

Universidade Federal do Triângulo Mineiro

Rodrigo Ferretti Silva

Sistema WEB de baixo custo aplicado à operação de dispositivos de irrigação

Uberaba

2016

Rodrigo Ferretti Silva

Sistema WEB de baixo custo aplicado à operação de dispositivos de irrigação

Dissertação apresentada ao Programa de  
Mestrado Profissional em Inovação  
Tecnológica da Universidade Federal do  
Triângulo Mineiro (UFTM).

Orientador: Prof. Dr. Wagner Roberto  
Batista

Coorientador: Prof. Dr. Wagner Fernando  
Delfino Angelotti

Uberaba

2016

**Catálogo na fonte: Biblioteca da Universidade Federal do  
Triângulo Mineiro**

S583s Silva, Rodrigo Ferretti  
Sistema WEB de baixo custo aplicado à operação de dispositivos de irrigação / Rodrigo Ferretti Silva . -- 2016.  
93 f. : il., fig., graf.

Dissertação (Mestrado Profissional em Inovação Tecnológica) -- Universidade Federal do Triângulo Mineiro, Uberaba, MG, 2016

Orientador: Prof. Dr. Wagner Roberto Batista

Coorientador: Prof. Dr. Wagner Fernando Delfino Angelotti

1. Irrigação por aspersores. 2. Mecanização agrícola. 3. Automação. 4. Raspberry Pi (Computador). I. Batista, Wagner Roberto. II. Universidade Federal do Triângulo Mineiro. III. Título.

CDU 631.347.3

RODRIGO FERRETTI SILVA

“SISTEMA WEB DE BAIXO CUSTO APLICADO À OPERAÇÃO DE PIVÔS CENTRAIS DE IRRIGAÇÃO”

Trabalho de conclusão apresentado ao Programa de Mestrado Profissional em Inovação Tecnológica da Universidade Federal do Triângulo Mineiro, como requisito para obtenção do título de mestre.


Uberaba, 04 de março de 2016

Banca Examinadora:




---

Prof. Dr. Wagner Roberto Batista  
Orientador – PMPIT - UFTM



Prof. Dr. Luiz Fernando Resende dos Santos Anjo  
Membro Titular – UFTM



---

Prof. Dr. Nélio Muniz Mendes Alves  
Membro titular – IFTM

Dedico esse trabalho à minha família, pelo incentivo, amor e compreensão.

## **AGRADECIMENTOS**

Agradeço a todos que participaram direta ou indiretamente do desenvolvimento desse projeto, principalmente minha noiva, meus pais e meus professores, e entre eles, especialmente os professores Wagner Batista e Wagner Angelotti pela orientação, e os professores David, Alex, Nelio e Luiz Fernando pela ajuda no refinamento deste trabalho.

Agradeço também a todos os profissionais do programa PMPIT pela oportunidade e pelo ótimo trabalho que desempenham.

## RESUMO

O modelo de irrigação por meio do sistema de pivô central utiliza, dentre outros, a aspersão de água e produtos químicos de maneira radial e garante uma irrigação homogênea do terreno compreendido em sua área de alcance. O mecanismo conta com uma estrutura que se desloca em torno de um pivô sobre rodas, enquanto impulsionada elétrica ou hidráulicamente por um equipamento de pressurização que permite que a água e os produtos percorram todo o comprimento da tubulação, movendo-a radialmente em torno do centro e possibilitando assim que seja alcançado seu objetivo. O processo de ativação do equipamento é muitas das vezes controlado de maneira manual, e por isso, em busca de uma maior produtividade, este trabalho tem por objetivo o desenvolvimento de um sistema capaz de automatizá-lo e a simulação de seu funcionamento. Essa automação tem como foco o baixo custo, para que possua uma boa acessibilidade por parte do produtor rural e deve fornecer uma ferramenta *on-line* para monitoramento de dados da área, e interação remota com o equipamento. Para alcançar os objetivos do projeto, foi utilizada a plataforma de desenvolvimento RASPBERRY PI™, e como resultado foi possível obter um sistema WEB capaz de em simulação atender ao proposto, tornando claro que é possível o controle remoto de um ambiente com qualidade e baixo custo.

Palavras-chave: RASPBERRY PI, sistema, automação, Internet.

## **ABSTRACT**

The central pivot irrigation model uses the water and chemical products aspersion in a radial manner and ensures a homogeneous irrigation of the terrain within its area of reach. The device counts on a structure which moves over wheels and around a pivot, while driven electrically or hydraulically by a water pump that allows water and chemicals to get through all the mobile tubing length, moving it radially around its center and allowing the objective to be met. The device activation process is often manually controlled, and because of that, in search of better productivity, this work has the goal to develop an automated system able control that process and the simulation of its functionalities. That automation focuses on low cost, which leads to a good accessibility by the rural producers, will offer an online tool for monitoring the area of the terrain, which allows remote interaction with the device if needed through the Internet. To achieve the goals of the project, the RASPBERRY PI™ development platform was used, and as result it was possible to obtain a WEB system capable of fulfilling the proposed according to the simulations, what makes clear the possibility of remote controlling an environment with quality and low costs.

Keywords: RASPBERRY PI, system, automation, Internet.



## SUMÁRIO

<b>1. INTRODUÇÃO</b> .....	<b>12</b>
<b>2. OBJETIVOS</b> .....	<b>13</b>
2.1. OBJETIVO GERAL .....	13
2.2. OBJETIVOS ESPECÍFICOS .....	13
<b>3. REVISÃO BIBLIOGRÁFICA</b> .....	<b>15</b>
3.1. PIVÔ CENTRAL .....	15
3.2. MÉTODOS DE DETERMINAÇÃO DO TEOR DE ÁGUA .....	19
3.3. SENSORES .....	19
3.4.1. ARDUINO .....	21
3.4. ARDUINO YÚN OU RASPBERRY PI? .....	27
3.5. INFRAESTRUTURA .....	29
3.5.1. BANCO DE DADOS .....	30
3.5.2. ADMINISTRAÇÃO REMOTA .....	32
3.5.2.1. APLICAÇÕES MÓVEIS NATIVAS .....	33
3.5.2.2. APLICAÇÕES WEB .....	35
<b>4. METODOLOGIA</b> .....	<b>36</b>
4.1. PLATAFORMA DE DESENVOLVIMENTO .....	38
4.2. BANCO DE DADOS .....	38
4.3. LINGUAGEM DE PROGRAMAÇÃO .....	38
4.4. SERVIDOR WEB .....	39
4.5. FERRAMENTAS DE DESENVOLVIMENTO .....	39
4.6. ETAPAS DE DESENVOLVIMENTO .....	40
4.7. O SISTEMA .....	42
4.7.1. REQUISITOS .....	42
4.7.2. A ESTRUTURA DO SISTEMA .....	46
4.7.3. O SOFTWARE .....	55
4.7.3.1. SOFTWARE DA RASPBERRY PI™ .....	55
4.7.3.2. SOFTWARE WEB .....	66
4.7.4. COMUNICAÇÃO SEM FIO .....	72
<b>5. RESULTADOS</b> .....	<b>74</b>
5.1. TELA DE LOGIN .....	74
5.2. SELEÇÃO DE DISPOSITIVOS .....	75
5.3. MENU PRINCIPAL .....	76
5.4. PAINEL DE CONTROLE .....	77
5.5. INTERAÇÃO EM TEMPO REAL .....	85
5.7. FLUXO DO SISTEMA .....	88
5.6. FUTUROS TRABALHOS .....	88
<b>6. CONCLUSÕES</b> .....	<b>89</b>
<b>7. REFERÊNCIAS BIBLIOGRÁFICAS</b> .....	<b>91</b>

## ÍNDICE DE FIGURAS

Figura 1: Pivô Central .....	15
Figura 2: Exemplo de Aspersor .....	16
Figura 3: Sensor para Teor de água no solo .....	20
Figura 4: ARDUINO™ UNO.....	22
Figura 5: ARDUINO Yún .....	23
Figura 6: RASPBERRY PI B+ .....	26
Figura 7: Estrutura do sistema utilizando RASPBERRY PI™ .....	46
Figura 8: Conversor analógico-digital MCP3008.....	47
Figura 9: Esquema RASPBERRY PI com conversor MCP3008 .....	48
Figura 10: Esquema Elétrico (RPI e Conversor MCP3008).....	48
Figura 11: Comunicação SPI com MCP3004/3008 utilizando segmentos de 8 bits.....	49
Figura 12: Estrutura para simulação de leituras .....	52
Figura 13: Esquema Eletrônico (simulação) .....	53
Figura 14: Relay Shield SRD-05VDC-SL-C .....	54
Figura 15: Pseudocódigo da lógica local dos dispositivos RASPBERRY PI™ .....	56
Figura 16: Fluxograma (RPI) .....	56
Figura 17: Modelagem de dados do banco de dados da RPI .....	58
Figura 18: Pseudocódigo do comportamento do banco de dados .....	65
Figura 19: Fluxograma (Banco de Dados) .....	66
Figura 20: Esquema do banco de dados do sistema WEB .....	67
Figura 21: Adaptador WiFi Alfa AWYS036H.....	73
Figura 22: Tela de Login .....	75
Figura 23: Seleção de dispositivos .....	76
Figura 24: Painel de Controle.....	78
Figura 25: Comunicação (Servidor WEB e RPI) .....	81
Figura 26: Agendamento para datas específicas.....	83
Figura 27: Interação em tempo real .....	86
Figura 28: Fluxograma do Sistema.....	88

## LISTA DE SIGLAS

FTP – File Transfer Protocol;

GPIO – General Purpose Input/Output;

HDD – Hard Drive Disk;

HTTP – Hypertext Transfer Protocol;

HTTPS – Hypertext Transfer Protocol Secure;

PC – Personal Computer;

PWM – Pulse Width Modulation;

RPI – Raspberry Pi™;

SGBD – Sistema Gerenciador de Banco de Dados;

SSH – Secure Shell;

USB – Universal Serial Bus;

VNC – Virtual Network Computing;

VPN – Virtual Private Network;

WEB – Rede, Internet;

## 1. INTRODUÇÃO

O sistema de irrigação em pivô central é amplamente utilizado na América Latina e no mundo, por isso há várias soluções de automação que visam o controle de seu funcionamento sem a necessidade de intervenção humana a partir de sistemas dedicados a garantir as condições ideais para o bom desempenho da plantação (Longo *et al.*, 2011; Oliveira *et al.*, 2011).

Todavia a aquisição desse tipo de sistema pode ter um custo elevado, chegando inclusive a milhares de reais, o que abre espaço para o desenvolvimento de formas de minimização de custos (Nascimento *et al.*, 2009; Morales, 2011).

Existem atualmente diversas plataformas de desenvolvimento que têm seu foco voltado para automação, e que são inerentemente de baixo custo, dentre elas as plataformas Arduino™, Raspberry Pi™, BeagleBone™, Teensy™, Pinguino™, MSP430 Launchpad™, entre outras (Bysani *et al.*, 2013; Wirth *et al.*, 2013).

Dentre as plataformas de desenvolvimento citadas, destacam-se as duas primeiras por serem nomes consolidados no mercado, o que tem propiciado a criação de vários trabalhos ao redor do mundo, o que direcionou os estudos realizados neste projeto (Wong *et al.*, 2015).

Inicialmente, procurou-se utilizar a plataforma Arduino™ para o desenvolvimento de um sistema de baixo custo capaz de realizar o controle automatizado de dispositivos de irrigação do tipo pivô central, porém, por um dos requisitos do sistema ser justamente o controle dos dispositivos pela Internet, a necessidade da visualização de informações sobre a cultura em tempo real, e ao serem levados em consideração fatores como custo e benefício optou-se pela utilização da plataforma de desenvolvimento Raspberry Pi™ por uma série de razões que serão discutidas ao longo do texto.

Embora o protótipo apresentado seja voltado à administração de pivôs centrais, a ideia central do projeto não possui essa restrição. Isso possibilita a utilização de seus métodos em uma gama de modalidades de irrigação racional, ou para qualquer outro fim em que seja necessária tomada de decisão de maneira automatizada com base em leituras de sensores locais, ou por um usuário ou grupo

de usuários, remotamente, em qualquer lugar do mundo, pela Internet. Justamente por essa grande diversidade de aplicações, o sistema também contempla o recurso de manutenção do histórico das interações, automáticas ou não, para análise da parte interessada por meio de relatórios.

Em outras palavras, o foco desse trabalho é demonstrar, por meio de simulação, como é possível fazer uso da plataforma de desenvolvimento Raspberry Pi™ no controle automatizado de dispositivos com base na leitura de informações do ambiente em que se encontram os sensores, e que ao mesmo tempo permita interação direta do responsável por meio da Internet.

Os maiores diferenciais buscados por esta abordagem, além de procurar manter sempre o baixo custo do sistema, foram a independência do sistema em relação à estrutura de hardware utilizada, permitindo a utilização de diversos tipos de sensores, leituras e equipamentos, as capacidades de seu aprimoramento futuro, com a adição de novos recursos e funcionalidades, e principalmente sua capacidade de armazenar informações históricas sobre o desenvolvimento da cultura, o que pode oferecer subsídios para estudos futuros ou geração de indicadores para tomada de decisões gerenciais pelo administrador da plantação.

## **2. OBJETIVOS**

### **2.1. OBJETIVO GERAL**

Este trabalho tem como objetivo o desenvolvimento de uma proposta de sistema WEB de baixo custo que permita o controle de irrigação utilizando Pivô Central, mas que possa ser utilizado em outros sistemas de tomada de decisão.

### **2.2. OBJETIVOS ESPECÍFICOS**

Os princípios fundamentais aos quais o desenvolvimento se apoiará são os seguintes:

- Baixo Custo;
- Funcionalidade;

- Simplicidade;
- Facilidade;
- Segurança;
- Escalabilidade.

O sistema, além de ser de baixo custo, deve possuir principalmente as seguintes funcionalidades, como seus objetivos específicos:

- Permitir acompanhamento *on-line* em tempo real sobre:
  - As condições do solo;
  - O status de funcionamento do pivô central;
- Armazenar informações históricas sobre a cultura;
- Permitir comandos manuais *on-line* em tempo real para:
  - Acionamento / Desligamento do pivô central;
  - Alteração de parâmetros para calibração das funcionalidades automáticas do controle de irrigação;
- Possuir comunicação sem fio entre o controle do pivô central e a plataforma de controle WEB;
- Ajudar a aumentar a produtividade da cultura ao oferecer as condições ideais de irrigação para o seu desenvolvimento;
- Aperfeiçoar a utilização de recursos energéticos e hídricos de maneira automática;
- Ser utilizado como uma alternativa mais viável financeiramente às soluções encontradas no mercado;
- Customização dos parâmetros de tomada de decisão automática pelo sistema de acordo com a cultura, permitindo diferentes tipos de leituras e sensores auxiliares ao de teor de água do solo;

### 3. REVISÃO BIBLIOGRÁFICA

#### 3.1. PIVÔ CENTRAL

O pivô central é um sistema de irrigação que consiste na preparação de áreas circulares para que sejam irrigadas por intermédio de tubulação suspensa em estruturas metálicas triangulares e móveis por meio de rodas, mais comumente pneus, que são empregados em escala mundial na agricultura (Oliveira *et al.*, 2011).

Estas estruturas possuem comprimento equivalente ao raio da área alcançada, são rotacionadas concentricamente em relação ao terreno e, durante este processo, água e possivelmente inseticidas, fertilizantes e fungicidas são aspergidos por toda sua extensão, de maneira a cobrir toda sua superfície após uma volta completa da estrutura (Figura 1) (Longo *et al.*, 2011).

**Figura 1:** Pivô Central



**Fonte:** Autor, 2015.

Esse sistema conta com uma bomba hidráulica que ao ser acionada faz a água percorrer a tubulação de maneira a ser aspergida por dispositivos dispostos ao

longo de seu comprimento (Figura 2), e pode gerar ao mesmo tempo o deslocamento radial dessa estrutura, a depender do modelo do sistema, em torno do pivô central (Longo *et al.*, 2011; Oliveira *et al.*, 2011).

O objetivo do pivô central é garantir que toda a área compreendida no raio de ação da tubulação tenha seu solo irrigado, além de aplicar produtos químicos pertinentes aos cuidados necessários à cultura que está sendo feita na área. Assim, é possível garantir que a irrigação seja feita de maneira uniforme e se obtenha um nível de teor de água do solo ideal ao desenvolvimento dessa cultura em toda a área de ação da estrutura de irrigação (Longo *et al.*, 2011).

**Figura 2:** Exemplo de Aspensor



**Fonte:** Autor, 2015

A automação desse processo utilizando sistemas de informação é um dos recursos da irrigação de precisão, e assim como traz benefícios à produção também traz custos com os quais muitos produtores podem preferir não investir, mesmo que pudessem se beneficiar desse sistema, em função do porte de sua cultura. Isso pode ser ocasionado pela avaliação do investimento inicial que muitas das vezes pode se tornar impeditivo ou até mesmo a questão de custo/benefício, onde mesmo



que seja possível a implantação do referido sistema, o retorno não será suficiente para justificar seu custo (Moraes *et al.*, 2014).

Assim, a contratação de um sistema de irrigação de precisão fica reservada a determinado nicho de agricultores que possuem maior capacidade de investimento, o que torna a utilização da irrigação, qualquer que seja seu tipo, pelos pequenos agricultores uma prática manual. Isso pode limitar seus resultados, ao tornar sua irrigação não tão eficiente quanto seria com a utilização de uma máquina dedicada a garantir as condições ideais do solo durante todas as horas do dia, alternativa capaz de aumentar a eficiência do processo de irrigação (Nascimento *et al.*, 2009; Morales, 2011).

Existem algumas maneiras diferentes de se otimizar um sistema de irrigação, tanto ao se considerar economia de energia quanto os resultados obtidos em relação ao teor de água do terreno, e entre eles se destaca o controle automático e temporizado dessa irrigação (Moreno *et al.*, 2012), o que, por consequência, torna útil o desenvolvimento de uma alternativa de baixo custo que se proponha a facilitar o acesso à automação do processo de irrigação, tornando-o mais acessível ao pequeno produtor.

O sistema de irrigação baseado em pivô central é amplamente utilizado e está entre os meios de irrigação mais populares em vários países como Brasil, Argentina, Espanha, entre outros (Moreno *et al.*, 2012; Dong *et al.*, 2013). Além disso, a chamada agricultura de precisão, que consiste em estabelecer o tempo e quantidade de irrigação corretos para se obter a maximização dos resultados na produção (Dong *et al.*, 2013), tem se tornado também cada vez mais interessante.

No Brasil, por exemplo, em 2006 havia cerca de 840.000 hectares que utilizavam a irrigação de pivô central, o que representava por volta de 19% da área total irrigada no país, segundo o Censo Agropecuário daquele ano. Em confirmação a esta tendência, houve um aumento de 32% na área irrigada por pivôs centrais desde o referido senso até o ano de 2013, segundo a Agência Nacional de Águas (Agência Nacional de Águas, 2015).

Isso é uma quantidade bastante expressiva e que deixa clara a viabilidade da realização de estudos que facilitem ou aprimorem a execução dos trabalhos e processos associados a este tipo de irrigação em busca de maior produtividade (Paulino *et al.*, 2011).

São vários os fatores que influenciam a produtividade e, por consequência, a rentabilidade de determinada plantação. Consoante a essa grande diversidade de variáveis o *Irrigation Advisory Service* desenvolveu métodos para realizar sugestões acerca da quantidade e uniformidade de água a se aplicar no terreno. Esses métodos ressaltam a relação entre a quantidade de água aplicada na irrigação com seu correspondente benefício econômico. Todavia, essa é uma relação muito complexa, e extremamente dependente do manuseio de quem cuida da área que, por sua vez, depende de seu sistema de irrigação (Moreno *et al.*, 2012).

Um exemplo da sensibilidade dos resultados econômicos em relação a alterações paramétricas durante o processo de irrigação é a importância da manutenção e troca dos aspersores de sistemas de irrigação de pivô central, que pode representar ganhos significativos de produtividade com o aumento do coeficiente de uniformidade do sistema, e existe inclusive a possibilidade de quantificar economicamente este coeficiente de aplicação da água (Zolin *et al.*, 2012).

Ao se observar informações como essas (Zolin *et al.*, 2012), nota-se a importância do aprimoramento de técnicas e a necessidade de inovação para que perdas econômicas relativas a dificuldades de gestão da área de plantio, no que concerne à sua irrigação, possam ser minimizadas. Isso pode ser feito ao se aprimorar as ferramentas de manuseio do solo e oferecer relatórios em tempo real com dados obtidos da área onde quer que o responsável esteja, e a qualquer momento possibilitar ações imediatas por parte dele em caráter de contingência quando necessário. Essas interações manuais complementam um sistema automatizado de irrigação baseado no retorno de nós de sensores dispostos pela área, que oferecem oportunidade de acompanhamento da cultura, além de evitar a dependência exclusiva de solução exclusivamente automatizada.

Há várias metodologias ou arquiteturas que podem ser aplicadas na montagem de um sistema como o proposto (Dong *et al.*, 2013). Infelizmente, são poucas as soluções, principalmente de baixo custo, que permitam o monitoramento da área cultivada, principalmente de maneira remota (pela *Internet*). Ainda mais raras são as que permitem intervenção remota pelo usuário em possíveis situações de emergência que possam ocasionar perdas econômicas, ou que produzam indicadores aos seus usuários.

### **3.2. MÉTODOS DE DETERMINAÇÃO DO TEOR DE ÁGUA**

Para que seja possível o desenvolvimento de um sistema capaz de realizar o que foi sugerido é natural que seja necessária a leitura de informações do ambiente onde se encontra o mecanismo de irrigação a fim de se obter informações como umidade relativa do ar, teor de água no solo ou temperatura referente à área afetada pelo mecanismo de irrigação para que se torne possível a realização de um controle automático (Moreno *et al.*, 2012), e isso pode ser feito com a utilização de sensores.

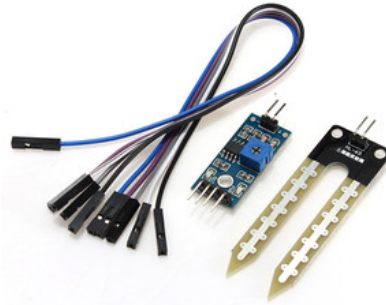
### **3.3. SENSORES**

Os sensores podem ser classificados como uma espécie de transdutores que, por sua vez, são componentes que transformam um tipo de energia em outra, assim como acontece com os motores, que transformam energia química ou elétrica em energia mecânica. Assim, os sensores são transdutores específicos que transformam algum tipo de energia em energia elétrica, o que é utilizado, geralmente, de maneira analógica para mensurar características ou condições do ambiente. (Chen *et al.*, 2012).

