/02/2016, como pode ser observado na Figura 20.

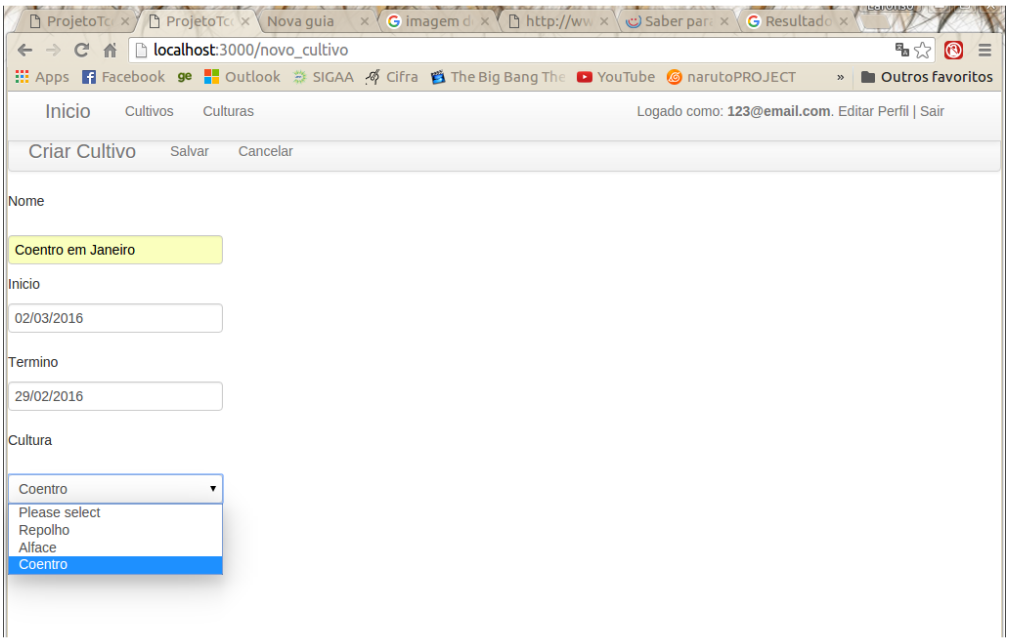

Figura 20: Inicio do cultivo.
Fonte: Produzida pelo autor.

Durante o teste do sistema, foram obtidos resultados satisfatórios. Já que o sistema teve um comportamento que atendeu a seus objetivos. Como pode ser observado na Figura 21, que mostra as informações captadas na horta durante a execução do cultivo, houve sucesso na geração de informações advindas do ciclo de verificação de dados durante o intervalo definido, como também na automação das irrigações, tendo em vista que o nível de humidade do solo se manteve sempre acima do nível mínimo estabelecido na cultura.

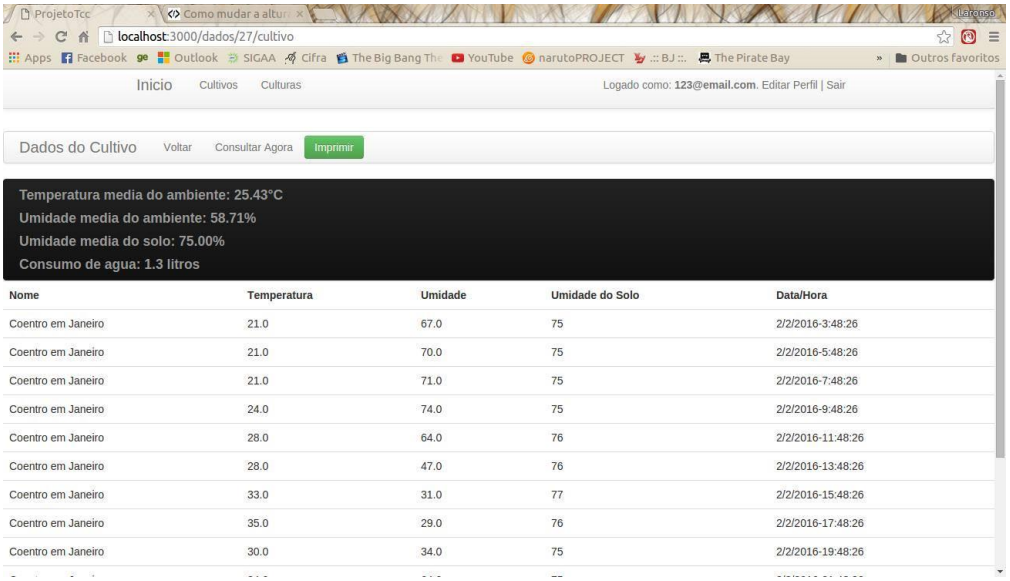

Figura 21: Informações sobre a execução do cultivo.
Fonte: Produzida pelo autor.

