UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SISTEMA DE MONITORAMENTO DE SOLO E LAVOURA

JOHNNY JARBAS HERTEL

BLUMENAU
2017

JOHNNY JARBAS HERTEL

PROTÓTIPO DE SISTEMA DE MONITORAMENTO DE SOLO E LAVOURA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer, Me - Orientador

BLUMENAU
2017

PROTÓTIPO DE SISTEMA DE MONITORAMENTO DE SOLO E LAVOURA

Por

JOHNNY JARBAS HERTEL

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Miguel Alexandre Wisintainer, Me. – Orientador, FURB
Membro:	Prof(a). Mauro Marcelo Mattos, Dr. – FURB
Membro:	Prof(a). Francisco Adell Péricas, Me. – FURB

Blumenau, 03 de julho de 2017.

Dedico este trabalho aos meus pais, meus sogros e minha esposa Juliana Volpi Hertel que, com muito carinho e apoio, não mediu esforços para que eu chegasse até esta etapa de minha vida.

AGRADECIMENTOS

Aos meus pais, Lorenz e Elisabeth Hertel, pelo apoio durante a minha graduação.

Aos meus sogros, por terem estado ao meu lado sempre me motivando a continuar a graduação.

A minha esposa Juliana Volpi Hertel, pela sua inesgotável paciência e por sua capacidade de acreditar e investir em mim.

Ao meu professor orientador, Miguel Alexandre Wisintainer, pela paciência na orientação e incentivo que tornaram possível a conclusão deste trabalho.

Na natureza nada se cria, nada se perde, tudo se transforma.

Antoine Lavoisier

RESUMO

A técnica de irrigação está se tornando cada vez mais utilizada, até mesmo em pequenas propriedades rurais, tanto para aumentar a produtividade como para garantir um fornecimento adequado de água em períodos de seca. Este trabalho apresenta o desenvolvimento de um sistema de monitoramento de solo e lavoura em tempo real com o uso de um microcontrolador ESP8266 Thing. Foi construído utilizando como componentes principais um sensor higrométrico, um painel solar, uma bateria e um microcontrolador ESP8266 Thing. As informações são coletadas pelo equipamento e disponibilizadas em um sistema Web através de gráficos, que exibem as leituras feitas e informam ao agricultor a necessidade de irrigação do solo. O trabalho está dividido em duas partes: a primeira parte que é responsável por captar as informações foi desenvolvida em Arduino para o microcontrolador ESP8266 Thing e a segunda parte que é responsável por receber as informações e exibi-las ao agricultor foi desenvolvida em Hypertext Preprocessor (PHP) e faz uso do banco de dados MySQL. A partir de testes realizados, foi demonstrado que o sistema oferece um bom grau de confiabilidade, determinando o momento ideal de irrigação da plantação.

Palavras-chave: Internet das coisas. Irrigação. Plantação. ESP8266.

ABSTRACT

Irrigation´s technique is becoming increasingly used, even on small rural properties, both for increase productivity as for ensure adequate water supply in periods of drought. This work presents the development of a monitoring system in real time of the ground and tillage with the use of an ESP8266 Thing microcontroller. It was elaborated using as main components a hygrometric sensor, a solar panel, a battery and an ESP8266 Thing microcontroller. The informations are collected by the equipment and made available on the Web system by means of graphs, that show the readings made and inform the farmer of the need for soil irrigation. The work is divided into two parts: the first part is responsible to collect the information and was developed in Arduino for the ESP8266 Thing microcontroller and the second one that is responsible for receiving the information and displaying it to the farmer was developed in Hypertext Preprocessor (PHP) and makes use of the MySQL database. From tests carried out, it has been demonstrated that the system offers a good level of reliability, determining the ideal time of irrigation of the plantation.