Há vários tipos de sensores, como o exemplificado pela Figura 3, que podem possuir diversas formas físicas e de funcionamento, e também podem ser constituídos com base em uma infinidade de diferentes tecnologias que envolvem diversos materiais e procedimentos para a obtenção das medições (Su *et al.*, 2012). Estes sensores também podem ser utilizados de várias maneiras, como, por

exemplo, aferição da umidade relativa do ar ou determinação do teor de água no solo, e podem ser utilizados desde a produção comercial de aparelhos de ar-condicionado até em processos industriais como a indústria farmacêutica e produção de semicondutores (Zhao *et al.*, 2011; Wang *et al.*, 2012).

**Figura 3:** Sensor para Teor de água no solo



**Fonte:** Autor, 2015.

De maneira geral, os sensores são a maneira mais usual de se obter dados do meio ambiente para tratamento em sistemas dedicados como o proposto nesse projeto (Dong *et al.*, 2013).

### 3.4. PLATAFORMAS DE DESENVOLVIMENTO

Para a realização das referidas leituras por meio dos sensores é necessário hardware destinado à realização de suas leituras, e no campo do baixo custo são comumente utilizadas plataformas de desenvolvimento tais como ARDUINO™ ou RASPBERRY PI™ (Wong *et al.*, 2015).

Assim, levando em consideração que há neste estudo o foco em programação de alto nível para possibilitar interoperabilidade de *hardware*, deverá ser de fácil implementação qualquer sensor que o usuário desejar utilizar, dessa forma, mesmo com a escolha da plataforma de desenvolvimento RASPBERRY PI™ para o desenvolvimento do sistema (Wirth *et al.*, 2013; Baggaley, 2014), é importante citar alguns pontos sobre a plataforma de desenvolvimento ARDUINO™.

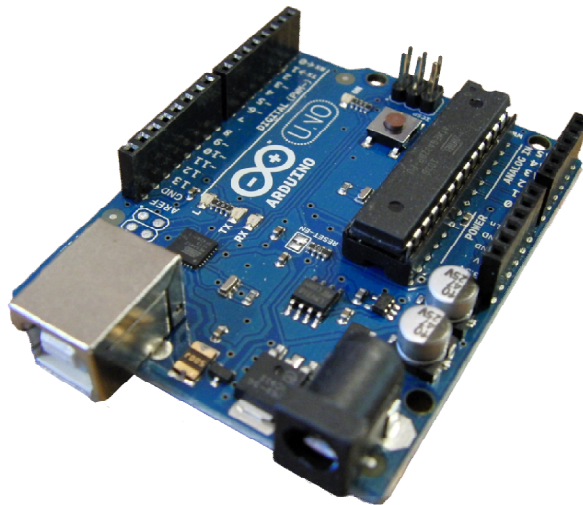
A importância da plataforma ARDUINO™ para este projeto específico se restringe ao fato de sua utilização em conjunto com a RPi para suprir sua falta de conectores GPIO analógicos durante a etapa de desenvolvimento do sistema, a ser exposta posteriormente, o que pode ser replicado em aplicações do mundo real conforme a necessidade, em substituição a um conversor analógico-digital por motivos de simplicidade (De Souza *et al.*, 2011; Wong *et al.*, 2015).

Esta interação entre ARDUINO™ e RASPBERRY PI™ pode ser feita de várias maneiras, até mesmo para efeito de compatibilidade com algum sensor ou *shield* específico. É possível também que exista uma necessidade específica do monitoramento que não possua sensores compatíveis com a RPi, ou que nela exista alguma restrição de funcionamento em relação à sua utilização com a ARDUINO™. Em todos esses casos, é possível combinar o funcionamento de ambas as tecnologias se necessário (De Souza *et al.*, 2011).

### 3.4.1. ARDUINO

Dada a utilização da plataforma ARDUINO™ nos testes executados no sistema, um maior aprofundamento dentro do tema se faz necessário (Arduino, 2016).

O ARDUINO™ é, em suas diversas versões, uma plataforma de prototipagem eletrônica de *hardware* livre e de placa única (Figura 4). Essa plataforma é baseada no microcontrolador Atmel AVR, que suporta entrada e saída e trabalha com uma linguagem de programação, essencialmente C/C++, que tem como objetivo justamente solucionar problemas em que seja necessária automação de baixo custo, flexível e de fácil uso e implementação, inclusive por amadores que não teriam acesso a mecanismos de maior complexidade (Wong *et al.*, 2015).

**Figura 4: ARDUINO™ UNO**

**Fonte: (Arduino, 2016)**

A referida plataforma é amplamente utilizada em projetos de pesquisa de diversas naturezas ao desempenhar funções como, por exemplo, controle de tensão e medida da frequência angular em motores DC (Gonçalves *et al.*, 2014), monitoramento de temperaturas (Bar-Zeev *et al.*, 2014; Nagy *et al.*, 2014; Oviedo *et al.*, 2014), obtenção e tratamento de valores provenientes da intensidade de luz observada em processos de quimiluminescência com a utilização de fotodiodos (Novo *et al.*, 2014; Oviedo *et al.*, 2014), percepção de alterações de umidade (Oviedo *et al.*, 2014), transmissão de informações obtidas remotamente por meio de sensores, com a utilização de tecnologias sem fio (Oviedo *et al.*, 2014) e desenvolvimento de agentes robóticos (Francisco *et al.*, 2014), entre outras aplicações.

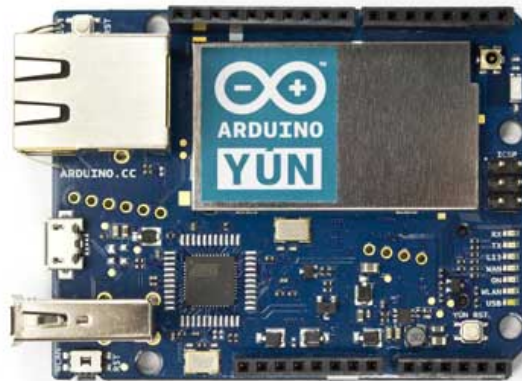
Com a utilização dessa plataforma é possível fazer um controle de baixo custo, automatizado e parametrizado com entradas e comandos personalizados de um operador que pode intervir nas decisões pré-configuradas que seriam realizadas de maneira automática, de acordo com o sua necessidade (Wong *et al.*, 2015).

O *hardware* do ARDUINO™ é baseado em microcontroladores, o que é ideal para processos de automação de baixo custo voltados simplesmente para

operacionalização de hardware de maneira automática, porém possui uma série de desvantagens em relação a microprocessadores quando se pensa no enriquecimento desses processos de automação com a utilização de recursos mais refinados em termos de software, e até mesmo em se comparando seus poderes de processamento de informações (Wirth *et al.*, 2013).

Para dar suporte a este tipo de recurso existem versões mais elaboradas da plataforma ARDUINO™, como a ARDUINO Yún™ (Figura 5), que conta com um microprocessador em adição ao microcontrolador que é característica das placas ARDUINO™ mais simples (De Souza *et al.*, 2011).

**Figura 5:** ARDUINO Yún



**Fonte:** (Arduino, 2016)

Em outras palavras, o ARDUINO Yún™ trabalha como se fosse um computador acoplado permanentemente a uma placa ARDUINO™, possuindo tanto um controlador ATmega32u4 quanto o Atheros AR9331, um processador que suporta uma distribuição Linux denominada OpenWrt-Yun (Arduino, 2016).

Desta maneira, é possível desenvolver aplicações mais robustas em termos de software utilizando diversas linguagens de programação diferentes, além de diversos serviços como servidores WEB (de páginas da *Internet* pelos protocolos HTTP e HTTPS), SGBDs (Sistema Gerenciador de Bancos de Dados), FTP (*File Transfer Protocol*), além de facilitar o controle remoto do *hardware* por meio de serviços como SSH (*Secure Shell*) ou mesmo com recursos de captura de tela,

utilizando soluções VNC (*Virtual Network Computing*), o que permite possíveis manutenções de *software* sem a necessidade de presença física no local para realizá-la. Pelos motivos listados, entre outros, especificamente a versão ARDUINO Yún™ é considerada uma das versões mais recomendadas para desenvolvimento WEB (Arduino, 2016), como é o escopo deste trabalho, dentre as opções da plataforma ARDUINO™ (Francisco *et al.*, 2014).

A plataforma de desenvolvimento ARDUINO™ é amplamente utilizada atualmente por diminuir a complexidade e o custo de projetos, o que viabiliza economicamente a automação de processos e desenvolvimento de novas tecnologias (De Souza *et al.*, 2011).

Isso pôde ser alcançado já que esta plataforma é voltada à solução de problemas com foco em baixo custo e possui comunidades cada vez mais crescentes de entusiastas, entre eles professores, alunos, artistas e até mesmo hobistas, por se tratar de uma ferramenta voltada ao público em geral, inclusive os amadores, o que a transforma em uma ótima opção na criação de dispositivos que permitam a interação com o ambiente (Wong *et al.*, 2015).

Dessa maneira, se trata de uma plataforma ideal em cenários onde os problemas não sejam altamente exigentes em termos de processamento, mas que tenham demandas em termos de interação com outros equipamentos, o que pode ser feito pelos conectores GPIO presentes nas diversas versões do ARDUINO™ (De Souza *et al.*, 2011).

Por ser uma plataforma aberta, ela pode ser montada com base no esquema disponibilizado gratuitamente em sua página oficial, como também pode ser comprada já montada e pronta para uso, de diversas empresas que montam clones das placas ARDUINO™ para venda, ou ainda placas originais ARDUINO™ (Arduino, 2016).

A plataforma conta com uma interface para desenvolvimento e conecta-se ao PC por meio de uma porta USB, o que simplifica a programação de seu controlador, sem a necessidade de compiladores ou hardwares adicionais (De Souza *et al.*, 2011). No caso de versões como a ARDUINO Yún™, a ligação é permanente, sendo



que sua parte microprocessada trabalha como se fosse um PC vinculado a uma placa ARDUINO™ mais simples, sendo a maneira de trabalho semelhante à realizada pela sua ligação USB, inclusive pelos procedimentos de gravação de código para execução em loop pelo microcontrolador, utilizando IDE específica.

Há também diversos componentes (ou módulos) compatíveis com a plataforma ARDUINO™, que trazem bibliotecas específicas para seu uso, e que podem ser facilmente incorporadas no código (De Souza *et al.*, 2011), além de muitas das vezes possuírem compatibilidade com a plataforma de desenvolvimento RASPBERRY PI™ sem grandes modificações de aplicabilidade, ou mesmo sem nenhuma diferença de utilização em comparação ao seu modo de operação quando conectadas a uma das diferentes versões das placas ARDUINO™.

#### **3.4.2. RASPBERRY PI**

Há, no entanto, certas aplicações para as quais a plataforma de desenvolvimento ARDUINO™ pode não ser a escolha ideal (Bysani *et al.*, 2013).

Assim, há também cenários onde uma alternativa à plataforma ARDUINO™ seria útil é exatamente o presente neste trabalho, ou seja, não a criação de qualquer sistema, mas a criação de um sistema essencialmente de baixo custo e com necessidades específicas em termos de *software* (Heeks e Robinson, 2013).

É fato que a plataforma ARDUINO™ representa inerentemente uma plataforma idealizada para desenvolvimento de automatização de baixo custo, todavia há alternativas mais atrativas em termos de investimento financeiro capazes de solucionar as demandas do sistema tão bem quanto a plataforma ARDUINO™, como a plataforma RASPBERRY PI™ (Figura 6), por exemplo (Mitchell e Lahtinen, 2012).

**Figura 6:** RASPBERRY PI B+



**Fonte:** <https://www.raspberrypi.org/>

A RASPBERRY PI™ é uma pequena placa que funciona como computador completo, presente inclusive em versões que custam apenas cinco dólares, baseada em um poderoso (considerando o tamanho do equipamento) processador multimídia de aplicações, sendo ele o BCM2835 ou BCM2836, a depender do modelo (BCM2836 para RASPBERRY PI 2, e BCM2835 para anteriores), além de contar com portas USB (*Universal Serial Bus*), Ethernet (rede), áudio/vídeo, HDMI (*High Definition Multimedia Interface*), em adição a pinos GPIO (*General Purpose Input/Output*), essenciais para a utilização de sensores e manipulação de outros *hardwares* (Heeks e Robinson, 2013; Wirth et al., 2013).

Assim como o supracitado ARDUINO Yún™, também pode funcionar com o sistema operacional Linux, existindo inclusive uma distribuição específica denominada Raspbian (Raspbian, 2016), baseada em Debian (Wong et al., 2015).

Não há, todavia, dispositivos internos de armazenamento de dados, sendo este armazenamento realizado por meio da utilização de um cartão SD, preferencialmente de Classe 10, muito embora classes inferiores também sejam compatíveis (ao custo do desempenho do dispositivo), como os cartões “classe 4”, por exemplo, mas para melhor desempenho podem ser utilizados HDDs externos (Sansing, 2013).

A alimentação de energia do RASPBERRY PI™ é feita por meio de uma porta MICRO USB, que pode ser diretamente ligada à porta USB de um PC qualquer, ou por uma fonte (ou bateria) de 5V, sendo que segundo o próprio fabricante (Raspberry PI, 2016) de 1200mA a 2500mA, se houver necessidade da utilização de todas as suas portas USB (quatro, no caso do modelo utilizado no desenvolvimento deste trabalho, o B+), sem a necessidade de utilização de um HUB USB com alimentação própria (Sansing, 2013).

A RASPBERRY PI™ é um computador completo do tamanho de um cartão de crédito, que tem apresentado um aumento de popularidade muito grande recentemente por suas mais diversas aplicabilidades, indo desde ensino de programação a jovens (Bysani *et al.*, 2013; Uwire, 2013) a projetos de pesquisa, principalmente os focados em aplicações de baixo custo (Bysani *et al.*, 2013; Sansing, 2013; Wirth *et al.*, 2013; Sreejith *et al.*, 2014).

A versão utilizada neste trabalho (RASPBERRY PI™ B+) utiliza o chip BCM2835, que inclui um processador ARM1176JZF-S de 700MHz, com 512MB SDRAM, GPU VideoCore IV, e por não possuir dispositivo de armazenamento próprio, possui um *slot* para cartões Micro SD para inicialização e armazenamento de longo prazo (Dong *et al.*, 2013; Sansing, 2013; Sreejith *et al.*, 2014).

A RASPBERRY PI™ propõe oferecer as mesmas ferramentas de um computador. Assim, é possível instalar servidores para serviços como banco de dados, WEB, SSH e SVN, por exemplo, além de possibilitar o uso de qualquer linguagem de programação para realizar a interação com o hardware, mais especificamente os conectores GPIO, como o caso do *Python*, que é uma linguagem comumente utilizada para manipulação de pinos GPIO na RASPBERRY PI™ (Wirth *et al.*, 2013; Baggaley, 2014).

### **3.4. ARDUINO YÚN OU RASPBERRY PI?**

Há alternativas ao ARDUINO™ e RASPBERRY PI, como, por exemplo, BeagleBone™, Nanode™ e LiberiumWaspote™, porém por motivos de aceitação no mercado, custo, disponibilidade de *hardware*, comunidade e material, abre-se

uma vantagem dos dois primeiros em relação aos demais, o que restringiu o estudo a uma dessas opções (Wong *et al.*, 2015).

Comparando-se as opções principais no tocante ao objeto do estudo, basicamente o processador do ARDUINO Yún™ é relativamente inferior ao da RASPBERRY PI™, sendo de 400MHz (contra 700MHz da RPi), além de não incluir uma GPU (*Graphics Processor Unit*), que possibilita maior versatilidade para aplicativos que se utilizem de recursos gráficos ou de multimídia (Sansing, 2013; Uwire, 2013).

Assim, o RPi (RASPBERRY PI) possui um hardware superior, de maneira a executar serviços como servidores de arquivo e servidores WEB com melhor desempenho, além de uma vantagem financeira, já que o modelo empregado neste trabalho (RASPBERRY PI B+) pode ser facilmente encontrado na faixa dos R\$ 200,00 no Brasil, sendo que a outra opção estava sendo disponibilizado por praticamente o dobro do preço, variando em torno de R\$ 350,00 a R\$ 400,00, preços esses levantados durante o período de análise para aquisição dos materiais a serem utilizados neste projeto (Sansing, 2013).

Há, naturalmente, vantagens e desvantagens em ambas as opções, por mais atrativo que o valor de compra do equipamento possa parecer. A plataforma ARDUINO™ seria a escolha mais indicada para trabalhos relacionados a controle de tempo, ou aplicações PWM (*Pulse Width Modulation*), ou “Modulação de Largura de Pulso”, além de possuir nativamente mais conectores GPIO e suporte a *input* analógico, algo não contemplado no suporte GPIO da RPi (Raspberry PI, 2016)

Todavia, há sempre a possibilidade de se trabalhar com multiplexadores para sanar essa deficiência de quantidade de conectores GPIO da RPi, e um simples conversor analógico-digital, ou mesmo a utilização de uma versão mais simples do ARDUINO™ trabalhando em conjunto com a RPi por meio de comunicação serial para o tratamento de entradas analógicas, se mostraria uma opção menos onerosa que o próprio ARDUINO Yún™, uma vez que uma das versões mais simples do ARDUINO™, a ARDUINO UNO™ que pode ser encontrada facilmente por menos de R\$ 50,00, e dessa maneira é possível integrar as duas tecnologias para que uma compense as limitações da outra (Heeks e Robinson, 2013; Sansing, 2013).

Existem vários *Shields* e componentes desenvolvidos para uso com a plataforma ARDUINO™ e sua esmagadora maioria pode ser utilizada também com o RASPBERRY PI™, ou seja, são compatíveis com ambas as tecnologias (Raspberry PI, 2016).

Em outras palavras, pode-se concluir que a utilização de quaisquer das duas opções seriam suficientes para o desenvolvimento deste projeto. Todavia, de maneira simples, e pelo que foi percebido nos levantamentos prévios à execução dos trabalhos, o ARDUINO Yún™ tem desvantagens em relação à RASPBERRY PI™ em termos de desempenho, processamento, preço e aplicações gráficas, mas tem como seu ponto forte o suporte a leituras analógicas em seus conectores GPIO (Sansing, 2013; Wong *et al.*, 2015).

Assim, a limitação GPIO da RASPBERRY PI™ pode ser sanada facilmente integrando-a com uma placa ARDUINO UNO™ ou um conversor analógico-digital, o que torna seu suporte GPIO suficiente para o desenvolvimento do sistema, além de possuir excelência na sua parte mais fundamental, ou seja, programação de alto nível e utilização de serviços que exigem processamento a um mais baixo custo e com melhor desempenho (Francisco *et al.*, 2014).

Essa diferença de *hardware* é importante, pois o atual projeto é extremamente dependente de interações com bancos de dados (será detalhado adiante), o que pode e vai se beneficiar do poder extra de processamento presente na plataforma RASPBERRY PI™ (Bysani *et al.*, 2013).

### **3.5. INFRAESTRUTURA**

É interessante ao desenvolvimento de um sistema de informação ou de automação, a avaliação da infraestrutura a ser criada para atender os objetivos desse sistema de maneira satisfatória. Assim, dentro do escopo deste projeto é necessária a discussão sobre aspectos relacionados a bancos de dados, tecnologias de transmissão de dados sem fio e tecnologias WEB para disponibilização de informações on-line e em tempo real, e como essas tecnologias podem se aplicar e agregar ao resultado final esperado para o trabalho (Eweek, 2009).

Dessa forma há vários aspectos a serem trabalhados, de maneira a se destacarem os que os que seguem nos próximos subtópicos.

### 3.5.1. BANCO DE DADOS

Independentemente da arquitetura de software utilizada no desenvolvimento de um sistema de informação, e a depender dos objetivos do mesmo, se faz necessário possuir um repositório de informações para que estas sejam tratadas pelo sistema automaticamente ou por meio de interação com seus usuários. A maneira mais comum de trabalhar com essa necessidade é a utilização de servidores de banco de dados, que são máquinas dedicadas a transpor estes obstáculos (Eweek, 2009).

A escolha do banco de dados é ponto de fundamental importância para assegurar a funcionalidade, confiabilidade e desempenho de um sistema de informação (Eric e Eric, 2002), portanto, assim como outras ferramentas que devem ser cuidadosamente escolhidas para outros fins que não manipulação direta de dados, em meio a inúmeras opções, proprietárias ou *open-source*, estas têm cada vez mais destaque no mercado devido a uma série de vantagens práticas.

As principais vantagens margeiam o fato de possuírem reduzido custo de desenvolvimento e de implantação em relação à utilização das opções proprietárias, com proporcional, e em algumas vezes superior, nível de recursos, ferramentas e qualidade, além de contarem com uma comunidade maior, solícita e aberta a auxiliar em sua utilização (Eweek, 2009).

Assim, entre as opções gratuitas mais populares, já há vários anos, destacam-se dois principais bancos de dados, MySQL™ e PostgreSQL™, que possuem diferentes fragilidades e potencialidades, porém ambos têm o mesmo objetivo e o alcançam com competência (Eweek, 2010).

Muito embora o banco de dados MySQL™ esteja consolidado já há vários anos, sendo um excelente banco de dados principalmente para desenvolvimento WEB, muito comumente utilizado em conjunto com a linguagem PHP, o banco de

dados PostgreSQL™ se mostra detentor de mais avançados recursos em geral (Eweek, 2010).

Grande parte disso se deve ao fato do PostgreSQL™ implementar melhores ferramentas de programação interna (no próprio banco de dados, ou seja, independente de software externo) com linguagens suportadas nativamente como a linguagem PL/SQL, e recursos como o tipo de dados HSTORE, que não se encontra disponível na versão gratuita do MySQL (Eweek, 2010).

Assim, o PostgreSQL™ pode ser considerado um dos mais avançados sistemas gerenciadores de bancos de dados *open source* que existem, se não o mais avançado, além de vantagens na agilidade das consultas, principalmente as mais complexas que geralmente são desenvolvidas visando a montagem de relatórios (Eweek, 2010).