O sistema também se mostrou eficaz, no que se diz respeito ao monitoramento e controle em tempo real da horta. A Figura 22, mostra uma consulta feita em tempo real de um determinado dado da horta e a Figura 23, mostra uma irrigação disparada, também em tempo real, pelo usuário.

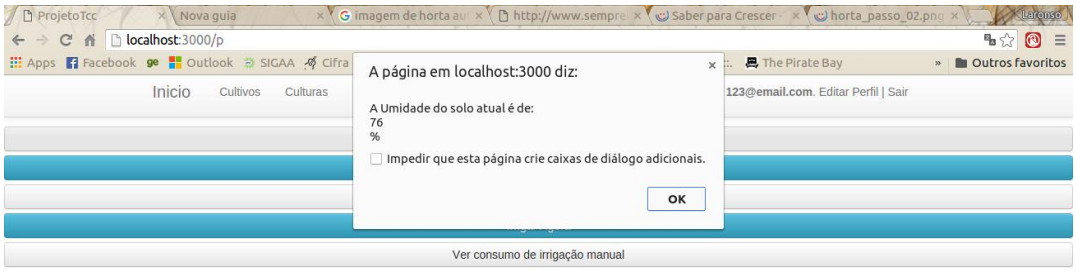

Figura 22: Consulta em tempo real.
Fonte: Produzida pelo autor.

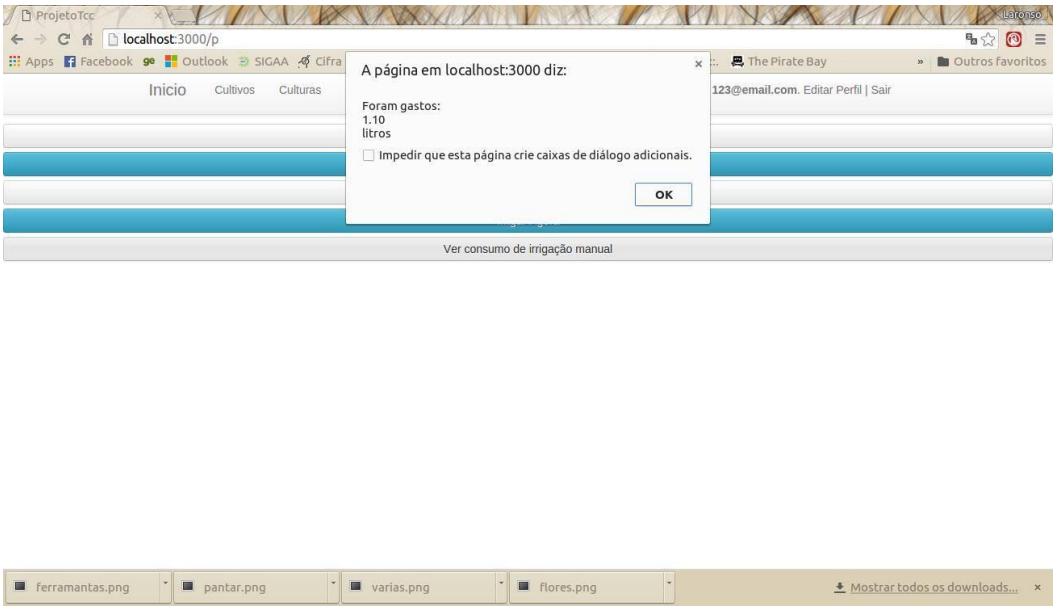

Figura 23: Disparo de irrigação pelo usuário.
Fonte: Produzida pelo autor.