Embora este trabalho não possua um nível de complexidade em termos de armazenamento de dados que exija a utilização de todos os recursos presentes no banco de dados PostgreSQL™, sua maior agilidade no acesso a dados por seus algoritmos de pesquisa, lembrando também da limitação de hardware presente na RASPBERRY PI™, que embora seja superior ao do ARDUINO Yún™, não se compara com um servidor de banco de dados especificamente dimensionado para tal fim, pode ser fator fundamental para o bom funcionamento do sistema e, conseqüentemente, proporcionar melhores resultados finais em termos de desempenho (Eweek, 2009).

Em outras palavras, o PostgreSQL™ parece ser a alternativa mais interessante para o projeto pela maneira com qual este banco de dados foi planejado, e também por suas capacidades técnicas em comparação ao outro sistema gerenciador de banco de dados considerado (Eweek, 2010).

O MySQL™ se apresenta como uma melhor opção para situações onde seja necessária interação entre placas ARDUINO UNO™, ou seja, uma versão mais simples da plataforma ARDUINO™, com um banco de dados, uma vez que é possível a utilização de um conector desenvolvido pela comunidade que propicia a

manipulação direta de dados entre a plataforma ARDUINO™ e o banco MySQL™ (MySQL Connector/Arduino, 2016).

Todavia, este é um recurso desnecessário para o presente projeto, pela utilização da plataforma RASPBERRY PI™, ou mesmo em se utilizando a ARDUINO Yún™, já que ambas podem se comunicar plenamente com qualquer banco de dados por meio de desenvolvimento de software utilizando linguagens de programação de alto nível (Eweek, 2010).

Com isso, conclui-se que a escolha do banco de dados requer uma análise do que se espera como resultado final para o sistema que está sendo planejado, o *hardware* que se tem disponível, a avaliação das ferramentas disponíveis para o desenvolvimento, os recursos que este banco de dados oferece em questão de agilidade, integridade, recuperação de dados, entre outros. Outro fator importante é a afinidade do desenvolvedor em sua utilização, garantindo código mais voltado a utilizar os recursos nativos do SGBD a favor da aplicação no sentido de ganho de desempenho e organização (Eric e Eric, 2002).

### **3.5.2. ADMINISTRAÇÃO REMOTA**

Uma das características que melhor definem, ou ajudam a definir, a qualidade de um serviço ou sistema atualmente é sua disponibilidade. É claro que outros fatores como confiabilidade, eficiência, utilidade e a capacidade de resolução de problemas, entre outros, sempre são fatores que agregam valor ao *software* e o tornam mais necessário e importante na vida das pessoas que procuram soluções mais eficientes para seus problemas, porém, atualmente e mais do que nunca, a disponibilidade do *software* é um diferencial que pode ser a chave de sucesso em qualquer ramo dentro do desenvolvimento de sistemas (Medvidovic *et al.*, 2010).

Com a disponibilidade pensa-se logo na mobilidade das aplicações, todavia, é claro que esse tipo de acesso remoto envolve a transmissão de dados, também de maneira remota, o que demanda uma preocupação ainda maior com a segurança das informações que trafegarão entre as localidades (Garrigues *et al.*, 2010; Wermelinger e Bandara, 2010).



Ocorre que existem essencialmente dois tipos de mobilidade ao se tratar de recursos computacionais, sendo elas a mobilidade de *hardware* e a mobilidade lógica. A primeira se refere à portabilidade de dispositivos, como um autofalante, por exemplo, que pode ser portátil e reproduzir sons onde quer que esteja, mesmo que em relação ao sistema no qual ele se encontra, como um *smartphone* ou *tablet*, o mesmo não seja considerado móvel, o sendo o conjunto do dispositivo em si (Wermelinger e Bandara, 2010; Charland *et al.*, 2011; De Souza *et al.*, 2011).

O problema se concentra no segundo tipo de mobilidade, e esta envolve a execução de códigos externos ao hardware, o que possibilita ataques por vírus de computador e o comprometimento da segurança das informações, ou a utilização de vulnerabilidades de segurança dos serviços disponíveis com a finalidade de adquirir acesso a informações confidenciais, fatos que têm inviabilizado a utilização de dispositivos móveis em grande escala na indústria e exigido que soluções proprietárias de segurança sejam desenvolvidas (Garrigues *et al.*, 2010; Wermelinger e Bandara, 2010).

Essas soluções, por sua vez, devem ser compatíveis preferencialmente com todos os sistemas operacionais de dispositivos móveis, assim como de computadores *desktop* caso se deseje uma integração abrangente de recursos entre essas plataformas, de maneira a dificultar ou inviabilizar o desenvolvimento de soluções de baixo custo (Wermelinger e Bandara, 2010; Charland *et al.*, 2011; De Souza *et al.*, 2011).

Desta forma, se torna importante a discussão sobre qual a melhor estratégia, de maneira a levar sempre em consideração a questão do custo de implantação para o desenvolvimento de aplicações móveis, no sentido de se optar pelo investimento no desenvolvimento de aplicações nativas ou aplicações WEB (Wermelinger e Bandara, 2010; Charland *et al.*, 2011; De Souza *et al.*, 2011).

### **3.5.2.1. APLICAÇÕES MÓVEIS NATIVAS**

O desenvolvimento de software para dispositivos móveis depende, entre outras coisas, do SDK (*Software Development Kit*) disponível para a plataforma-alvo,

ou seja, do conjunto de ferramentas disponibilizadas pelo desenvolvedor do sistema operacional do dispositivo móvel, para que o programador possa atingir seus objetivos e desenvolver seu aplicativo (Charland *et al.*, 2011).

Grandes empresas, como as que desenvolvem aplicativos comerciais genéricos ou como parte de um pacote de produtos e serviços, desenvolvedores de jogos eletrônicos, ou qualquer tipo de *softwarehouse* que se proponha a comercializar um produto em forma de aplicativos para dispositivos móveis procuram produzir versões diferentes de seus *softwares* para que possam ser utilizados na maior quantidade possível de aparelhos e ambientes diferentes (Charland *et al.*, 2011).

O problema é que o desenvolvimento de aplicativos com a utilização de diferentes SDKs pode não ser facilmente portátil para outras plataformas, pois há diferenças estruturais, tanto lógicas como de hardware, que podem dificultar sua conversão (Garrigues *et al.*, 2010).

Em consequência, grande parte do código deve ser reescrita para que a aplicação funcione em diversos dispositivos diferentes, e essa necessidade é uma grande desvantagem que aumenta o custo e adiciona retrabalho no desenvolvimento do aplicativo (Garrigues *et al.*, 2010).

O problema tende a ficar ainda mais crítico se for pretendida a disponibilização do referido aplicativo em PCs, que já entre si possuem vários sistemas operacionais diferentes e maneiras diferentes de se desenvolver em cada um deles, mesmo que linguagens baseadas em máquinas virtuais tendam a minimizar esse problema, ao menos entre sistemas operacionais distintos para computadores *desktop*, já que há certo impacto em desempenho (Charland *et al.*, 2011).

Outro aspecto que dificulta o desenvolvimento de aplicativos nativos de baixo custo é a já citada necessidade de desenvolver soluções próprias de segurança, o que pode ser complexo, moroso, caro e nem sempre tão eficiente, o que aumenta, conseqüentemente, o custo do projeto por se tratar de um tema complexo que

envolve estudo aprofundado de métodos de criptografia e segurança da informação (Garrigues *et al.*, 2010).

### 3.5.2.2. APLICAÇÕES WEB

Em contrapartida, existe algo em comum a todos os dispositivos, sejam eles *tablets*, *smartphones*, PCs, e em todos os sistemas operacionais: os navegadores de Internet. Basta a escolha de uma linguagem de programação como ASP.Net, PHP ou Java para que seja possível desenvolver o código de um sistema de informação ou automação para a Internet, que pode ser acessível em qualquer dispositivo desde que possua um *browser* (navegador) e acesso à Internet (Databaseandnetwork, 2002; Garrigues *et al.*, 2010).

Isso ocorre porque não há diferença de programação *server-side* para contemplar diferentes navegadores, apenas algumas adaptações mínimas relativas a *layout* (CSS) e código *client-side* (*javascript*, *vbscript*), que são muito mais simples de serem implementadas, em se comparando às diferenças entre os diversos SDKs para desenvolvimento de código nativo, já que existem padrões definidos para sites da *Internet* que os navegadores procuram contemplar em suas implementações, independentemente de seu desenvolvedor, definidos pela W3C (Databaseandnetwork, 2002; Garrigues *et al.*, 2010), e mesmo para o que não é padrodizado, é possível que se utilize um dos *bootstraps* disponíveis gratuitamente na *Internet*, o que torna a interface totalmente compatível independentemente de navegador.

Desse modo, a WEB pode funcionar como uma espécie de interface que integra todos os dispositivos, independentemente de marca, modelo, sistema operacional ou linguagem. É necessário apenas conectar-se ao site utilizando um navegador WEB qualquer e toda a informação estará disponível em qualquer dispositivo que tenha acesso à rede, desde que processamento *offline* não seja requerido na aplicação, como é o caso deste projeto (Databaseandnetwork, 2002; Garrigues *et al.*, 2010).

Conclui-se então que o desenvolvimento de uma ferramenta *WEB* (um site) acaba por ser mais interessante dos pontos de vista financeiro e de abrangência dos dispositivos que representam os alvos da aplicação, mesmo porque se pode contar com a utilização do protocolo HTTPS, mais seguro que o protocolo padrão da *Internet*, o HTTP, que conta com uma camada extra de segurança, o protocolo SSL/TLS, e mesmo não sendo totalmente seguro (assim como qualquer outra tecnologia), quando comparado com o resultado possível de ser obtido a partir do desenvolvimento de baixo orçamento para uma solução proprietária, oferece várias vantagens e um bom custo-benefício (Garrigues *et al.*, 2010).

#### **4. METODOLOGIA**

Dada importância do conjunto de diretrizes, consoantes em sua grande maioria com a ISO/IEC 9126 (norma ISO para qualidade de produto de software), que representam os princípios fundamentais deste projeto e por fazerem parte de seu objetivo, é necessária também uma breve explanação sobre como cada uma delas guiou o desenvolvimento deste trabalho.

O primeiro princípio adotado foi o de “baixo custo”. No entanto, é importante salientar que para o melhor desenvolvimento do projeto não há de se ter uma interpretação estritamente literal e isolada de cada um desses princípios e, sim, um balanceamento entre eles e sua interpretação em sentido amplo.

Assim, o “baixo custo” que fora referido acima significa que há de se escolher as opções menos onerosas financeiramente dentre as disponíveis e descobertas mediante pesquisa, que atendam aos requisitos do sistema, todavia, sem abrir mão da “segurança”, da “simplicidade” e da “funcionalidade”, por exemplo.

Por isso, é importante encontrar a solução menos onerosa possível, que ofereça um satisfatório nível de atendimento em comparação aos demais princípios estabelecidos, e não o menor custo total em termos brutos.

“Funcionalidade”, por sua vez, foi elencada para traduzir a necessidade do sistema de atender às expectativas do usuário ao entregar a solução a que se

propõe da melhor maneira possível, ao satisfazer as necessidades de interação de cada um dos usuários com sua cultura, maximizar os resultados do plantio, economizar recursos e oferecer melhor resultado final.

“Simplicidade” se refere à interação do usuário com a ferramenta que será oferecida a ele durante a utilização do sistema. Essa interação tem que ser feita de maneira simples, autoexplicativa e pouco burocrática, retornando ao usuário apenas as informações que ele necessita, e com a maior agilidade possível, escondendo ou minimizando qualquer nível de complexidade estrutural ou lógica que possa existir no sistema.

Enquanto a “simplicidade” se direciona mais especificamente à interação do usuário com o produto final, a “facilidade” vai de encontro mais diretamente à parte lógica e estrutural do sistema, ou seja, se algo pode ser implementado ou desenvolvido de duas maneiras, a escolha deve pesar em favor da menos complexa e mais direta, respeitando o balanceamento entre os outros princípios.

Já a “segurança” é algo fundamental em todo sistema. É sabido que não existe sistema totalmente seguro, principalmente os mais úteis ao usuário, que são justamente os serviços publicados na *Internet*, como é o caso deste trabalho. Todavia, é necessário que exista uma preocupação em nível de segurança que dificulte ações de pessoas mal-intencionadas, mas ela não pode limitar as funcionalidades do sistema, assim como não deve aumentar muito a complexidade de sua implementação e, principalmente, não influenciar negativamente a experiência do usuário com o sistema, novamente havendo a necessidade de balanceamento entre os princípios listados.

Por fim, a “escalabilidade” se refere à capacidade do sistema de se adequar a novas realidades de acordo com a demanda, ou seja, exemplificando no contexto do atual trabalho, seria a possibilidade, por exemplo, de se adicionar novos e diferentes sensores, que não estavam programados no início para serem utilizados, permitir sua calibração, ou até mesmo adicionar futuramente recursos multimídia para acompanhamento em tempo real da cultura por vídeo, se a largura de banda da conexão permitir.

#### 4.1. PLATAFORMA DE DESENVOLVIMENTO

O projeto foi desenvolvido utilizando a plataforma de desenvolvimento RASPBERRY PI™ modelo B+, embora para a realização de testes de mesa e simulações tenha sido utilizada uma placa ARDUINO UNO™ por possuir suporte a *input* analógico em seus pinos GPIO, o que possibilitou a ligação de um potenciômetro utilizado para simular diferentes situações de leituras que podem ocorrer quando se utiliza sensores, de maneira a fornecer maior controle de *input*, garantindo que várias situações pudessem ser testadas, o que seria difícil com o uso direto de sensores.

Esta placa ARDUINO™ por sua vez foi conectada diretamente à RASPBERRY PI™, em uma de suas portas USB e, enquanto simulando os sensores, teve sua comunicação estabelecida com a RPi de maneira serial, para o envio de comandos ao dispositivo hidráulico simulado e leitura de dados simulados sobre o solo.

#### 4.2. BANCO DE DADOS

Foi utilizado o banco de dados PostgreSQL™, em sua versão mais recente à data do fechamento do trabalho (a versão 9.5.1), para armazenar os dados que gerarão relatórios de uso do sistema, tanto na placa RPi quanto em um PC que foi utilizado para simular um servidor WEB para hospedagem do site.

#### 4.3. LINGUAGEM DE PROGRAMAÇÃO

O desenvolvimento do projeto foi feito utilizando a linguagem de programação PHP para a parte WEB, e para o tratamento das informações pela própria RPi foi utilizada a linguagem *Python*.

Foi utilizada também uma derivação da linguagem C++, padrão da IDE oficial da plataforma ARDUINO™ para programação de funcionalidades de teste no referido microcontrolador.

#### 4.4. SERVIDOR WEB

Um PC (*Personal Computer*) comum foi utilizado para as simulações como servidor WEB, rodando o sistema operacional Windows™ em sua versão 8.1, o serviço Apache™, em sua última versão (v2.5) para hospedar a versão de testes do *website*, além do banco de dados PostgreSQL™ em sua última versão (v9.5.1).

O PC em questão foi equipado basicamente da seguinte maneira:

- Processador Intel Core I5 4670k 3.4ghz Haswel Bx80646i54670k;
- Memória Corsair Vengeance 1600mhz C9 Box (2x4GB);
- Placa mãe Asus TUF Sabertooth Z87;
- Placa de Vídeo XFX Radeon R9 280x, 3gb, 384-bit GDDR5;
- SSD Corsair Neutron GTX F240, 240Gb, Sata 3.

#### 4.5. FERRAMENTAS DE DESENVOLVIMENTO

Ao desenvolver o código, tanto PHP quanto *Python*, foi utilizado o editor Notepad++ (Notepad++, 2016), e para a realização de simulações com a plataforma ARDUINO™ foi utilizada sua própria ARDUINOIDE (Arduino, 2016) com a finalidade de gravação do código em seu microcontrolador.

Na placa RPi foram instalados e configurados os seguintes componentes:

- Sistema operacional Raspbian (versão March 2016);
- PostgreSQL™ (versão 9.5.1);
- Psycopg2 (implementação para acesso ao banco PostgreSQL pela linguagem *Python*);
- TightVNC Server (para acesso remoto ao dispositivo);
- Adaptador Wireless USB Alfa AWUS036H.

#### 4.6. ETAPAS DE DESENVOLVIMENTO

Para realizar o desenvolvimento do sistema proposto nesse trabalho, foi seguido um roteiro que norteou os processos necessários, sendo estes os que seguem:

1. Análise de requisitos de sistema para identificar as entradas e as saídas do processo e mapear o funcionamento e arquitetura do projeto;
2. Levantamento de referências bibliográficas para embasamento teórico acerca dos trabalhos a serem realizados, e estudo sobre as tecnologias existentes que poderiam colaborar para o funcionamento do sistema;
3. Complementação dos requisitos do sistema de acordo com as informações levantadas com a leitura de artigos científicos acerca do tema, refinando a estrutura originalmente planejada com melhorias, inclusive remodelando o projeto para diminuir a complexidade e melhorar sua relação de custo-benefício, aproximando-o ainda mais do objetivo de baixo custo, inclusive abandonando a ideia inicial da utilização exclusiva da plataforma ARDUINO™, substituindo-a pela RASPBERRY PI™;
4. Montagem de uma máquina virtual rodando o sistema operacional Linux™, em sua distribuição Ubuntu™ (versão 15.10), e sua configuração para simular o ambiente de desenvolvimento que seria encontrado na RASPBERRY PI™;
5. Modelagem dos bancos de dados, tanto o local da RPi (provisoriamente representada por uma máquina virtual) quanto o geral, localizado no servidor WEB, de acordo com o levantado na análise inicial de requisitos;
6. Desenvolvimento de um código em *Python*, capaz de realizar leitura de sensores e ativação de bomba segundo parâmetros levantados durante a fase inicial (levantamento de requisitos), sendo que nesse momento por força da simulação da RPi por meio de uma máquina virtual, a leitura dos sensores foi feita provisoriamente gerando valores aleatórios ao invés de uma leitura real de sensores, assim como a



ativação da bomba do pivô central também sendo apenas representada por uma mensagem de ativação na tela, porém, já com a capacidade de se comunicar com o banco de dados do servidor WEB;

7. Desenvolvimento da interface WEB, ou seja, o *website* de maneira a ser capaz de ler as informações geradas aleatoriamente pelo programa de testes desenvolvido na RPi e interagir com os parâmetros que guiam as ações automáticas desse dispositivo, assim como permitir manipulação em tempo real da lógica de ativação da bomba;
8. Aquisição do material necessário para testes incluindo:
  - Placa RASPBERRY PI™ modelo B+;
  - Fonte de alimentação 5V, 2000mA (micro USB para uso com a RPi);
  - Placa ARDUINO™ para testes;
  - Adaptador Wireless USB Alfa AWUS036H;
  - Protoboard;
  - Materiais relacionados à eletrônica, como fios, leds, capacitores e potenciômetros.
9. Montagem de um circuito lógico utilizando os materiais adquiridos para simular de maneira mais próxima do mundo real, fazendo adaptações na parte de leitura de dados dos sensores e manipulação da bomba para leitura dos valores de um potenciômetro e manipulação de um led, respectivamente;
10. Testes gerais, correções e aprimoramentos;
11. Validação dos resultados obtidos em comparação com os objetivos iniciais do projeto, e verificação do grau de atendimento do mesmo segundo levantado como requisitos do sistema na etapa inicial do processo;
12. Desenvolvimento deste documento reunindo todas as informações e processos anteriormente trabalhados.

## 4.7. O SISTEMA

### 4.7.1. REQUISITOS

Assim como todo e qualquer sistema computacional a ser desenvolvido, a necessidade de levantamento de requisitos foi flagrante neste trabalho, e agora começará a ser detalhado qual o processo e diretrizes estabelecidas para a realização dessa etapa tão importante no desenvolvimento de um *software*.

Com a análise do problema a ser solucionado, além de pesquisas sobre a utilização das ferramentas escolhidas e leitura de artigos, foi possível chegar à seguinte lista de requisitos para o sistema:

- O sistema deve analisar as informações provenientes das leituras dos sensores de teor de água no solo e, com base nessas leituras, controlar o funcionamento (ligamento e desligamento) do mecanismo que controla o pivô central, colocando-o ou retirando-o de operação da seguinte maneira:
  - Caso o teor de água se encontre menor que o mínimo estabelecido como parâmetro pelo administrador do referido pivô, ele deve ser ATIVADO para que comece a irrigação até o teor de água superar o mínimo parametrizado;
  - Caso o teor de água se encontre maior que o máximo estabelecido como parâmetro pelo administrador do referido pivô, ele deve ser DESATIVADO a fim de evitar excesso de irrigação na cultura, prejudicando os resultados do plantio;
  - Deve ser possível configurar um tempo mínimo para que o pivô permaneça ativado mesmo após o teor de água atingir o mínimo parametrizado, para que não ocorra o ligamento e desligamento do mecanismo do pivô central com uma frequência muito alta, e para que também não se mantenha a cultura tão próxima do mínimo em relação ao teor de água a maior parte do tempo em um clima seco;
  - Se, por acaso, durante o tempo mínimo para que o pivô permaneça ativado mesmo após o teor de água atingir o

mínimo, o teor de água ultrapassar o parâmetro máximo definido no sistema, deve ocorrer a desativação do pivô central, mesmo antes de passado o tempo estabelecido.

- O sistema deve guardar em seu banco de dados local informações como data e hora, parâmetro mínimo de teor de água permitido, parâmetro máximo de teor de água permitido, teor de água atual, entre outras, todas as vezes que houver mudança de status no dispositivo de irrigação, ou seja, todas as vezes que ele for ativado ou desativado pelo sistema;
- Em um intervalo de tempo estabelecido parametricamente pelo administrador do pivô central, as informações salvas localmente na RPi que controla o referido pivô devem ser replicadas ao banco de dados WEB para maior segurança das informações, além de possibilitar consultas históricas sobre o funcionamento do equipamento *on-line*, por este sistema WEB;
- Deve ser possível o controle de frequência para a verificação das condições do solo para tomada de decisão pelo sistema automático, ou seja, deve ser possível por meio de parâmetros de sistema definir o intervalo de tempo entre uma verificação do teor de água do solo, e a próxima, para (por exemplo) 1 minuto, 30 segundos, 1 segundo ou o mais rápido possível de acordo com as capacidades do *hardware* da RPi vinculada àquele pivô central. Essa frequência deve ser informada em segundos no *website*;
- Deve ser possível, por meio de parâmetros, informar à RPi associada ao pivô central quais os valores mínimo e máximo entre os quais o teor de água do solo deve se manter para que a RPi tome as decisões automaticamente de modo a garantir que a irrigação ocorra no sentido de manter o teor de água entre esses valores mínimos e máximos a todo momento.
- Deve ser possível, por meio do sistema, cadastrar agendamentos, tanto para dias fixos da semana, por exemplo, “todas as segundas-feiras das 10:00 às 13:00”, quanto para datas específicas, por exemplo “dia 31/12/2015 das 10:00 às 13:00”, determinando os horários em que

o pivô deve ser ativado independente de o teor de água já estar entre o mínimo e máximo parametrizados no sistema para casos específicos como fertirrigação, por exemplo;

- Deve ser interrompida a aspersão por agendamento de maneira automática caso o teor de água passe a ser superior ao máximo parametrizado no sistema, mesmo dentro do período agendado.
- Deve ser possível mudar o sistema para modo MANUAL, ou seja, desativar todas as ações automáticas por parte da RPi vinculada ao pivô central selecionado para que os comandos de ativação e desativação sejam feitos exclusivamente pela WEB, de maneira remota, pelo usuário que tem permissões sobre aquele pivô central;
- Deve ser possível definir um intervalo de tempo de segurança entre o último comando enviado de maneira manual por um usuário administrador do pivô central pela WEB até o momento em que a RPi deverá entrar em modo AUTOMÁTICO novamente, mesmo sem a solicitação do usuário, para que não se corra o risco de, por exemplo, quando em modo automático o usuário ativar a aspersão e, posteriormente, por algum problema com sua Internet perder a conexão com o site, e não conseguir mais desativá-la quando deveria. Dessa forma, se parametrizados dez minutos, por exemplo, após a última interação manual, em dez minutos a RPi voltará a trabalhar automaticamente, avaliando as condições do solo e ativando ou desativando o pivô conforme o necessário;
- Deve ser possível impedir que o recurso acima seja processado, forçando o sistema a nunca voltar automaticamente ao modo automático quando em modo manual, independentemente do tempo após o último comando manual recebido (não recomendado, mas possível);
- Deve ser possível, por meio do sistema WEB, monitorar em tempo real informações relativas ao pivô central, tais como:
  - Teor de água atual do solo;
  - Situação do pivô central no momento (ativado ou desativado);
  - Data e hora do próximo desligamento previsto;

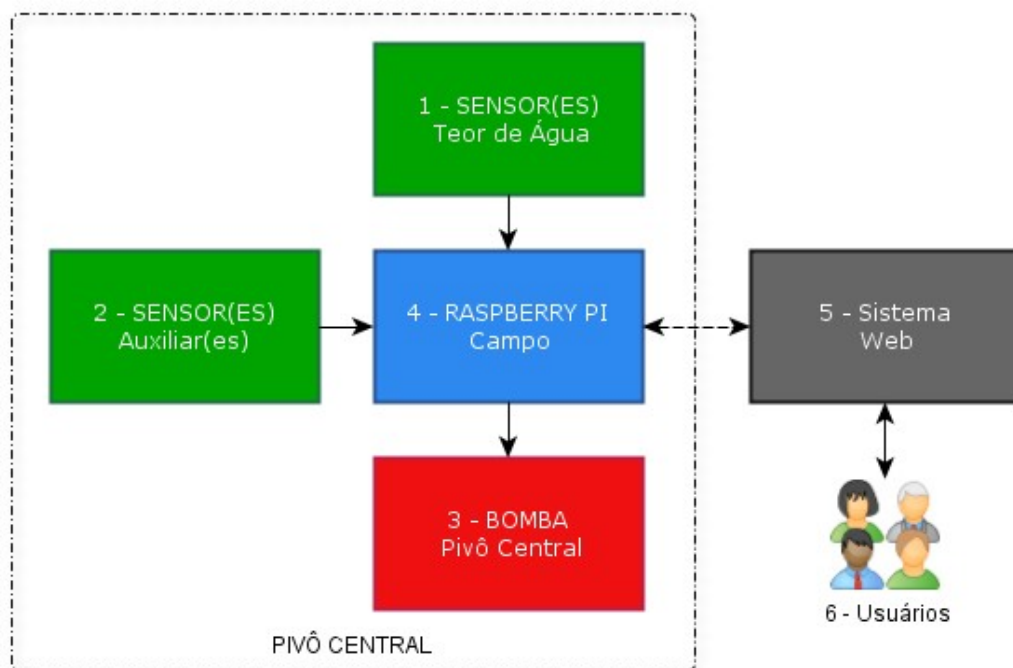
- Data e hora da última atualização de dados no sistema WEB de maneira automática pelo dispositivo RPi vinculado ao pivô central selecionado;
  - Se o sistema se encontra em modo automático ou manual;
  - Quanto tempo falta para que o sistema saia do modo manual automaticamente.
- Deve ser possível forçar a atualização dos dados WEB, independentemente de não estar no horário em que a RPi o faria automaticamente;
  - Deve ser possível informar de quanto em quanto tempo o banco de dados WEB deve ser atualizado automaticamente por parte do RPi que controla o pivô selecionado;
  - Deve contemplar a manipulação de vários pivôs centrais, possibilitando a seleção de com qual se quer trabalhar no momento;
  - Deve conter controle de usuários, com nome de usuário e senha, para dificultar o acesso de pessoas não autorizadas às funcionalidades do sistema;
  - Deve permitir associação de usuários a pivôs centrais, de maneira a determinados usuários só poderem visualizar na lista dos pivôs selecionáveis os quais ele deve ter permissão para visualizar, garantindo assim o acesso de vários usuários ao mesmo pivô, e de vários pivôs ao mesmo usuário, mas desde que exista tal associação cadastrada. Assim, o sistema pode controlar pivôs de várias culturas, de diferentes proprietários em um só sistema;
  - Em caso de mais de um sensor para teor de água ser utilizado em um determinado pivô, trabalhar a informação do teor de água como sendo a média entre as leituras realizadas por cada um deles no momento;
  - Permitir vinculação de outros sensores que possam ser interessantes ao administrador da cultura, por exemplo, sensores de temperatura e umidade relativa do ar, além do sensor principal de teor de água no solo, que deve estar presente sempre para funcionamento do sistema, e armazenar as leituras de todos eles sempre que houver uma ativação ou desativação, manual ou automática do pivô central;

- Permitir a calibração dos sensores no sistema.

#### 4.7.2. A ESTRUTURA DO SISTEMA

A estrutura geral do sistema está representada pela Figura 7.

**Figura 7:** Estrutura do sistema utilizando RASPBerry PI™



**Fonte:** Autor, 2015.

Desta forma, o dispositivo RASPBerry PI™ é suficiente para controlar a ativação ou desativação do mecanismo do pivô central, sendo que sensores podem ser conectados diretamente a este dispositivo, que, ao receber as leituras e os comandos do usuário via WEB, decidirá se o pivô central deve ser ligado ou desligado e enviará as devidas instruções ao mecanismo de irrigação.

No mundo real, a depender da cultura e das preferências do agricultor, diferentes tipos de sensores podem ser necessários. Com isso devem ser feitas adaptações lógicas na RASPBerry PI™ para contemplar estes diferentes tipos de *input* de dados por parte dos sensores, caso a caso.

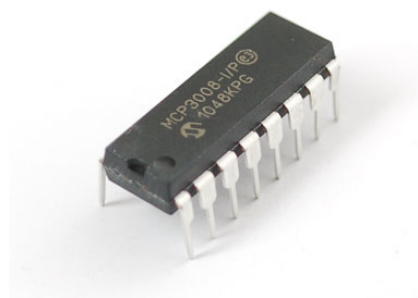
Assim, a aquisição de informações por parte dos sensores não pode ser fixada de maneira padrão no *software*, necessitando a implementação e configuração específicas na instalação do equipamento, de posse do *hardware* que fará a leitura para a realização dos devidos testes.

Deve ser possível, porém, definir padrões relativos aos sensores e quais medições seriam realizadas (temperatura, umidade relativa do ar, teor de água no solo, etc), mas é importante garantir a escalabilidade do *software* mantendo abertas as opções de sensores e de tipos de medição, e daí surge a necessidade de generalização do *software* em sua versão final, antes da implantação desse sistema em um dispositivo do mundo real, já que não existe essa necessidade no modelo de testes proposto.

O maior problema se dá pela falta de *input* analógico nos pinos GPIO da RASPBERRY PI™. Muitos sensores utilizam uma configuração de três pinos, sendo o *GROUND*, o 3.3/5V, e o responsável por enviar os dados da leitura, geralmente de maneira analógica.

Para suportar a utilização desse tipo de sensor, é necessária a utilização de um conversor analógico-digital como o MCP3008 (Figura 8).

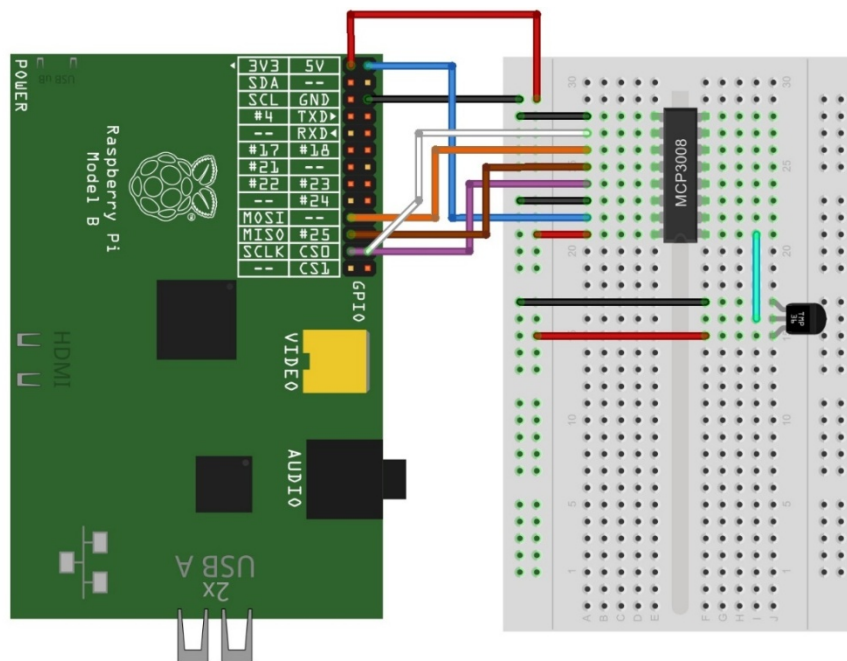
**Figura 8:** Conversor analógico-digital MCP3008



**Fonte:** Autor, 2015.

Com a utilização do referido conversor, é possível para a RASPBERRY PI™ recuperar dados de sensores que tem *output* analógico, como é mostrado no exemplo a que se refere a Figura 9.

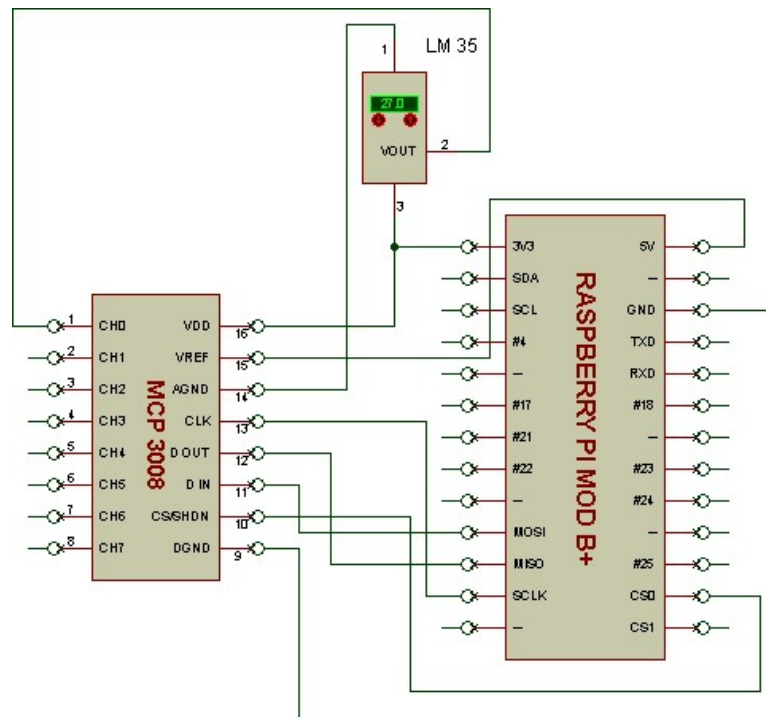
**Figura 9:** Esquema RASPBERRY PI com conversor MCP3008



**Fonte:** Autor, 2015.

O esquema elétrico desse circuito pode ser representado pela Figura 10.

**Figura 10:** Esquema Elétrico (RPI e Conversor MCP3008)



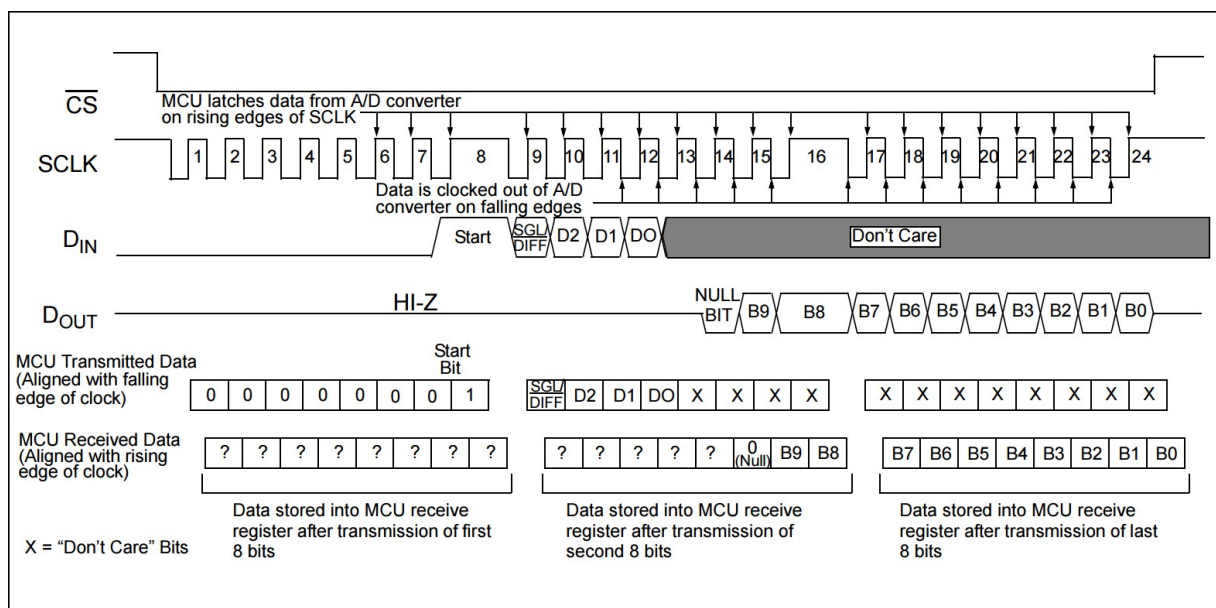
**Fonte:** Autor, 2015.



Outros sensores não necessitam da utilização do recurso acima referido por possuírem *output* digital, e podem ser ligados diretamente aos pinos GPIO da RASPBERRY PI™, porém, assim como os diferentes tipos de conversor analógico-digitais que existem no mercado, há também diferentes implementações que devem ser feitas para que a RPi consiga interpretar os dados binários provenientes do sensor, quando em modo digital.

Um sensor se comunicando pelos pinos GPIO em modo digital com a RPi funciona da mesma maneira que um sensor analógico, ou um potenciômetro, por meio de um conversor analógico-digital, ou seja, são enviados grupos de *bits* de informação de forma sequencial como *output*, em direção à RPi, sendo que no caso do conversor MCP3008 (do exemplo acima) a comunicação é feita da forma descrita na Figura 11, retirada do manual do componente.

**Figura 11:** Comunicação SPI com MCP3004/3008 utilizando segmentos de 8 bits



**Fonte:** Manual MCP3004/3008.

Inferese, a partir da Figura 11, que a comunicação entre o conversor e o microcontrolador é feita pelo protocolo SPI (*Serial Peripheral Interface*), que se baseia em comunicação por grupos de bits entre dispositivos categorizados como “mestre” (RPi) e “escravo” (conversor).

O dispositivo mestre configura o *clock*, usando uma frequência de alguns MHz que o dispositivo escravo suporta, e durante cada ciclo SPI há comunicação *full-duplex* (de duas vias), pelas portas MOSI (mestre) e MISO (escravo).

De maneira simples, e voltando para o exemplo proposto, a RPi deve enviar um conjunto de três palavras de 8 bits cada uma, sendo que a primeira é composta de 7 (sete) *leading zeroes* (zeros à esquerda) seguidos de 1, considerado o *start bit*, ou seja, o que basicamente fará com que o dispositivo escravo (conversor) tenha conhecimento de que está sendo feita uma requisição pela RPi.

Esta requisição é do tipo indicado no primeiro bit da segunda palavra, e seu valor pode ser 1, para *single-ended* ou 0 para *differential*.

Não é a intenção o grande aprofundamento neste tema, mas basicamente a diferença entre *single-ended* e *differential* é que a medição *single-ended* ocorre com a diferença de voltagem entre um fio e o *ground*. Desse modo, o ruído é medido juntamente com a voltagem de saída do sensor.

Já o modo *differential*, com o qual trabalham alguns sensores, é flutuante, ou seja, funciona medindo a diferença de voltagem entre dois fios, não havendo referência ao *ground*. Sensores que utilizam este modo diferencial podem trabalhar no modo *single-ended* quando tendo seu lado *low* ligado ao *ground*, reduzindo o número de canais necessários para que se faça a comunicação com o mesmo.

A depender de qual tipo de sensor está sendo usado, informa-se no primeiro bit da segunda palavra a se enviar ao conversor, seguido de qual canal se espera retorno, entre CH0 e CH7, tendo seu código binário (de 0 a 7, ou de 000 a 111 em binário) ocupando os próximos 3 (três) bits.

As demais posições podem receber qualquer valor, uma vez que serão ignoradas pelo dispositivo escravo (por este motivo estão representadas com um “X” na Figura 11).

Ao enviar para o conversor 3 (três) conjuntos de 8 (oito) bits, a resposta do conversor também segue a mesma configuração, iniciando nos 3 (três) últimos bits da segunda palavra, sendo que o primeiro desses bits é sempre 0 (zero), e os outros

dois, assim como todos os bits da terceira palavra formam juntos um conjunto de 10 (dez) bits, que é a resolução do dispositivo.

A resolução de 10 bits significa que a quantidade de valores possíveis para a leitura das informações é de  $2^{10}$  valores diferentes, ou seja, entre 0 até 1023, quando convertidos em números inteiros sem sinal (ou seja, *unsigned integers*), deste modo, a porcentagem que representa a leitura do sensor pode ser adquirida a partir de uma simples conversão deste valor de retorno para a escala de 0 a 100.

Dependendo do dispositivo a ser utilizado, seja ele um conversor analógico-digital ou um sensor com saída digital, é necessária uma implementação manual na lógica de leitura da RPi para que ela entenda os valores de leitura (assim como para que ela saiba como solicitá-los ao sensor).

Há, todavia, uma maneira ainda mais simples de alcançar os mesmos resultados descritos no exemplo anterior, que envolve fazer com que a RPi trabalhe em conjunto com uma placa ARDUINO™, substituindo o conversor analógico-digital por ela, uma vez que a mesma possui suporte para entradas analógicas.

Para isso bastaria conectar uma placa como a ARDUINO UNO™ em uma das portas USB da RASPBERRY PI™ e realizar comunicação de maneira serial entre os dois dispositivos.

Assim, a estrutura elaborada para o sistema que foi desenvolvido, aliada à versatilidade da RASPBERRY PI™ faz com que as especificações relativas a *hardware* não influenciem em nada o funcionamento da parte WEB desse sistema, tornando-o compatível com qualquer arquitetura de *hardware* no tocante a sensores, integração com outras tecnologias como ARDUINO™, ou quaisquer recursos extras que podem ser adicionados mediante necessidades pontuais futuras, transformando essa parte de *hardware* em uma camada de complexidade transparente para a parte da aplicação WEB do projeto.

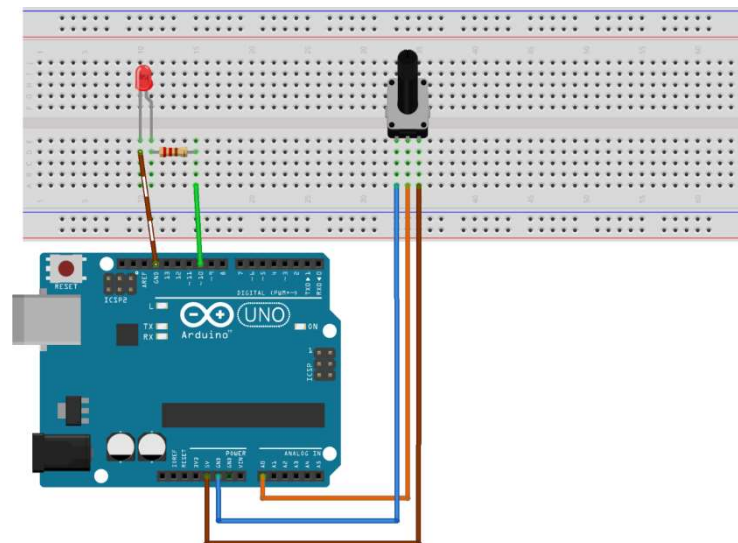
Dessa maneira, o sistema WEB não precisa ter conhecimento de como a RPi está informando os dados que ele necessita, desde que esteja recebendo todas as informações necessárias, e esteja conseguindo também enviar comandos a este dispositivo.

O propósito do sistema é justamente este: possuir uma parte WEB generalista, que possibilite os usuários a se conectarem com seus dispositivos e interajam com ele, porém tratar de maneira específica às necessidades de cada cultura oferecendo capacidades de customização máxima de *hardware*, possibilitando adaptação à maioria das necessidades da cultura.