Portanto, a partir dos testes realizados, houve satisfação quanto aos resultados obtidos.

5 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A elaboração do presente projeto obteve um protótipo de um sistema que auxilia no controle básico da área agronômica, automatizando parte do processo e possibilitando o acesso a informações captadas durante a execução de um cultivo, como também, um monitoramento em tempo real do mesmo, por meio da *web*. Para a realização de tal tarefa, foram realizados estudos sobre o desenvolvimento de sistemas *web*, tal qual de tecnologias que possibilitassem uma atuação no meio físico integrada à esses sistemas.

Com base nos resultados obtidos com os testes realizados depois da implementação, obteve-se a eficácia da atuação do sistema no ambiente do experimento, tendo em vista que o mesmo obteve os resultados esperados.

Para trabalhos futuros, propõe-se a implementação de escalabilidade, de maneira que seja possível o gerenciamento de mais de um cultivo por vez, como também a adição de outros dispositivos de monitoramento.

REFERÊNCIAS

ACQUATICOS, **Estados da válvula solenoide**. Disponível em <http://acquaticos.blogspot.com.br/2010/10/valvula-solenoide.html >. Acessado em 19 de Jan de 2016.

ALTOÉ, M. A. C. **Sistema Automatizado de Irrigação Para Culturas Específicas.** Brasília, Distrito Federal: FATECS, 2012.

ARDUINO, site ***Arduino***. Disponível em <http://www.*Arduino*.cc>. Acessado em 12 Nov. 2014.

ARDUINOELETRONICA.COM, **Sensor DHT11**. Disponível em <http://www.*Arduino*eletronica.com.br/2014/03/tutoriais-sensores-xv-umidade.html#.Vp590CorLIU>. Acessado em 19 de Jan de 2016.

ASCO. **Informações técnicas sobre válvulas ASCO**. Disponível em: <http://www.autoval.com.br/sites/default/files/3-inform-tecnicas.pdf>. Acesso em 25 jan. 2016.

BERNERS-LEE, T. et al. ***The World-Wide Web***. Communications of the ACM, Nova York, 1994.

CAELUM, Apostila curso: **Desenv. Ágil para *Web* com *Ruby on Rails*, 2014**. Disponivel em: < http://www.caelum.com.br/apostila-*Ruby*-on-*Rails*/ > Acesso em: 26 de julho de 2014.

CASTRUCCI, P. B. L.; BOTTURA, C. P. **Apresentação. Em: Enciclopédia de automática, volume 1**. São Paulo: Blucher, 2007.

FELIZARDO, I.; BRACARENSE, A. Q**. SENSORES – Conceitos Fundamentais.** Universidade Federal do Rio Grande do Norte, 2013.

FERREIRA, V. M. **Irrigação e Drenagem.** Floriano, PI: EDUFPI, 2011, p 2.

FLANAGAN, D.; MATSUMOTO, Y.. ***The Ruby Programming Language.*** 1 ed. [S.l.]: O'Reilly Media, 2008.

FLIPFLOP.COM, **Sensor de Fluxo**. Disponível em < http://www.filipeflop.com/pd-206c5b-sensor-de-fluxo-de-agua-1-2-yf-s201.html>. Acessado em 19 de Jan de 2016.

FONSECA, E, G.P.; VEGA, A. S. **Tutorial sobre Introdução a Projetos Utilizando o Kit de Desenvolvimento *Arduino*. Anais.** XXXIX Congresso Brasileiro de Educação em Engenharia, Cobenge. Blumenau: FURB, 2011.

FUENTES, V. B. ***Ruby on Rails*: Coloque Sua Aplicação *Web* nos Trilhos.** São Paulo: Casa do Código, 2012.

HARTL, M. ***Ruby on Rails* Tutorial:** Learn *Web* Development with *Rails*. 2 ed. Michigan: Addison-Wesley, 2012.

IHRIG, C. J. ***Pro Node.js* para Desenvolvedores**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2014

JUNIOR, F. A. R.. **Programação Orientada a Eventos no lado do servidor utilizando *Node.js***. Fortaleza –CE: Ifactory Solutions, 2013.

MANTOVANI, E. C.; BERNARDO, S.; PALARETTI, L. F. **Irrigação: princípios e práticas**. Vicosa: UFV, 2006.