Para desenvolvimento do software, optou-se por utilizar um conjunto entre RASPBERRY PI™ e ARDUINO UNO™, onde toda a interação GPIO é feita pela plataforma ARDUINO™ (embora a RPi seja plenamente capaz de atender a esta demanda), ligando as duas plataformas por conexão USB, com o intuito de exemplificar a flexibilidade do sistema em termos de *hardware*.

Para realização dos testes, a plataforma RASPBERRY PI™ além de ligada à ARDUINO UNO™ por conexão USB, também conta com um dispositivo WiFi para conexão sem fio com a Internet, e a plataforma ARDUINO™ foi montada da forma representada na Figura 12.

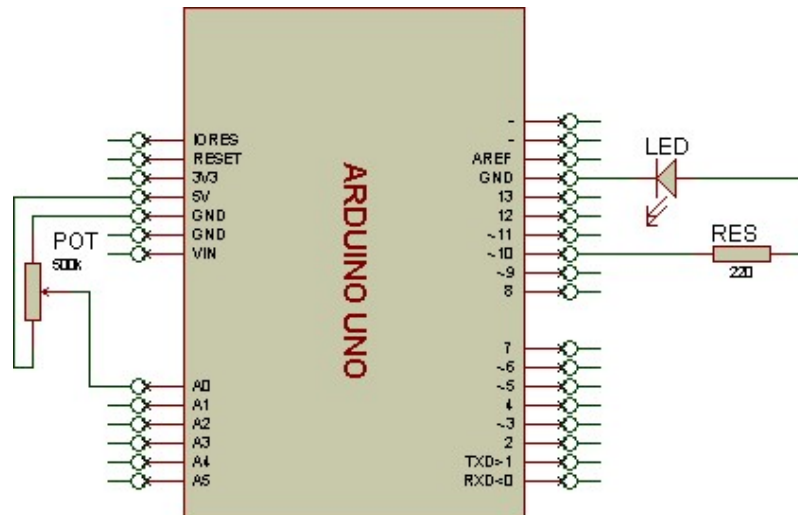
**Figura 12:** Estrutura para simulação de leituras



**Fonte:** Autor, 2015.

O esquema elétrico para a estrutura para simulação de leituras utilizada nos testes do software pode ser representado pela Figura 13.

**Figura 13:** Esquema Eletrônico (simulação)



**Fonte:** Autor, 2015.

Na estrutura representada pela figura acima, o LED representa o status do pivô central no que se refere a estar ou não em funcionamento (ligado ou desligado, respectivamente), e para simular sensores (ou grupos de sensores) de teor de água foi utilizado um potenciômetro (também como mostrado na figura acima).

Há razões para que se fosse utilizado o potenciômetro em vez de um sensor real, como o fato de que para o sistema WEB não é importante a origem dos dados, e sim que eles sejam gerados e informados.

Isso pode ser feito facilmente por um potenciômetro, que possibilita o recebimento ajustes finos de maneira manual, a fim de facilitar os testes em relação à utilização de um sensor de teor de água, por exemplo.

Trata-se então de uma vantagem do potenciômetro em relação aos sensores, em termos de testes, pois durante sua execução houve a necessidade de se detectar ou chegar a um teor específico para verificar se determinadas ações seriam tomadas pelo sistema quando isso ocorresse.

Seria muito mais complexo conseguir um teor de água específico por adição ou remoção de água em uma amostra de solo quando comparado a um simples ajuste no potenciômetro para que ele passe a retornar o valor necessário para que

se satisfaçam as condições necessárias para a realização do teste, e isso facilitou a simulação de ambientes mais específicos.

Da mesma maneira, foi utilizado um LED comum para representar o circuito secundário de ativação da bomba hidráulica, por ser a maneira mais usual de verificar se sinais estão sendo enviados apropriadamente.

No mundo real seria possível a utilização de um *relay shield* (Figura 14), em se tratando de ativação de dispositivos que necessitam de menos de 250VAC, sendo que o *shield* de exemplo funcionaria (mesmo requerendo uma entrada de 5V) a 3.3V, já que somente 3V são necessários para sua ativação.

**Figura 14:** Relay Shield SRD-05VDC-SL-C



**Fonte:** Autor, 2015.

O SRD-05VDC-SL-C (Figura 14) é o que se chama de um SPDT (*Single Pole Double Throw*) Relay, significando que o fio central (*pole*, ou COM) pode estar conectado a dois pinos diferentes, N.O. (*Normally Open*) ou N.C. (*Normally Closed*), dependendo de a bobina estar gerando ou não campo eletromagnético, respectivamente.

Este modelo de relé funciona com a voltagem de 5V e corrente de 60mA, e como saída suporta até 250VAC ou 30VDC, e corrente de 10A, mais que suficiente para ativar um motor comum, e como visto acima 3.3V (o fornecido pelo GPIO da RPi) são suficientes para sua utilização, uma vez que ele se ativa aos 3V.

Todavia, bombas hidráulicas geralmente trabalham em altas tensões (380 a 440V), o que inviabiliza o uso do *shield* referido acima, e faz necessária a efetiva montagem de um circuito secundário de ativação, de maneira a receber os 3.3V (ou

0V) de sinal da RPi e ser capaz de ativar ou desativar a bomba hidráulica por alimentação externa e, por consequência, o pivô central propriamente dito, além de preservar a integridade da RPi com a correta proteção de seus componentes contra possíveis danos provindos deste segundo circuito, que não é objeto deste trabalho.

### **4.7.3. O SOFTWARE**

O funcionamento básico do *software* se resume à interação com dois bancos de dados distintos, sendo o primeiro o próprio da plataforma WEB, que é alimentado por todos os dispositivos RPi que representam os diferentes pivôs centrais, o que o torna um repositório de dados de irrigação de todos os sistemas, e outro instalado em cada um dos dispositivos RPi que fazem parte do sistema. Suas estruturas serão detalhadas a seguir.

#### **4.7.3.1. SOFTWARE DA RASPBERRY PI™**

Cada placa RPi associada ao sistema representa um pivô central, e o controla diretamente por meio de software escrito na linguagem *Python*. Este software alimenta um banco de dados local com informações pertinentes às condições do solo (e, se necessário do ambiente) sempre que o dispositivo (o pivô central) é ativado ou desativado, além de alimentar periodicamente o banco de dados central localizado no servidor WEB e remover os dados locais no processo, por motivos de economia de espaço de armazenamento e desempenho.

O código *Python* que roda dentro das placas RPi consiste em um *loop*, ou laço lógico que é executado em um intervalo de segundos estabelecido por parâmetro na parte WEB do sistema, e faz as leituras dos sensores, controla o dispositivo de irrigação e armazena informações no banco de dados local da maneira representada pelo pseudocódigo da figura 15.

**Figura 15:** Pseudocódigo da lógica local dos dispositivos RASPBERRY PI™

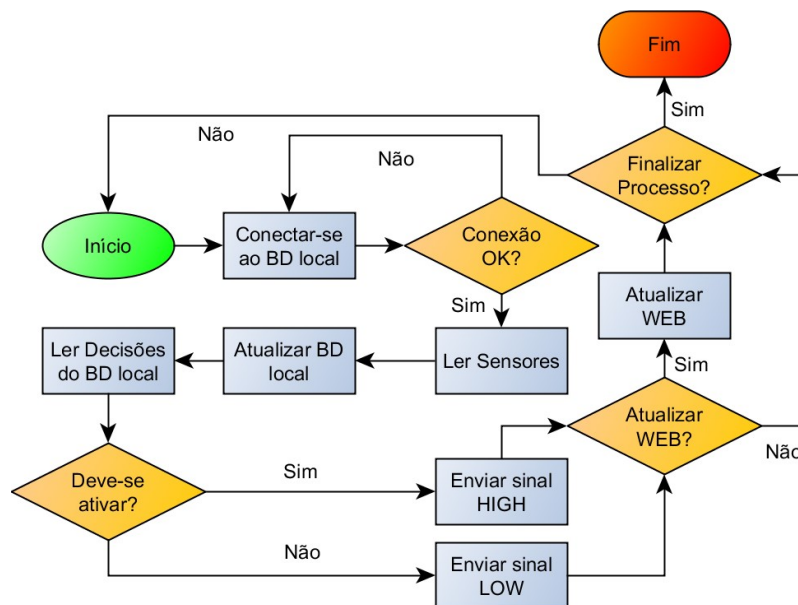
```

01 Início do Laço
02 Enquanto não houver conexão com o banco local:
03     Tentar conectar-se;
04
05 Realizar leitura dos sensores;
06 Atualizar o banco de dados local com as novas leituras;
07 Ler as decisões do banco de dados com base nas novas leituras;
08
09 Se decidido ativar a bomba hidráulica:
10     Enviar sinal HIGH para a bomba hidráulica;
11 Caso contrário:
12     Enviar sinal LOW para a bomba hidráulica;
13
14 Se já for o momento de atualizar o banco de dados WEB:
15     Atualizar o sistema WEB;
16
17 Aguardar até o próximo ciclo;
18 Fim do Laço

```

**Fonte:** Autor, 2015.

Para melhor visualização do processo, foi elaborada a figura 16 que representa seu fluxograma.

**Figura 16:** Fluxograma (RPI)

**Fonte:** Autor, 2015.

Como pode ser percebido a partir do pseudocódigo acima, as verificações, cálculos de tempo e tomada de decisões são tomadas diretamente pelo banco de



dados por meio de *triggers* durante a atualização descrita pela linha “06”, assunto que será desenvolvido a seguir.

Essas *triggers*, também chamadas de “gatilhos”, são lógicas que são executadas automaticamente pelo banco de dados a partir do acontecimento de um evento específico, como uma inserção, deleção ou atualização de dados.

A opção para utilização de programação diretamente no banco de dados foi pensada para que fosse possível manter a regra de negócios fixa diretamente no banco de dados, o que gera independência em relação à linguagem de programação e permite fácil implementação dos projetos em outras linguagens, e também pelo banco de dados possuir meios, funções específicas e tipos de dados nativos que tornam o tratamento de intervalos de tempo muito mais simples de serem implementados em comparação com o desenvolvimento manual desses recursos em outras linguagens.

Dessa maneira, todas as vezes que houver interação com a tabela que contém os parâmetros do sistema (tabela “params”), desde que exista alteração de dados (consultas não estão incluídas, assim como inserções e deleções), antes que essas alterações sejam efetivadas na tabela, são feitas validações (a serem vistas posteriormente) e alterações na solicitação de alteração de dados conforme necessário para o funcionamento do sistema.

Antes, porém, de entrar em detalhes sobre o funcionamento da *trigger* referida acima, torna-se conveniente observar em uma visão geral como foi modelado o pequeno banco de dados interno para os componentes RPi, de acordo com a Figura 17.



dessa bomba, em períodos ou dias específicos, por parte do usuário do sistema WEB.

Todavia, esta estrutura, em uma versão final do software, deve ser melhorada, contemplando entidades como agendamentos ou programações, que agrupariam os horários permitindo diferentes configurações de acordo com a cultura a ser irrigada em caso de rotatividade de culturas, ou outras facilidades inerentes ao mundo real que não se fazem presentes no modelo de simulação.

A tabela “params” é o que pode se chamar de “o coração” do sistema. Seu propósito é abrigar as mais diversas informações paramétricas, o que faz necessária a verificação de cada uma de suas colunas de maneira simplificada, porém, individual:

- Coluna “id\_local”: é a chave primária da tabela, e seu valor identifica a RPi no sistema WEB. Em outras palavras, o valor dessa coluna funciona como o código interno para aquele dispositivo para interação por parte do sistema WEB;
- Coluna “descrição”: Uma breve descrição da RPi, para fácil identificação da mesma no sistema. Pode conter qualquer informação que facilite seu reconhecimento no sistema WEB, como a propriedade em que se encontra o dispositivo, identificação do pivô central o qual ela controla, entre outros;
- Coluna “auto”: Controla se o sistema está em modo automático ou manual, sendo que em modo automático o sistema controla a bomba hidráulica sem a necessidade de interferência do usuário, já em modo manual o sistema sobrescreve seu comportamento automático com requisições recebidas direto do usuário;
- Coluna “port”: Porta em que o servidor do banco de dados está escutando por conexões. Como pode haver mais de uma RPi por propriedade rural a ser irrigada, assim como pode haver mais de um pivô central (pela relação ser de um para um entre dispositivos RPi e pivôs centrais) a maneira para o servidor WEB acessar o banco de dados da RPi correta será pelo seu número de porta, ou seja, o

sistema WEB se conectará pelo IP do roteador da propriedade (que deve estar configurado com *port forwarding* definindo acesso externo direto ao servidor de banco de dados das RPi) adicionado à porta especificada nesse campo. Em outras palavras, o IP do roteador será o mesmo para todas as RPi, supondo que seja 189.1.1.12, para se conectar na RPi que tenha “port” igual a 5000, o servidor WEB externo se conectará em 189.1.1.12:5000, assim como para se conectar em uma segunda RPi, por esse mesmo roteador, que tenha sua porta definida para 5151, a *string* de conexão para o sistema WEB será 189.1.1.12:5151, e assim por diante (esse valor estará também configurado no servidor WEB);

- Coluna “ligada”: Indica a situação atual da bomba hidráulica, sendo valor “*true*” para ligada e “*false*” para desligada. Este campo assim como outros é atualizado pela RPi em um intervalo pré definido de tempo;
- Coluna “umidade\_atual”: Indica o teor de água do solo na última medição, ou seja, o teor de água atual do solo, e é atualizado juntamente com a coluna “ligada” entre outros;
- Coluna “umidade\_mínima”: Indica qual é o parâmetro limítrofe inferior de teor de água a ser mantido, sendo que se o teor de água atual do solo for inferior ao valor parametrizado neste campo, a bomba hidráulica deve ser acionada;
- Coluna “umidade\_máxima”: Da mesma maneira que o parâmetro ou coluna anterior se refere ao limite superior, acima do qual o teor de água não pode estar. A bomba hidráulica é desligada automaticamente pelo sistema caso o teor de água ultrapasse o valor estabelecido por este parâmetro;
- Coluna “timeout\_manual”: Quantidade de tempo (em segundos) que o sistema deve aguardar para entrar novamente em modo automático após a última intervenção manual do usuário pelo sistema WEB (se o número zero for informado, o sistema nunca alternará automaticamente para o modo automático);

- Coluna “timeout\_bomba”: Quantidade de tempo (em segundos) que a bomba hidráulica deve permanecer ligada mesmo depois de atingido o mínimo parametrizado na coluna “umidade\_mínima”, a fim de evitar ligamento e desligamento frequente e desnecessário do equipamento, mas com a garantia de que se o teor de água atual ultrapassar o parâmetro “umidade\_máxima”, o dispositivo será desativado independentemente deste tempo;
- Coluna “atualizacao\_web”: Tempo em minutos entre a última atualização da base de dados WEB pelo conteúdo *offline* do banco de dados local da RPi e a próxima;
- Coluna “ultima\_ligacao”: Contém a informação de quando a bomba foi ligada pela última vez;
- Coluna “sleep\_time”: Tempo de descanso entre um laço e o próximo dentro da lógica em Python que roda dentro da RPi;
- Coluna “ultima\_alteracao\_manual”: Contém a informação de quando foi feita a última interação manual do usuário com a RPi pelo sistema WEB, e serve de parâmetro para calcular quando a RPi deve mudar automaticamente para modo automático após passado o tempo estabelecido na coluna “timeout\_manual” a partir dessa última interação manual;
- Coluna “usuario”: Controla qual usuário do sistema realizou a última alteração manual feita pelo sistema WEB diretamente na RPi;
- Coluna “server\_ip”: Endereço do servidor WEB para comunicação da RPi com o mesmo para atualização da base de dados geral.
- Coluna “server\_port”: Porta para comunicação com o servidor WEB;
- Coluna “server\_username”: Usuário para conexão com o servidor WEB;
- Coluna “server\_password”: Senha para a conexão com o servidor WEB;
- Coluna “forçar\_atualizacao”: Se “true” significa que a RPi deve atualizar o sistema WEB imediatamente.

Cada vez que a bomba hidráulica é ativada ou desativada, informações do solo e de parâmetros são salvas no banco de dados local da RPi, para posterior atualização da base de dados central no sistema WEB. A tabela "HISTORICO" foi intencionalmente criada sem acentuação para maior facilidade de desenvolvimento de código, e suas colunas são as seguintes:

- Coluna "data\_hora": Indica quando ocorreu aquela ação (ligar ou desligar a bomba hidráulica), e também é a chave primária da tabela;
- Coluna "ligada": Indica se a bomba hidráulica foi ligada (*true*) ou desligada (*false*) na ocasião;
- Coluna "umidade\_atual": Informa qual era a leitura para teor de água do solo no momento em que houve a ativação ou desativação da bomba hidráulica;
- Coluna "umidade\_minima": Informa qual era o valor mínimo estabelecido como parâmetro no evento da ativação ou desativação da bomba hidráulica;
- Coluna "umidade\_maxia": Informa qual era o valor máximo estabelecido como parâmetro no evento da ativação ou desativação da bomba hidráulica;
- Coluna "usuario": Informa qual usuário realizou a ativação ou desativação da bomba hidráulica em caso de interação manual pelo sistema WEB. Se a interação foi automática pela RPi, a coluna recebe o valor "*null*".

Como os testes com o sistema foram realizados em caráter de simulação, com a utilização de um potenciômetro para representar o sensor de teor de água do solo, a única informação a ser salva na tabela de históricos é a própria leitura correspondente ao que seria o teor de água do solo no mundo real, ou seja, o campo "umidade\_atual".

Porém, em funcionamento no mundo real, o dispositivo RPi pode estar conectado a vários outros tipos de sensores, como de temperatura, umidade relativa do ar, luz, ou o que mais for necessário para futura geração de dados para estudo pelo responsável pela cultura irrigada pelo sistema.

Estes outros sensores teriam suas leituras gravadas no evento de ativação ou desativação da bomba hidráulica da mesma maneira que o teor de água do solo está sendo gravado na coluna “umidade\_atual”, com a adição de novas colunas para os novos tipos de sensor, obviamente.

É importante para o sistema a possibilidade de programação manual de em quais horários e dias a bomba hidráulica deve ser ligada, independentemente das leituras mínimas de umidade. Dessa forma, se a cultura exige irrigação ou ainda fertirrigação do solo três vezes por semana, é possível programar no sistema que em todas as segundas, quartas, e sextas-feiras, por exemplo, ou terças, quintas e sábados, para que a bomba hidráulica seja ativada independentemente do teor de água atual se encontrar inferior ao teor de água mínimo padronizado no sistema, por um dado intervalo de tempo.

Assim, também pode ser necessário especificar uma data e hora para que a aspersão se inicie, e outra para que ela se finalize de maneira pontual e extraordinária.

A tabela responsável por parametrizar estes comandos é a tabela “horarios” que permite fazer esse tipo de programação, lembrando que mesmo que esteja dentro de uma janela de horários em que a bomba hidráulica deveria estar ligada, se o teor de água do solo chegar a ser superior ao máximo parametrizado no sistema em algum momento, essa bomba hidráulica é desligada automaticamente pelo sistema.

A tabela "HORARIOS" foi intencionalmente criada sem acentuação para maior facilidade de desenvolvimento de código, e suas colunas serão explicadas de maneira mais detalhada abaixo:

- Coluna “horario”: Identificador numérico para o horário no sistema local da RPi;
- Coluna “dia”: Valor de 0 a 6, indicando o dia da semana ao qual a programação se refere, obedecendo à associação abaixo (caso o campo não tenha sido preenchido, significa que se trata de uma data específica, e não um dia da semana):

- 0: Domingo;
  - 1: Segunda-Feira;
  - 2: Terça-Feira;
  - 3: Quarta-Feira;
  - 4: Quinta-Feira;
  - 5: Sexta-Feira;
  - 6: Sábado;
- Coluna “inicio”: Horário, se for uma programação de dia da semana, ou data e hora se for uma programação para uma data específica, de ativação da bomba hidráulica;
  - Coluna “fim”: Horário, se for uma programação de dia da semana, ou data e hora se for uma programação para uma data específica, de desativação da bomba hidráulica;
  - Coluna “ativo”: É possível excluir programações previamente cadastradas, e é possível também simplesmente desativá-la no sistema, ou seja, ela continuará cadastrada, mas inativa, esperando sua reativação futura se conveniente. Os valores são “*false*” para “inativa” ou “*true*” para “ativa”;
  - Coluna “nome”: Associa um nome ao agendamento.

A linha 6 do pseudocódigo apresentado na Figura 15 atualiza o banco de dados local com os valores mínimo, máximo (parâmetros) e atual (leitura dos sensores). Existe uma *trigger* no banco de dados para possibilitar, sempre que houver modificação do banco de dados pela RPI, diversas validações e tomadas de decisão, seguindo o pseudocódigo da Figura 18.

Dessa maneira, as verificações e tomadas de decisão não foram feitas no código *Python* que roda em *loop* durante todo o funcionamento da RPI, que tem como principal objetivo atualizar a coluna “umidade\_atual” da tabela de parâmetros do banco de dados local, informando as leituras de teor de água atuais do solo, para que nesse processo de atualização do banco de dados ele tenha subsídios para fazer cálculos e tomar decisões de acordo com os parâmetros cadastrados, atualizando outros campos como “ligada” e “ultima\_ligacao” da tabela “params”.



**Figura 18:** Pseudocódigo do comportamento do banco de dados

```

01 Se já passou o tempo "timeout_manual" desde a última alteração manual:
02     Voltar para o modo automático;
03
04 Se o sistema estiver em modo automático:
05     (Se ainda não passou o tempo "timeout_bomba" desde que ela foi ligada pela última vez,
06     Ou se a bomba deveria estar ligada por programação de horários,
07     Ou se a "umidade_atual" for menor que a "umidade_minima")
08     Desde que a "umidade_umidade" atual seja menor que a "umidade_maxima":
09         A bomba deve ser/permanecer ligada;
10         Enviar sinal HIGH para a bomba hidráulica;
11     Senão:
12         A bomba deve ser/permanecer desligada;
13         Enviar sinal LOW para a bomba hidráulica;
14
15     Se anteriormente a bomba estava desligada, e nesta atualização foi ligada:
16         Atualizar o campo "ultima_ligacao" para data/hora atuais;

```

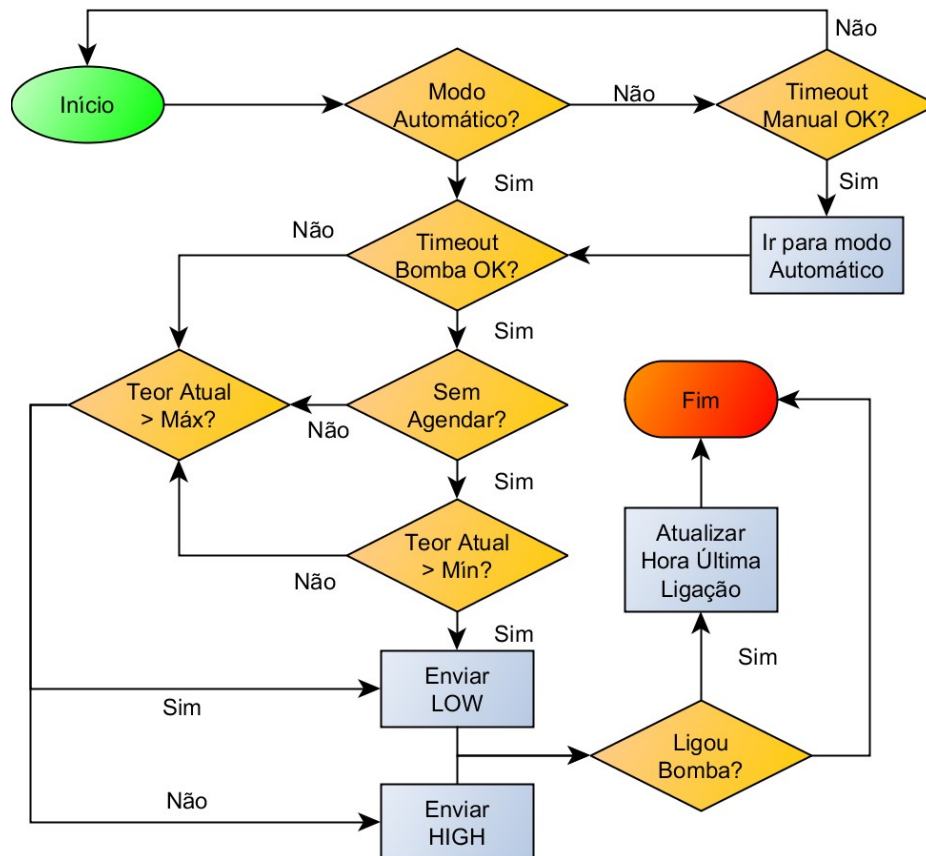
**Fonte:** Autor, 2015.

Uma vez atualizados os campos ou colunas da tabela "params" com base no *input* da leitura realizada pela RPi, uma versão atualizada das outras colunas é enviada ao algoritmo em *Python* que controla a bomba hidráulica indicando se esta deve estar ligada ou desligada no momento atual.

Este processo acontece em uma frequência definida no campo "sleep\_time", em segundos, por parâmetros informados pelo usuário pela interface WEB.

A figura 19 representa o fluxo das decisões tomadas pelo banco de dados sempre que uma atualização é realizada.

**Figura 19:** Fluxograma (Banco de Dados)



**Fonte:** Autor, 2015.

#### 4.7.3.2. SOFTWARE WEB

Visto o processo de funcionamento do hardware (RPi), é necessário descrever a interface WEB e suas funcionalidades.

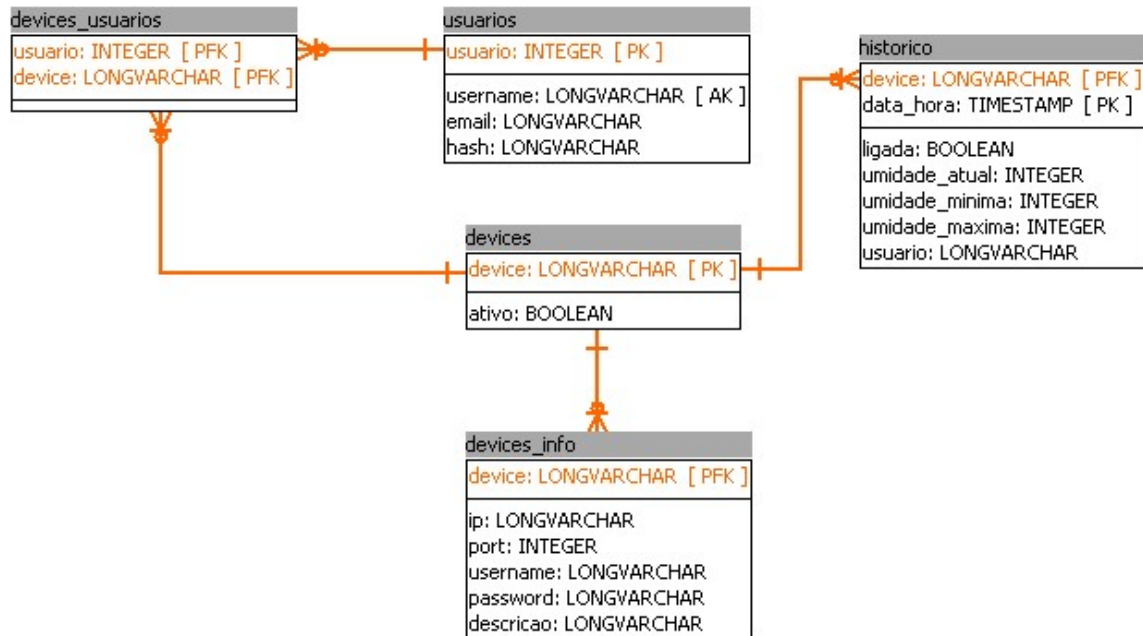
A interface WEB é responsável por alimentar todos os bancos de dados do sistema, desde o principal, o da própria interface, até os demais, os presentes nos dispositivos RASPBERRY PI™, fisicamente alocadas no campo.

A interação do usuário com os dispositivos RPi localizados no campo ocorre com a conexão do *website* no banco de dados interno das RPi, remotamente por meio da Internet, e dessa maneira o usuário da interface WEB pode interagir com os parâmetros que estão definidos na tabela “params” da RPi, modificando-os conforme

o necessário, e assim que as modificações são feitas, o próximo ciclo de verificação da RPi já identificará as mudanças. Seu detalhamento será feito a seguir.

A estrutura do banco de dados WEB é demonstrada na Figura 20:

**Figura 20:** Esquema do banco de dados do sistema WEB



**Fonte:** Autor, 2015.

A tabela “historico” do sistema WEB, embora de estrutura similar à dos dispositivos RPi, serve a um outro propósito. Nesse banco de dados geral há a discriminação de dispositivo, ou seja, esta tabela, por ser do banco de dados geral, abriga todo o histórico de aspersão de todos os dispositivos que estiverem ou estiveram em funcionamento dentro do sistema.

Para contemplar a parte de segurança de usuários no sistema final, também pode ser implementados pacotes de controle de usuários de terceiros, como, por exemplo, o OAuth 2.0, para maior segurança, em vez de controle próprio como proposto neste projeto, a critério das necessidades do mundo real, não presentes no ambiente de simulação.

Assim que o dispositivo RPi alimenta essa tabela no banco de dados WEB com os registros que guardara *offline* (o que é feito periodicamente pela RPi, obedecendo o tempo definido no parâmetro “atualizacao\_web” da tabela “params”

em seu banco de dados local), estes registros são permanentemente excluídos do banco de dados local passando a existir somente no banco de dados WEB, o que favorece uma maior economia de espaço de armazenamento nos dispositivos RPi.

É importante ressaltar que esta tabela pode sofrer a adição de novas colunas conforme a necessidade de gravação de novos dados pela RPi surgir (como umidade relativa do ar, temperatura, ph, etc), quando em funcionamento no mundo real.

A tabela "HISTORICO" foi intencionalmente criada sem acentuação para maior facilidade de desenvolvimento de código, e as colunas dessa tabela, como utilizada no desenvolvimento de testes, são as que seguem:

- Coluna "device": Corresponde ao código da RPi no sistema e serve para identificar a qual dispositivo cada registro dessa tabela pertence;
- Coluna "data\_hora": Registra a data e a hora em que ocorreu a entrada de histórico a que se refere o registro. Juntamente com o campo "device" forma a chave primária da tabela;
- Coluna "ligada": Registra se a bomba hidráulica fora ligada ou desligada no evento do registro, recebendo valores "true" ou "false", respectivamente;
- Coluna "umidade\_atual": Registra qual o teor de água do solo encontrado no ambiente da cultura no momento do registro;
- Coluna "umidade\_minima": Registra qual o valor do parâmetro que estipula o mínimo aceitável de teor de água do solo vigente no momento do registro;
- Coluna "umidade\_maxima": Registra qual o valor do parâmetro que estipula o máximo aceitável de teor de água do solo vigente no momento do registro;
- Coluna "usuario": Se refere a qual usuário do sistema efetuou a operação que gerou o registro, caso tenha sido feita de maneira manual. Caso a operação tenha sido feita automaticamente pelo dispositivo RPi, esta coluna recebe valor nulo (*null*).

A tabela “devices” nada mais é do que um simples cadastro de dispositivos (que são usados como chave estrangeira ou *foreign keys* em outras tabelas, como a “historico”), controlando se este dispositivo está ou não ativo no sistema.

Com o advento de uma implementação do sistema no mundo real, uma estrutura de servidores deverá ser mantida para suportar os serviços WEB. Dessa forma, custos relacionados à hospedagem desses serviços naturalmente devem ser repassados aos usuários interessados em manter o sistema em funcionamento em suas culturas.

Para controlar quais os dispositivos que estão em atividade no sistema, ou seja, em situação regular de utilização, é necessário manter essas informações cadastradas no banco de dados.

As colunas que realizam o exposto são as que seguem:

- Coluna “device”: Código interno que identifica o dispositivo no sistema;
- Coluna “ativo”: Controla se o dispositivo está ou não em situação regular no sistema, recebendo valores “*true*” ou “*false*”, respectivamente.

Já a tabela “devices\_info” é uma tabela complementar à tabela “devices”, e sua criação de maneira complementar envolve questões de segurança.

Para oferecer a possibilidade dos dispositivos RPi se comunicarem com o banco de dados geral para realizarem atualizações, inserindo novos registros de histórico, por exemplo, é necessário disponibilizar um usuário de acesso ao banco de dados para que esses dispositivos RPi consigam a ele se conectar.

Ocorre que além de informações de histórico, as RPi também atualizam suas informações de conexão com o sistema WEB, para que o mesmo consiga entrar em contato com o banco de dados local das RPi.

Essas informações devem ser atualizadas com frequência, pois os serviços de conexão à Internet geralmente não funcionam com a utilização de um IP fixo. Assim, se a conexão com a Internet for perdida por parte do modem que proporciona acesso à Internet para o dispositivo RPi, quando houver a reconexão possivelmente

estará associado a outro endereço de IP diferente do anterior. Logo, esta nova informação de contato deve ser replicada para o banco de dados geral do servidor WEB, já que esse servidor não conseguirá encontrar a RPi novamente para utilização pelo sistema até que isso aconteça. Assim, não é necessária a configuração de uma VPN, o que simplifica o processo.

Desta maneira, é necessário oferecer acesso também à tabela que guarda as informações do dispositivo para conexão pelo usuário de banco de dados disponibilizado para os dispositivos RPi, e este é o motivo da separação das informações dos dispositivos em duas tabelas distintas (“devices” e “devices\_info”).

A diferença entre as duas é que à primeira somente o sistema WEB possui acesso, já a segunda tabela permite ao usuário disponibilizado para os dispositivos RPi atualizarem as informações pertinentes ao funcionamento do sistema.

Assim é possível bloquear que, com o usuário disponibilizado para as RPi, seja possível se conectar ao banco de dados WEB e alterar a situação do dispositivo quanto à sua regularidade, o que impede que o sistema seja utilizado de maneira indevida.

As informações que são registradas nessa tabela são as seguintes:

- Coluna “device”: Indica a qual dispositivo as informações se referem;
- Coluna “ip”: Indica o endereço de IP que deve ser utilizado pelo sistema WEB para localizar o dispositivo RPi na Internet, atualizada a cada interação da RPi com o banco de dados WEB;
- Coluna “port”: Indica a porta de comunicação que deve ser utilizada em conjunto com o endereço IP, para localizar o dispositivo RPi na rede interna onde ele se encontra. Essa informação também é atualizada a cada interação do dispositivo RPi com o banco de dados WEB;
- Coluna “username”: Indica qual o usuário para acesso ao banco de dados interno das RPi, para conexão remota pelo servidor WEB;
- Coluna “password”: Indica qual a senha correspondente ao usuário da coluna “username”, para que seja possível acessar o banco de dados interno das RPi de maneira remota pelo servidor WEB;

- Coluna “descricao”: Descrição breve que ajuda a identificar a localização do dispositivo no sistema.

É claro que um sistema que suporta o controle de vários pivôs centrais, em várias propriedades, e oferece informações sobre várias culturas diferentes precisa de um controle de usuário para que seja possível o filtro das informações de acordo com as permissões de cada um, bem como a limitação de interação das pessoas somente com os dispositivos que lhes dizem respeito. Este é o objetivo da tabela “usuarios”.

Para efeitos de teste de funcionalidade do sistema, todavia, informações complementares como o nome da pessoa à qual o usuário se refere, seu endereço, telefones de contato ou outras informações acessórias não se fizeram necessárias. Assim, na utilização do sistema no mundo real, dependendo de novas demandas de usuários, não previstas no modelo inicial do projeto, ou necessidades futuras, podem ser adicionadas novas informações cadastrais sem problemas ou impacto ao funcionamento do sistema.

Dessa forma, para a realização de um controle de permissões é necessário o cadastramento dos usuários que acessam o sistema, mesmo que de maneira simples, e é este o objetivo da tabela “usuarios”, que foi intencionalmente criada sem acentuação para maior facilidade de desenvolvimento de código, e que possui as colunas a listar:

- Coluna “usuario”: É o código que identifica os usuários internamente no sistema;
- Coluna “username”: É o nome que o usuário utiliza para acessar o sistema WEB;
- Coluna “email”: É o e-mail de contato do usuário;
- Coluna “hash”: É um conjunto de letras e números calculado a partir da senha do usuário. Em outras palavras, por motivos de segurança, o sistema não guarda diretamente as senhas dos usuários no banco de dados, e sim uma combinação de caracteres que só é possível de ser calculada se como parâmetro para a lógica de geração for passada a senha correta. Embora existam vários recursos criptográficos para

realizar tal tarefa, foi escolhido o algoritmo MD5, para fins de testes, e o aprofundamento nesse mérito não é escopo do projeto.

Obviamente, para que um controle de permissões funcione é necessário realizar a associação dos usuários com os recursos sistêmicos que a eles se relacionam, e isso, nesse projeto, foi feito associando usuários a dispositivos pela tabela “devices\_usuarios”.

Dessa maneira, se determinado usuário está associado a dez dispositivos diferentes, ele poderá manipular todos os dez pelo sistema, mas somente estes que estão relacionados a ele, não possuindo acesso a outros dispositivos.

Para tal, foi criada uma simples tabela de associação, cujas colunas serão listadas abaixo:

- Coluna “device”: Indica o dispositivo que será associado ao usuário;
- Coluna “usuario”: Identifica o usuário que ganhará acesso ao dispositivo relacionado no campo “device”.

#### **4.7.4. COMUNICAÇÃO SEM FIO**

Nos testes de sistema realizados, havia uma distância desprezível entre a RPi de testes e o roteador que proporciona a conexão necessária para o funcionamento do sistema, por isso foi utilizado um adaptador de rede WiFi USB na RPi, modelo Alfa AWUS036H, mostrado na figura 21:



**Figura 21:** Adaptador WiFi Alfa AWYS036H



**Fonte:** Autor, 2015.

O adaptador USB utilizado (padrão de fábrica) pode ter sua antena substituída por outra de maior potência, o que é indispensável quando o sistema estiver sendo utilizado no mundo real, a menos que a comunicação seja feita por um dispositivo USB 3G ligado diretamente à RPi para fornecer acesso à *Internet* sem a necessidade de um roteador.

A vantagem de se utilizar comunicação WiFi, é que não é necessário cabeamento do terreno para interligar o dispositivo RPi com o roteador que proporciona acesso à Internet, todavia, é necessária a definição dos equipamentos a serem utilizados para que seja possível a comunicação *wireless* entre o *Access Point* (AP ou ponto de acesso, no caso o roteador).

Existem diferentes tipos de antenas para comunicação por ondas de rádio, sendo elas omnidirecionais, que têm um raio de ação de 360°, mas menor alcance (Electronicdesign, 1997; Wang *et al.*, 2012; Tarun e Sahu, 2013), as direcionais (*dish*, ou prato) que possuem o maior alcance, mas com foco direcionado, ou seja, menor raio de ação (Wirelessdesign, 2003; Horns *et al.*, 2012), e as semidirecionais (Painel e Yagi) que são praticamente um modelo intermediário (Wirelessreview, 1998; Ahyat *et al.*, 2013; Floc, 2013).

Todavia, ao se considerar também o cabeamento e a conexão direta na RPi pela sua porta ethernet, seria possível dispensar a necessidade da utilização de um adaptador de rede WiFi, e gastos com as antenas (projeto e equipamentos) e,

portanto, pode ser um meio de conexão mais viável que o WiFi, dependendo das necessidades do mundo real.

É possível ainda simplesmente se ignorar qualquer complexidade nesse sentido ao proporcionar *dongles* para conexão 3G ou 4G, que são dispositivos do tamanho de *pendrives* e podem ser diretamente acopladas às RASPBERRY PI™ no campo, que pode ser inclusive a solução de mais baixo custo dependendo da realidade encontrada no local de implantação.

De qualquer maneira, o escopo deste estudo não é a previsão sobre as necessidades a serem encontradas na implantação e utilização prática do sistema em nível de hardware, e sim no desenvolvimento do *software*. A estrutura física (tanto sensores quanto arquitetura de rede) não interfere em suas funcionalidades.

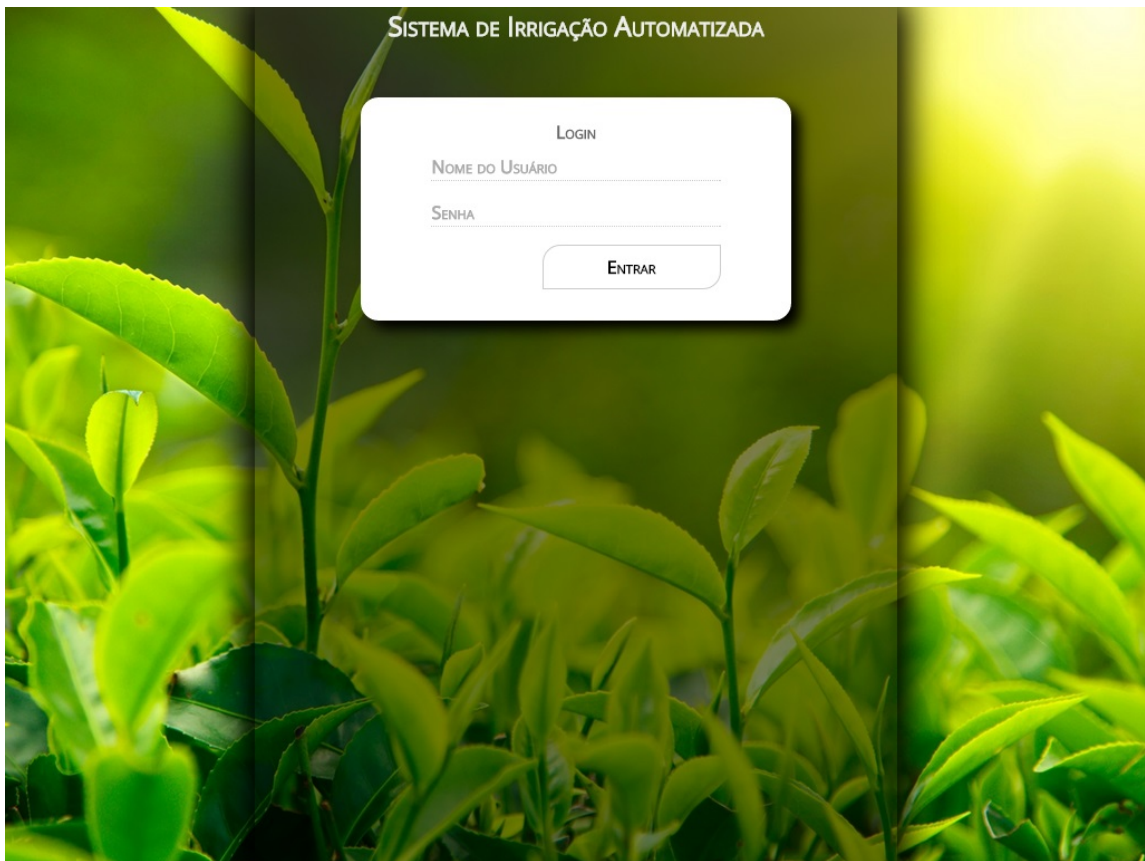
## **5. RESULTADOS**

O resultado dos trabalhos é um software, cuja interface e funcionalidade serão vistas nas próximas subseções.

### **5.1. TELA DE LOGIN**

Essa é a tela inicial do sistema, e possui apenas dois campos que solicitam as informações de autenticação ao usuário, como mostra a Figura 22.