PÁDUA, W. **Engenharia de *Software*:** fundamentos, métodos e padrões. [S.l]: LTC, 2000.

PINTO, F. **Sistemas de Automação e Controle**. Vitória, Espirito Santo. SENAI, 2005.

*POSTGRE*SQL 8.0.0. ***The PostgreSQL* Global Development Group**. California: Copyright © 1996-2005 The *PostgreSQL* Global Development Group.

PRESSMAN, R. S. **Engenharia de *Software***: uma abordagem profissional. 7 ed. Porto Alegre: AMGH, 2011.

ROBÓTICA. **História da Robótica**. Disponível em: <<http://roboticagrupo4.blogspot.com.br/2009/05/historia-da-robotica.html>>. Acesso em: 23 jan. 2016.

SABBAGH, R. ***SCRUM*:** Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013, p 65.

SOMMERVILLE, I. **Engenharia de *Software*.** 9 ed. São Paulo: Pearson Prentice Hall, 2011, p 71.

SOUZA, L. ***Ruby*: Aprenda a Programar na Linguagem mais divertida.** São Paulo: Casa do Código, 2012.

STAIR, R. M. ; REYNOLDS, G. W. **Princípios de Sistemas de Informação**. 6 ed. São Paulo: Pioneira Thomson Learning, 2006

TAKAI, O.K.; ITALIANO, I.C.; FERREIRA, J.E. **Introdução a Banco de Dados**. DCC-IME-USP, 2005, p 9.

TEIXEIRA, P. ***Hands-on Node.js,*** 2013. 1p.

TIOBE.COM, **Índice TIOBE, para junho de 2015**. Disponível em <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acessado em 19 de Jan de 2016.

WENDLING, M. **Sensores Versão 2.0**. Colégio Técnico Industrial de Guaratinguetá, 2010.

APÊNDICES

APENDICE A – Código embarcado no *Arduíno*

```
“ #include <dht.h>


#define dht_dpin A1 //Pino DATA do Sensor ligado na porta Analogica A1

dht DHT; //Inicializa o sensor



float vazao; //Variável para armazenar o valor em L/min
int contaPulso; //Variável para a quantidade de pulsos


int umidade;

void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  attachInterrupt(0, incpulso, RISING); //Configura o pino 2(Interrupção 0) para trabalhar como
interrupção
  delay(1000);
}
void loop()
{

  if (Serial.available() > 0) {
    // le os dados da porta serial armazena na variavel 'entrada'
    char entrada = Serial.read();
    digitalWrite(13, LOW);

    DHT.read11(dht_dpin); //Lê as informações do sensor
    umidade = analogRead(A0);
    int Porcento = map(umidade, 1023, 0, 0, 100);
    switch (entrada) {
      case 'A':

              Serial.print(Porcento);
              Serial.print(0);
              Serial.println(entrada);

              break;

      case 'E':
```

```
            Serial.print(DHT.humidity);

            Serial.println(entrada);

            break;


      case 'T':

            Serial.print(DHT.temperature);
            Serial.println(entrada);

            break;


      case 'I':

            digitalWrite(13, HIGH);
            delay(5000);
//=======================================================
            contaPulso = 0;   //Zera a variável para contar os giros por segundos
            sei();      //Habilita interrupção
            delay (1000); //Aguarda 1 segundo
            cli();      //Desabilita interrupção

            vazao = contaPulso / 4.5; //Converte para L/min

//=======================================================
            delay(5000);


            digitalWrite(13, LOW);
            umidade = analogRead(A0);
            Porcento = map(umidade, 1023, 0, 0, 100);
            Serial.print(Porcento);
            Serial.print(0);
            Serial.println("A");

            break;



     case 'L':

            Serial.print(vazao * 10);
            Serial.println(entrada);
```

```
            break;



} }

//delay(2000);
}

void incpulso ()
{
  contaPulso++; //Incrementa a variável de contagem dos pulsos
} ”
```

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Henrique Laronso Macedo Cardeal, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação SISTEMAS DE AUTOMAÇÃO, MONITORAMENTO E CONTROLE DE HORTA PELA WEB de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 29 de Fevereiro de 2016.

Assinatura

Assinatura