**Figura 22:** Tela de Login



**Fonte:** Autor, 2015.

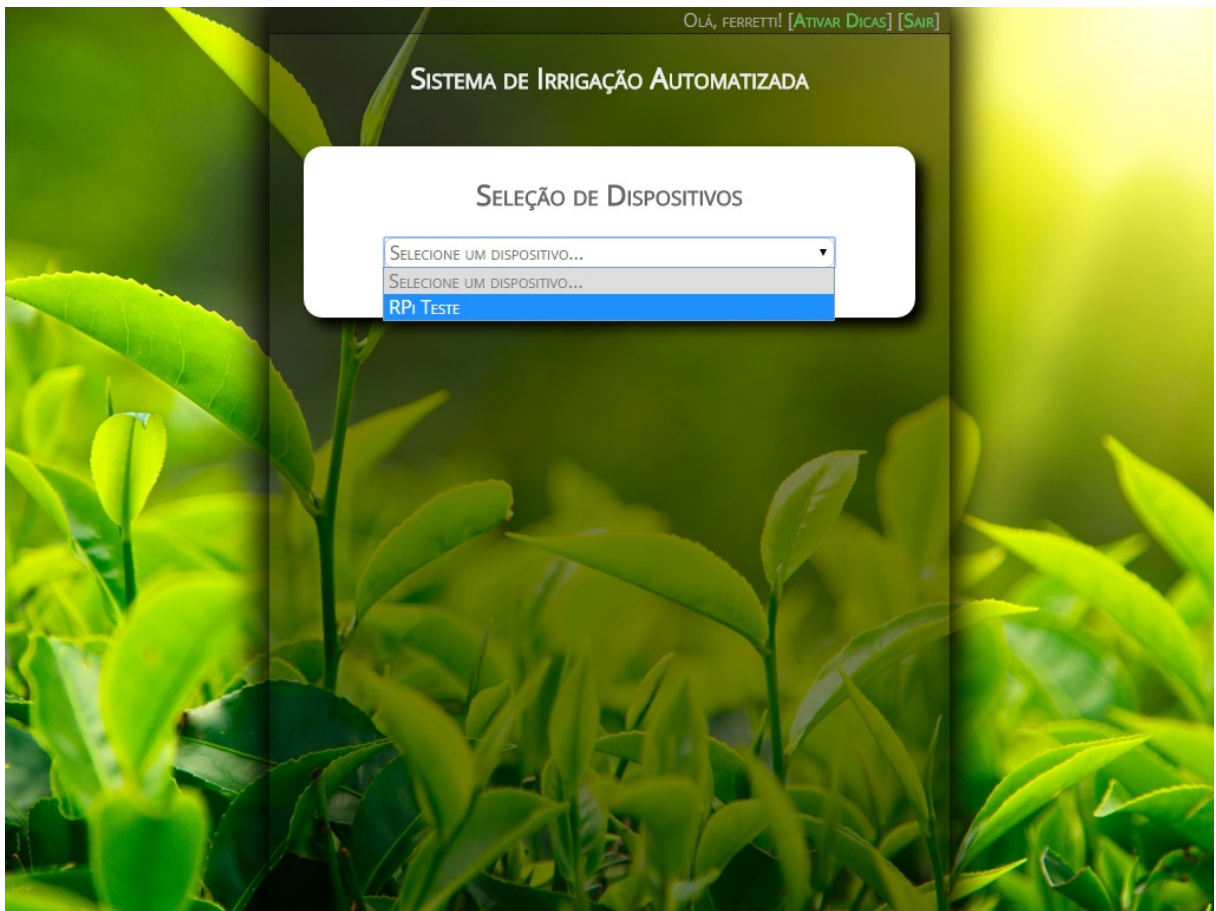
Entrando com as informações solicitadas pela tela de *login*, basta que o usuário clique no botão “Entrar” que a tela de seleção de dispositivos será disponibilizada, caso as credenciais estejam corretas.

## 5.2. SELEÇÃO DE DISPOSITIVOS

Ao efetuar o *login* com sucesso, o usuário terá à disposição uma lista onde poderá escolher os dispositivos que estiverem associados a ele na tabela “dispositivos\_usuarios”, e surge também um menu no topo da página com duas opções, além de uma mensagem de boas-vindas ao usuário.

A tela de seleção de dispositivos é representada pela Figura 23.

**Figura 23:** Seleção de dispositivos



**Fonte:** Autor, 2015.

### 5.3. MENU PRINCIPAL

No exemplo da Figura 23, há apenas um dispositivo RPi disponível, denominado RPi Teste, informação essa provinda do campo “descricao” da tabela “devices\_info”, mas haverão tantos dispositivos quantos estiverem associados ao usuário ativo.

Após escolher um dos dispositivos, são oferecidas ao usuário quatro opções no sistema, em seu menu principal, que aparece logo acima da janela de seleção de dispositivos.

As opções são as seguintes:

- “Seleção de Dispositivos”: Onde, no exemplo, foi selecionada a opção “RPI Teste” na lista. É a primeira opção do menu principal;
- “Painel de Controle”: Permite definição de parâmetros para o dispositivo selecionado bem como realização de agendamentos de irrigação. É a segunda opção do menu principal,
- “Interação em Tempo Real”: Permite manipulação do dispositivo de maneira remota. É a terceira opção do menu principal;
- “Relatórios”: Exibe relatórios de funcionamento do dispositivo selecionado, possibilitando filtragem por dia e hora (iniciais e finais). É a última opção do menu principal.

#### **5.4. PAINEL DE CONTROLE**

A interface do painel de controle, embora simples, proporciona várias ferramentas fundamentais para o funcionamento do sistema. Seus recursos estão agrupados em duas subdivisões na tela: “Parâmetros Automáticos” e “Agendamento”.

A subdivisão “Parâmetros Automáticos” está representada pela Figura 24.

**Figura 24:** Painel de Controle



**Fonte:** Autor, 2015.

De acordo com a figura acima se pode ver diversas opções a serem explicadas mais detalhadamente abaixo:

- Intervalo de teor de água a se manter: trata-se de um intervalo seguro entre os quais devem ser mantidos os níveis de teor de água do solo para que a cultura tenha seu melhor desempenho. No caso do exemplo simulado foi escolhido o intervalo entre 20% e 80% como um dos testes, ou seja, a bomba hidráulica deve ser ligada se a leitura for inferior a 20%, e desligada se ela for superior a 80% de teor de água. Correspondem aos campos “umidade\_minima” e “umidade\_maxima” da tabela “params” no banco de dados local da RPi selecionada, respectivamente;

- Tempo Descanso: tempo, em segundos, entre cada verificação das condições do solo pelo dispositivo que controla o sistema de irrigação para sua ativação/desativação. Se “0” (zero), as verificações serão feitas o mais rapidamente possível de acordo com as capacidades de processamento do dispositivo. Corresponde ao campo “sleep\_time” da tabela “params” no banco de dados local da RPi selecionada;
- Atualização WEB: Indica o tempo em minutos entre uma atualização e outra dos dados de histórico pelo dispositivo que controla o sistema de irrigação. Corresponde ao campo “atualizacao\_web” da tabela “params” no banco de dados local da RPi selecionada;
- Tempo Manual: Indica o tempo em segundos em que o sistema permanecerá em modo manual a partir do último comando manual que receber, antes de retornar ao modo automático, se o usuário se esquecer de sair do modo manual após terminar de realizar suas intervenções remotas. Se informado “0” (zero) o sistema não sairá do modo manual automaticamente. Corresponde ao campo “timeout\_manual” da tabela “params” no banco de dados local da RPi selecionada;
- Tempo Bomba: Indica o tempo mínimo em segundos em que deve ser permanecido ligado o sistema de irrigação após seu acionamento automático, lembrando que o usuário pode desligar manualmente o sistema em qualquer tempo, e caso os sensores detectem umidade superior à máxima estabelecida como parâmetro, haverá o desligamento automático do sistema de irrigação independente de ter se passado esse tempo. Corresponde ao campo “timeout\_bomba” da tabela “params” no banco de dados local da RPi selecionada.

Configuradas essas opções paramétricas no sistema, de acordo com as necessidades da cultura, o usuário deve clicar no botão “Salvar Parâmetros”, que executará o processo de gravar na RPi selecionada, de maneira remota, as informações.

É nesse momento que se torna importante a comunicação da RPi que está no campo com o servidor WEB, que acontece no intervalo de tempo definido do

parâmetro “Atualização WEB”. Essa comunicação será vista com mais detalhes a seguir.

Há dois motivos para que a RPi se conecte no servidor WEB para atualizar suas informações de maneira periódica. A primeira é manter o histórico atualizado para que seja possível gerar relatórios com uma precisão de no mínimo o tempo definido no campo “Atualizar WEB”, e a segunda é manter as “informações de contato” da RPi atualizadas no banco de dados WEB.

Essas informações de contato incluem o endereço de IP e a porta de acesso da RPi, que via o recurso *port forwarding* do roteador tornará possível encontrar a RPi dentro da rede interna, em meio a computadores e outras RPi, além de outras informações como usuário e senha do banco de dados local da RPi.

Todavia, há um campo entre os citados acima que necessita de um cuidado especial, o que representa o endereço IP.

Embora o endereço de IP dos servidores WEB sejam fixos, possibilitando que a RPi se conecte sempre ao mesmo IP para transmitir dados para o servidor WEB, a recíproca não é verdadeira, pois os IPs de conexões de clientes mudam com o tempo.

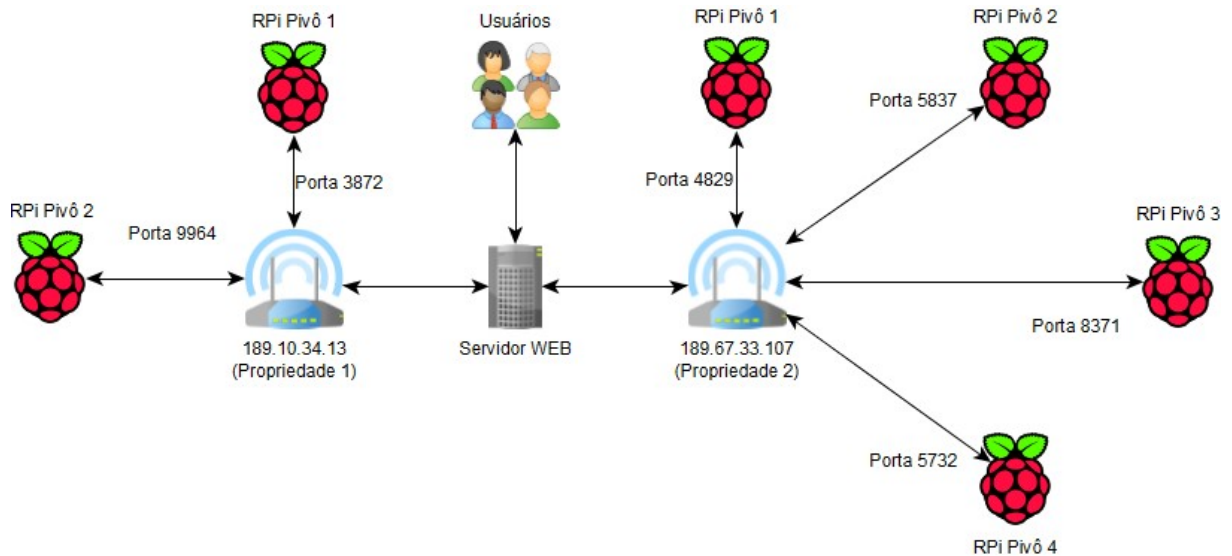
Para contornar essa limitação que inviabilizaria o contato direto com a RPi em tempo real, é importante que a RPi se comunique em um intervalo de tempo definido com o banco de dados central no servidor WEB, e a cada atualização, este banco de dados central detecta o IP de onde provêm a conexão e o atualiza automaticamente na tabela “devices\_info”, relacionando este IP com o código do dispositivo, coluna “device”.

Dessa maneira, caso a conexão do campo com a Internet seja perdida por algum tempo, a partir da próxima interação das RPi presentes na rede interna com o banco de dados central assim que ela for reestabelecida, será também reestabelecida a ligação entre o sistema WEB e aquela rede, pois mesmo que seja alterado o endereço IP público daquela rede, as próprias interações das RPi com o banco de dados central resolverão este problema automaticamente.



Na Figura 25, está demonstrado o caminho que o servidor WEB percorre para contatar diferentes dispositivos RPi para melhor visualização.

**Figura 25:** Comunicação (Servidor WEB e RPi)



**Fonte:** Autor, 2015.

De acordo com a figura acima, em um cenário hipotético onde o sistema WEB controla várias propriedades diferentes, supondo que o usuário que estiver acessando o sistema WEB possui acesso a todos os dispositivos RPi mostrados na figura, pertencentes a duas propriedades distintas (Propriedade 1 e 2), todos esses dispositivos seriam listados para seleção na tela de seleção de dispositivos (Figura 22).

Ao selecionar o “RPi Pivô 3” que está associado à “Propriedade 2”, por exemplo, o caminho a ser utilizado para conexão com o banco de dados local desse dispositivo “RPi Pivô 3” seria o IP 189.67.33.107, na porta 8371, ou seja, a *string* de conexão com o banco de dados local do pivô selecionado teria como *host* o valor “189.67.33.107:8371”, assim como para acessar o dispositivo “RPi Pivô 1” da mesma “Propriedade 2”, seria utilizado o valor “189.67.33.107:4829” e para acessar o dispositivo “RPi Pivô 2” da “Propriedade 1”, seria utilizado “189.10.34.13:9964”.

Não só a tela “Painel de Controle” utiliza esse método de comunicação, mas todas as outras interações do sistema WEB com os dispositivos RPi alocados nas propriedades seguirão a mesma lógica, então, a partir desse ponto, ao ser citada

“comunicação com os dispositivos RPi” em outros processos do sistema, já estarão claras as características dessa comunicação.

A partir do momento em que o usuário clicar no botão “Salvar Parâmetros” na parte de parametrização da tela “Painel de Controle”, e salvos os parâmetros no banco de dados local do dispositivo RPi selecionado, a próxima iteração desse dispositivo em seu laço lógico já contará com os novos parâmetros, e os aplicará onde couberem.

Para a realização dos testes do sistema não foram necessárias informações extras, todavia, no mundo real é sabido que algumas culturas, a depender de vários fatores incluindo o tipo de sensor a ser utilizado, podem sim precisar de mais algumas informações, como parâmetros de calibração, por exemplo.

Estes parâmetros podem tanto ser fixados diretamente na RPi que os controla, quanto parametrizados no sistema com a adição de novos campos na tela “Painel de Controle”, aba “Parâmetros Automáticos”, assim como qualquer outra nova informação que se faça necessária, mas esse tipo de necessidade deve ser levantada em momento oportuno, sendo irrelevante para o escopo do trabalho.

Há ainda a segunda subdivisão da tela “Painel de Controle”, intitulada “Agendamento”, que controla quando deve ser acionado o pivô central independentemente, mesmo que o nível de teor de água do solo não esteja abaixo do mínimo tolerado pelos parâmetros do sistema. Sua interface está representada pela Figura 26.

**Figura 26:** Agendamento para datas específicas

**PAINEL DE CONTROLE**

PARÂMETROS AUTOMÁTICOS | AGENDAMENTO

PROGRAMAÇÃO SEMANAL

TIPO DE AGENDAMENTO: DATAS ESPECÍFICAS

---

NOVO AGENDAMENTO

DATA ATIVAÇÃO: 31/10/2015

DATA DESATIVAÇÃO: 31/10/2015

HORA ATIVAÇÃO: 11:00    HORA DESATIVAÇÃO: 12:00

NOME: AGENDAMENTO2

ADICIONAR AGENDAMENTO

---

AGENDAMENTOS CADASTRADOS

[EXCLUIR][ATIVAR] 31/10/2015 00:00 ATÉ 31/10/2015 01:00  
[AGENDAMENTO-TESTE];

[EXCLUIR][DESATIVAR] 31/10/2015 11:00 ATÉ 31/10/2015 12:00  
[AGENDAMENTO2].

**Fonte:** Autor, 2015.

Nessa opção há a possibilidade de escolher se a programação será feita para cada dia da semana, por exemplo, todas as segundas-feiras das 10 às 11h da manhã, ou para datas específicas, como mostra o exemplo da figura acima. Não há um limite de quantidade para os agendamentos e as funcionalidades dessa tela são descritas a seguir:

- Tipo de Agendamento: Possibilita escolher entre os dias da semana (de domingo a sábado), ou a opção “Datas Específicas” como no exemplo acima, onde a programação não se repetirá todas as semanas, e sim acontecerá somente na data especificada nos demais parâmetros;
- Data Ativação: corresponde à data em que a ativação do pivô central deve acontecer. Esta opção só está disponível se selecionado o tipo de

agendamento “Datas Específicas”, caso contrário, ela não estará presente na tela;

- Data Desativação: corresponde à data em que a desativação do pivô central deve acontecer. Esta opção só está disponível se selecionado o tipo de agendamento “Datas Específicas”, caso contrário, ela não estará presente na tela;
- Hora Ativação: corresponde ao horário, no dia da semana selecionado na opção “Tipo de Agendamento”, ou na data especificada no campo “Data Ativação”, caso o tipo de agendamento selecionado seja “Datas Específicas”, em que deve ser ativado o pivô central;
- Hora Desativação: corresponde ao horário, no dia da semana selecionado na opção “Tipo de Agendamento”, ou na data especificada no campo “Data Ativação”, caso o tipo de agendamento selecionado seja “Datas Específicas”, em que deve ser desativado o pivô central;
- Nome: Um rótulo que identifica mais facilmente a que se refere o agendamento.

Quando o usuário clicar em “Adicionar Agendamento”, haverá comunicação do servidor WEB com o dispositivo RPi selecionado, cadastrando o agendamento no banco de dados local da RPi para que ela tenha parametrizados os eventos especificados pelo usuário onde deve ser acionado o pivô central, mesmo se o teor de água não for inferior ao mínimo.

A parte “Agendamentos Cadastrados” mostra ao usuário do sistema quais agendamentos estão gravados no banco de dados local da RPi selecionada, e possibilita que este usuário os exclua desse banco de dados, ou os mantenha lá, mas de maneira inativa (como mostra o texto riscado referente ao segundo agendamento cadastrado).

Caso o agendamento esteja desativado, obviamente ele não será considerado na tomada de decisão da RPi em relação à ativação ou não do pivô central, porém, a vantagem de desativar ao invés de excluir é que será possível reativá-lo de maneira mais rápida, sem necessidade de cadastrar outro agendamento, recurso especialmente útil quando se desejar manter um

agendamento que ocorre semanalmente fora de funcionamento em caráter temporário.

Tanto a exclusão, ativação e desativação de agendamentos cadastrados, como a inclusão de um novo agendamento pelo botão “Adicionar Agendamento”, utilizam comunicação direta com o dispositivo RPi.

A adição de recursos de agendamento se deve à vantagem de manter uma rotina de irrigação predefinida, capaz de manter o nível de teor de água no solo, caso o usuário do sistema pretenda entregar à tomada de decisão da RPi apenas situações que fogem da rotina, como, por exemplo, em um momento do ano onde o ambiente se encontra em temperatura elevada e, por consequência, seja necessária irrigação extra na cultura.

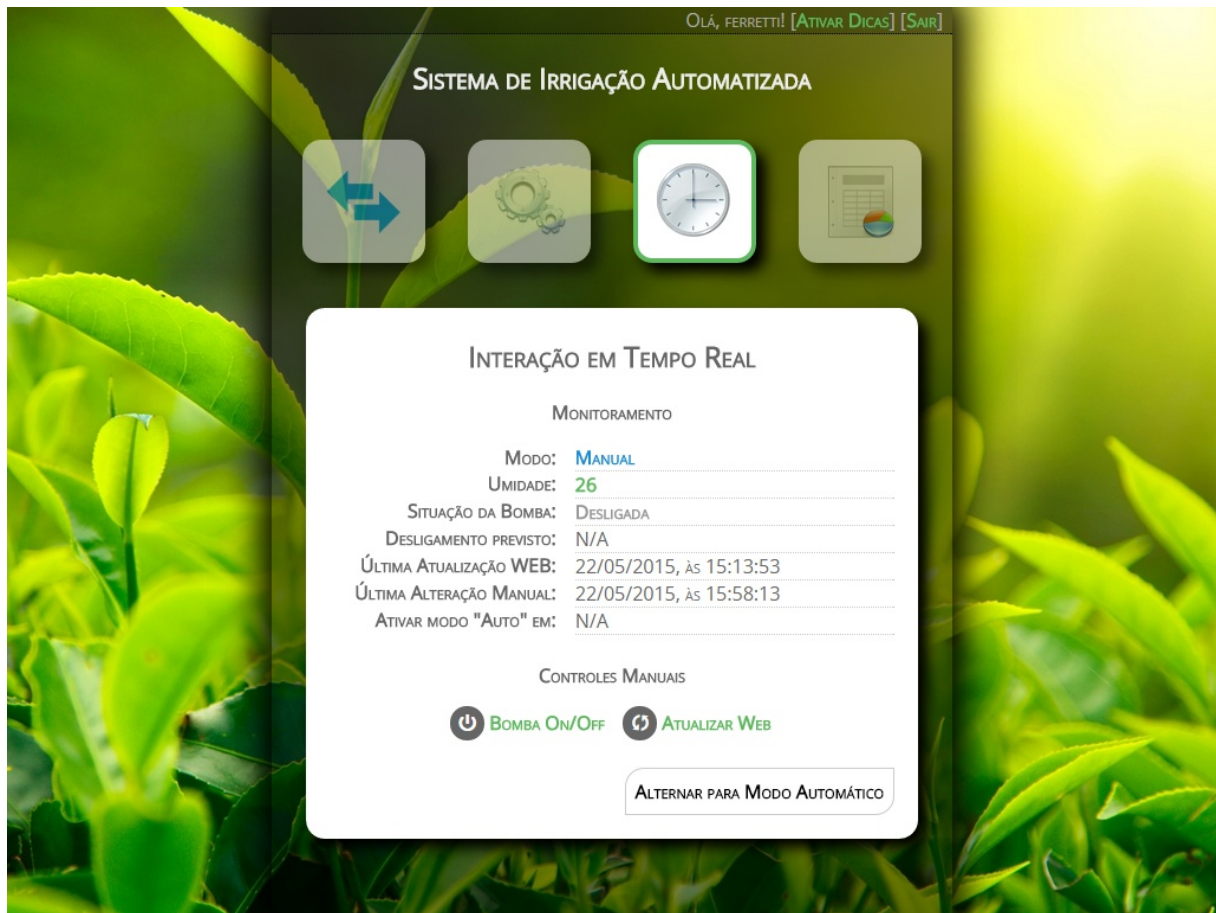
Tomando esse cenário como base, as irrigações automáticas por teor de água mínimo do solo, ou as manuais pelo próprio usuário do sistema, seriam em caráter mais emergencial que rotineiro.

Há também a possibilidade de utilização somente de agendamentos para ativação automática, dispensando a utilização de sensores de teor de água, o que não é recomendado, pois em situações atípicas que exigiriam ações imediatas o sistema não teria dados suficientes para tomá-las, mas se a irrigação rotineira de uma determinada cultura for suficiente ao agricultor, não deixaria de ser uma possibilidade dentro do sistema.

## **5.5. INTERAÇÃO EM TEMPO REAL**

Outra funcionalidade importante do sistema é permitir a interação do usuário em tempo real com o equipamento como na Figura 27.

**Figura 27:** Interação em tempo real



**Fonte:** Autor, 2015.

A tela representada pela figura acima proporciona algumas ferramentas que possibilitam a interação em tempo real do usuário com o dispositivo RPi selecionado, no sentido de obter informações sobre o solo e sobre o funcionamento do pivô central, além de possibilitar o acionamento/desativação do pivô central de maneira manual e remota e solicitar atualização imediata da base de dados geral por parte desse dispositivo.

Cada uma das funcionalidades ou informações presentes na referida tela são mais bem detalhadas abaixo:

- Modo: Indica se a RPi se encontra em modo automático, ou se espera somente por comandos manuais;
- Umidade: Corresponde à porcentagem proveniente da leitura do sensor de teor de água do solo;

- Situação da Bomba: Controla se a bomba hidráulica do pivô central se encontra ligada ou desligada no momento;
- Desligamento Previsto: Estimativa de quando será desativada a bomba hidráulica do pivô central caso ela esteja ligada;
- Última Atualização WEB: Data e hora de quando ocorreu a última atualização dos dados no sistema WEB pelo dispositivo RPi selecionado;
- Última Alteração Manual: Data e hora em que o último comando de origem manual foi recebido pelo dispositivo RPi;
- Ativar modo “Auto” em: Indica a data e a hora de quando o sistema vai voltar a ser colocado no modo automático pelo sistema. Corresponde à soma do tempo definido no campo “timeout\_manual” da tabela “params” do banco de dados local da RPi selecionada com a data e hora da última alteração manual realizada no sistema. No caso do exemplo não é possível determinar essa soma porque o sistema se encontra configurado para não voltar automaticamente para o modo automático;
- Bomba On/Off: Envia sinal para ativação ou desativação da bomba hidráulica para o dispositivo RPi selecionado;
- Atualizar WEB: Envia sinal solicitando atualização imediata da base de dados WEB por parte do dispositivo RPi selecionado.

O botão “Alternar para Modo Automático” só é visível se o dispositivo se encontra em modo manual, caso contrário, o botão disponível é o “Alternar para Modo Manual”, e eles tem funcionamento relativo à seleção de modos de funcionamento da RPi selecionada.

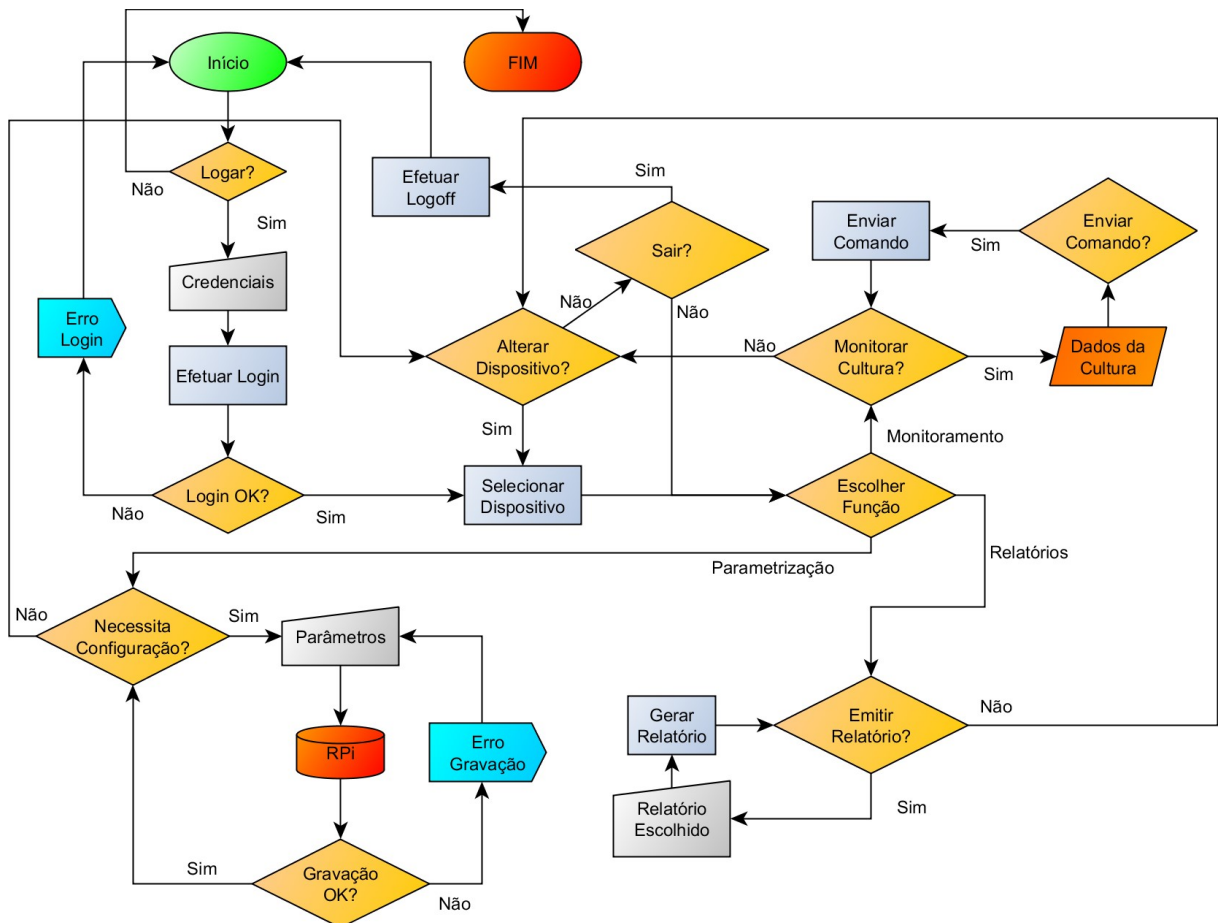
Caso a RPi se encontre em modo automático, as opções relacionadas como “Controle Manual” não permanecem visíveis para utilização.

O site tenta atualizar as informações relativas ao monitoramento a cada três segundos, o que pode ser acrescido pelo tempo de resposta do dispositivo RPi selecionado.

## 5.7. FLUXO DO SISTEMA

Para melhor visualização do processo abrangido pela interface WEB do sistema, foi adicionado seu fluxograma (representado pela Figura 28), desenvolvido a partir do resultado mostrado.

**Figura 28:** Fluxograma do Sistema



**Fonte:** Autor, 2015.

## 5.6. FUTUROS TRABALHOS

Foi disponibilizada no sistema uma opção no menu principal relativa à exibição de relatórios, porém, a análise estatística dos dados não seria possível, já que todos os dados disponíveis foram gerados em ambiente de testes. Essa análise também fugiria do escopo de desenvolvimento da ferramenta de administração remota de dispositivos de irrigação pela WEB ao qual se propõe o trabalho, por isso



optou-se por realizar um estudo estatístico em um trabalho futuro após a coleta de dados no mundo real pela utilização do sistema aqui desenvolvido, validando sua funcionalidade fora do ambiente de testes e cruzando as informações oferecidas por ele para avaliações do solo e do ambiente.

Dessa forma, uma vez que as métricas para análise desses dados serão desenvolvidas em outra oportunidade, embora já seja possível a extração de um simples relatório de horários e eventos relacionados à ativação da bomba hidráulica, os relatórios de natureza mais elaborada voltadas para esse fim também serão desenvolvidos em trabalho futuro, pois não é possível prever o tipo de relatório que se fará necessário para desenvolvimento imediato sem definir quais informações serão pertinentes ao próximo estudo.

Há também a necessidade de implementação da possibilidade da calibração dos sensores pelo sistema WEB, bem como a generalização do *software* para contemplar outros tipos de leituras e realidades que fogem à simulação realizada.

Com isso, a estrutura de tabelas deve ser aprimorada, adicionando entidades como, por exemplo, “agendamento”, o que agruparia os horários em diferentes conjuntos de informação, que podem variar de cultura para cultura no caso de uma plantação rotativa entre culturas de diferentes necessidades, assim como a verificação da possibilidade de união entre as tabelas de histórico e parâmetros, preocupações ausentes no modelo de testes por serem inerentes a necessidades do mundo real, ausentes no ambiente simulado.

Dessa maneira, a título de sugestão, também seria interessante, a depender do cenário encontrado no mundo real, a implementação de pacotes de segurança como o OAuth 2.0, com uma garantia extra em termos de segurança e controle de usuários.

## **6. CONCLUSÕES**

Foi produzida uma estrutura automatizada que permite, com baixo custo, promover o controle de irrigação com utilização do pivô central, mas que também

possibilita a intervenção externa por parte do responsável por meio da Internet para que este possa tomar decisões acerca de contingências e também obter relatórios sobre as condições de sua plantação no que diz respeito às condições do solo em relação ao teor de água para embasar suas decisões.

Em comparação com outros sistemas do mercado voltados ao mesmo objetivo é difícil estabelecer uma avaliação direta de custo em relação ao sistema aqui proposto, uma vez que cada solução tem recursos específicos e trabalha diferentemente.

Assim, em uma breve pesquisa de mercado realizada foram encontrados todos os tipos de custo, mas todos eles altos em relação ao proposto nesse trabalho, chegando à casa dos milhares de reais.

Este trabalho demanda apenas um dispositivo RASPBERRY PI™ que tem o custo de US\$25,00, e dependendo da cultura a ser trabalhada, devem ser considerados valores adicionais destinados ao custeio dos sensores específicos, comunicação sem fio, do circuito secundário de ativação, manutenção do servidor WEB entre outros.

Dessa maneira, uma implementação bastante completa do sistema proposto neste estudo não deve ultrapassar em muito o custo de US\$ 200,00, o que é um investimento ínfimo em comparação a suas funcionalidades, que estão inclusive ausentes em boa parte dos sistemas de automação estudados na pesquisa de mercado. Alguns sistemas simplesmente enviam alertas de texto a celulares (quando há essa opção) em vez de contemplarem um controle WEB completo das funcionalidades do sistema como é sugerido neste estudo, por exemplo, e não têm grande parte de sua flexibilidade e escalabilidade, além de geralmente se preocupam somente em manter o funcionamento da irrigação, e não levantar e armazenar dados históricos para a elaboração de indicadores para estudo, o que adiciona um viés gerencial ao processo.

Sobre os testes, eles deram resultado positivo, ou seja, o sistema atende aos requisitos levantados durante sua fase inicial, e segue os princípios adotados como básicos para uma boa qualidade do resultado.

A intenção do projeto não é esgotar as possibilidades do estudo, deixando inclusive caminho aberto para posteriores pesquisas, tanto de melhoria do processo quanto sobre implantação do sistema no mundo real a fim de colher dados estatísticos para o desenvolvimento de relatórios dentro do sistema que ofereçam cruzamento de dados suficientes para estudo das peculiaridades de determinada cultura em face das condições de solo e métricas de produtividade.

Dessa maneira, esse sistema tendo sido desenvolvido de maneira escalável, poderá implementar novas funcionalidades no futuro e tornar-se uma ferramenta cada vez mais completa e útil ao produtor rural.

Neste estudo, foi possível obter uma aplicação WEB que atende a todos os requisitos e objetivos propostos, de simples instalação e manutenção, baixo custo e que contém compatibilidade para uma configuração de hardware flexível em termos de equipamentos, sensores, e estrutura física, uma vez que a parte de hardware se tornou transparente ao sistema, que somente se preocupa com os dados colhidos pelos equipamentos, e não como esses dados são adquiridos por eles, o que proporciona grande liberdade e flexibilidade para futura adequação e aprimoramento.

O sistema desenvolvido, embora tenha sido elaborado usando o sistema de irrigação em pivô central como base, pode também controlar outros tipos de sistemas de irrigação.

## 7. REFERÊNCIAS BIBLIOGRÁFICAS

Agência Nacional de Águas. 2015. Disponível em: < <http://www.ana.gov.br/> >. Acesso em: Março, 2015.

AHYAT, E. N. et al. Frequency and pattern reconfigurable dipole-Yagi antenna. **Microwave and Optical Technology Letters**, Hoboken, v. 55, n. 2, p. 447-450, 2013. ISSN 0895-2477.

Arduino. 2016. Disponível em: < <https://www.arduino.cc/> >. Acesso em: Março, 2016.

BAGGALEY, K. **Learn a nerdier way to call mom.** Popular Science. 284: 80 p. 2014.

BAR-ZEEV, E. et al. The importance of microscopic characterization of membrane biofilms in an unconfined environment. **Desalination**, v. 348, n. 0, p. 8-15, 9/1/ 2014. ISSN 0011-

9164. Disponível em: <

<http://www.sciencedirect.com/science/article/pii/S0011916414003245> >.

BYSANI, C.; PRASAD, T. S. R. K.; CHUNDI, S. Raspberry Pi for Commercial Applications. **2013**, v. 11, n. 2, 2013. ISSN 2277-3061. Disponível em: <

<http://cirworld.com/journals/index.php/ijct/article/view/2476> >.

CHARLAND, A. et al. Mobile application development: Web vs. native. **Communications of the ACM**, v. 54, n. 5, p. 49, 2011. ISSN 0001-0782.

CHEN, S. et al. A graphene field-effect capacitor sensor in electrolyte. **Applied Physics Letters**, v. 101, n. 15, 2012. ISSN 00036951.

DATABASEANDNETWORK. W3C progress. (Internet Focus News & Products). **Database and Network Journal**, v. 32, n. 4, p. 14, 2002. ISSN 0265-4490.

DE SOUZA, A. R. et al. The Arduino board: a low cost option for physics experiments assisted by PC. **Rev. Bras. Ensino Fis.**, v. 33, n. 1, 2011. ISSN 1806-1117.

DONG, X.; VURAN, M. C.; IRMAK, S. Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems. **Ad Hoc Networks**, v. 11, n. 7, p. 1975-1987, 9// 2013. ISSN 1570-8705. Disponível em: < <http://www.sciencedirect.com/science/article/pii/S1570870512001291> >.

ELECTRONICDESIGN. **Powerful Omni Antenna Is Compact Enough For PCMCIA**. *Electronic Design*. 45: 176 p. 1997.

ERIC, L.; ERIC, L. Putting Database Performance to the Test. **eWeek**, 2002. ISSN 1530-6283.

EWEEK. How to Bring OpenSource Software into the Enterprise. **eWeek**, 2009. ISSN 1530-6283.

EWEEK. PostgreSQL vs MySQL How to Select the Right OpenSource Database. **eWeek**, 2010. ISSN 1530-6283.

FLOC. Broadband Quasi-Yagi Antenna for WiFi and WiMax Applications. **Wireless Engineering and Technology**, v. 4, n. 2, p. 87-91, 2013.

FRANCISCO, M. R. et al. Designing a minimalist socially aware robotic agent for the home. 2014.

GARRIGUES, C. et al. Promoting the development of secure mobile agent applications. **Journal of Systems and Software**, v. 83, n. 6, p. 959-971, 6// 2010. ISSN 0164-1212. Disponível em: < <http://www.sciencedirect.com/science/article/pii/S0164121209002829> >.

GONÇALVES, P. J. P. et al. The dynamic behavior of a cantilever beam coupled to a non-ideal unbalanced motor through numerical and experimental analysis. **Journal of Sound and Vibration**, v. 333, n. 20, p. 5115-5129, 9/29/ 2014. ISSN 0022-460X. Disponível em: < <http://www.sciencedirect.com/science/article/pii/S0022460X14004489> >.

HEEKS, R.; ROBINSON, A. Ultra-low-cost computing and developing countries.(Viewpoints / Emerging Markets)(Column). **Communications of the ACM**, v. 56, n. 8, p. 22, 2013. ISSN 0001-0782.

HORNS, D. et al. Searching for WISPy Cold Dark Matter with a Dish Antenna. 2012.

LONGO, L. A. et al. **Avaliação do desempenho de um pivô central da baixa pressão equipado com tubos de descida**. EMBRAPA, C.: Revista Ceres, Viçosa, v38, n216, p108-116, 1991 2011.

MEDVIDOVIC, N. et al. Software architecture and mobility: A roadmap. **Journal of Systems and Software**, v. 83, n. 6, p. 885-898, 2010. ISSN 01641212.

MITCHELL, H.; LAHTINEN, S. Preparing Students for Working Life. **ITNow**, v. 54, n. 2, p. 48-49, 2012. ISSN 1746-5702.

MORAES, M. J. et al. Automação em sistema de irrigação tipo pivô central para economia de energia elétrica. **Automation on central pivot irrigation system to energy savings**, v. 34, n. 6, p. 1075-1088, 2014. ISSN 01006916.

MORALES, L. R. V. **A utilização de sistemas fotovoltaicos de bombeamento para irrigação em pequenas propriedades rurais**. ZILLES, R.: Biblioteca Digital de Teses e Dissertações da USP 2011.

MORENO, M. A. et al. Optimal design of center pivot systems with water supplied from wells. **Agricultural Water Management**, v. 107, n. 0, p. 112-121, 5// 2012. ISSN 0378-3774. Disponível em: <

<http://www.sciencedirect.com/science/article/pii/S0378377412000327> >.

MySQL Connector/Arduino. 2016. Disponível em: < <https://launchpad.net/mysql-arduino> >. Acesso em: Março, 2015.

NAGY, Z. et al. Balancing envelope and heating system parameters for zero emissions retrofit using building sensor data. **Applied Energy**, v. 131, n. 0, p. 56-66, 10/15/ 2014. ISSN 0306-2619. Disponível em: <

<http://www.sciencedirect.com/science/article/pii/S0306261914006060> >.

NASCIMENTO, J. M. S. D. et al. Avaliação da uniformidade de aplicação de água em um sistema de gotejamento para pequenas propriedades Uniformity of water application for a drip irrigation system to small farms. **Ciência e Agrotecnologia**, v. 33, n. spe, p. 1728, 2009. ISSN 1413-7054.

NOTEPAD++. 2016. Disponível em: < <https://notepad-plus-plus.org/> >. Acesso em: Março, 2015.

NOVO, P.; CHU, V.; CONDE, J. P. Integrated optical detection of autonomous capillary microfluidic immunoassays:a hand-held point-of-care prototype. **Biosensors and Bioelectronics**, v. 57, n. 0, p. 284-291, 7/15/ 2014. ISSN 0956-5663. Disponível em: <

<http://www.sciencedirect.com/science/article/pii/S0956566314000876> >.

OLIVEIRA, L. F. C. D.; ALVES FILHO, A. S.; SILVEIRA, P. M. D. **Distribuição de água no solo aplicada por um pivô central**: Bioscience Journal, Uberlândia, v19, n2, p79-87, May/Aug 2003 2011.

OVIEDO, D. et al. Multiple intelligences in a MultiAgent System applied to telecontrol. **Expert Systems with Applications**, v. 41, n. 15, p. 6688-6700, 11/1/ 2014. ISSN 0957-4174. Disponível em: < <http://www.sciencedirect.com/science/article/pii/S0957417414002711> >.

PAULINO, J. et al. Situação da agricultura irrigada no Brasil de acordo com o censo agropecuário 2006. **Brazil agriculture irrigated status according to the agricultural census of 2006**, v. 16, n. 2, p. 163-176, 2011. ISSN 14137895.

Raspberry Pi. 2016. Disponível em: < <https://www.raspberrypi.org/> >. Acesso em: Março, 2015.

Raspbian. 2016. Disponível em: < <https://www.raspberrypi.org/downloads/raspbian/> >. Acesso em: Março, 2015.

SANSING, C. **Life With Raspberry Pi**. New York. 59: 34 p. 2013.

SREEJITH, A. G. et al. A Raspberry Pi-Based Attitude Sensor. **Journal of Astronomical Instrumentation**, v. 03, n. 02, p. 1440006, 2014. Disponível em: < <http://www.worldscientific.com/doi/abs/10.1142/S2251171714400066> >.

SU, P. G. et al. Electrical and sensing properties of a flexible humidity sensor made of polyamidoamine dendrimer-Au nanoparticles. **Sensors and Actuators, B: Chemical**, v. 165, n. 1, p. 151-156, 2012. ISSN 09254005.

TARUN, D.; SAHU, O. P. Omni Directional Antenna Assisted Scheme to Minimize Redundancy in Wireless Sensor Networks. **International Journal of Computer Network and Information Security**, v. 5, n. 4, p. 57, 2013. ISSN 2074-9090.

UWIRE. **Raspberry Pi. UWIRE Text**: 1 p. 2013.

WANG, X. et al. An all-printed wireless humidity sensor label.(Report). **Sensors & Actuators: B. Chemical**, v. 166 167, p. 556, 2012. ISSN 0925-4005.

WERMELINGER, M.; BANDARA, A. Commentary on ‘Software architectures and mobility: A Roadmap’. **Journal of Systems and Software**, v. 83, n. 6, p. 899-901, 6// 2010. ISSN 0164-1212. Disponível em: < <http://www.sciencedirect.com/science/article/pii/S0164121210000634> >.

WIRELESSDESIGN. Parabolic dish antenna.(EDITOR'S SPOTLIGHT: ANTENNAS). **Wireless Design & Development**, v. 11, n. 7, p. 16, 2003. ISSN 1076-4240.

WIRELESSREVIEW. YAGI ANTENNA. **Wireless Review**, 1998. ISSN 1099-9248.

WIRTH, M. et al. Keep an eye on the Pi – Using the Raspberry Pi as inexpensive, yet powerful platform for vision research. **Acta Ophthalmologica**, v. 91, p. 0-0, 2013. ISSN 1755-3768. Disponível em: < <http://dx.doi.org/10.1111/j.1755-3768.2013.T028.x> >.

WONG, L. et al. Interactive Museum Exhibits with Microcontrollers: A Use-Case Scenario. 2015.

ZHAO, Z. et al. A low cost fiber-optic humidity sensor based on silica sol-gel film. **Sensors and Actuators, B: Chemical**, v. 160, n. 1, p. 1340-1345, 2011. ISSN 09254005.

ZOLIN, C. A. et al. ECONOMIC VIABILITY OF RETROFITTING EMITTERS IN CENTER PIVOT IRRIGATION SYSTEMS. **Eng. Agric.**, v. 32, n. 3, p. 602-608, 2012. ISSN 1809-4430.