Key-words: Internet of things. Irrigation. Plantation. ESP8266.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CSS - Cascading Style Sheets

CSV - Comma-separated values

HTTP - HyperText Transfer Protocol

IDE - Integrated Development Environment

IOT - Internet of Things

JSON - JavaScript Object Notation

MHz - Megahertz

MVC - Model View Controller

Ohms - Unidade de medida da resistência elétrica

ONU - Organização das Nações Unidas

PHP - Hypertext Preprocessor

RF - Requisito funcional

RNF - Requisito não funcional

RSSF - Rede de sensores sem fio

SMSL – Sistema de Monitoramento de Solo e Lavoura

SSID - Service Set Identifier

UART - Universal Asynchronous Receiver/Transmitter

UFSM/RS - Universidade Federal de Santa Maria/Rio Grande do Sul

URL - Uniform Resource Locator

USB - Universal Serial Bus

Wi-Fi - Wireless Fidelity

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Desde o final da segunda guerra mundial teve início um processo de declínio do conjunto de técnicas de cultivo que vem sendo utilizado durante vários séculos pelos camponeses e pelas comunidades indígenas que denominamos de agricultura tradicional. Na década de 60, começa a ser implantada uma nova agricultura, chamada moderna, que se caracteriza pelo grande uso de insumos externos, utilização de máquinas pesadas, mau manejo do solo, uso de adubação química e biocidas. A agricultura moderna existe há poucos anos e já demonstra o colapso de suas técnicas. Desta forma, não pode ser considerada uma agricultura de fato sustentável, ao contrário da agricultura tradicional, que tem centenas de anos de história e sustentabilidade em longo prazo (WOLF, 2017).

Atualmente a humanidade enfrenta desafios cada vez maiores para produzir alimentos, fibras, energia produtos madeireiros e não madeireiros de forma compatível com a disponibilidade de recursos naturais. Neste sentido, são intensos os apelos para que seja difundida em todo o mundo a concepção de Agricultura Sustentável (BALBINO et al., 2012).

O projeto “Sistema Irriga”, tecnologia desenvolvida e patenteada pela Universidade Federal de Santa Maria/Rio Grande do Sul (UFSM/RS), tem uma proposta de Agricultura de Precisão voltada para irrigação. O projeto tem por objetivo recomendar quando e quanto de água deve ser aplicado em cada irrigação, sendo que esta demanda está parametrizada pelos seguintes fatores: parâmetros agronômicos da cultura, características do solo, equipamento de irrigação e dados climáticos obtidos através de Estações Meteorológicas Automáticas (SISTEMA IRRIGA, 2017).

A proposta deste trabalho é desenvolver uma rede de dispositivos capazes de monitorar informações de umidade do solo de uma plantação de morangos e uma aplicação que coleta estes dados e transforma em informações ao agricultor, onde se pretende tornar possível identificar a área com necessidade de irrigação, possibilitando reduzir custos e principalmente o consumo de água.

1.1 A PERGUNTA DA PESQUISA

Como aplicar tecnologia no campo para determinar o melhor momento de irrigar uma plantação de morangos?

1.2 OBJETIVOS

O objetivo deste trabalho é desenvolver um sistema de monitoramento de solo e lavoura para a produção de morangos.

Os objetivos específicos do trabalho são:

a) desenvolver uma rede de dispositivos utilizando o módulo ESP8299 Thing capaz de realizar a captura da umidade do solo numa plantação de morangos;
b) transmitir os dados coletados para uma aplicação que vai consumi-los para apresentação ao agricultor;
c) desenvolver uma aplicação para consumir os dados e transformá-los em informações para que o agricultor possa tomar decisões, como identificar a área com necessidade de irrigação.

1.3 ESTRUTURA

Este trabalho está dividido em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O primeiro capítulo apresenta a introdução e os objetivos do trabalho. O segundo capítulo apresenta a fundamentação teórica sobre os temas abordados no trabalho, como: a falta de agua no planeta, sustentabilidade, *internet of things* (IOT), ESP8266 e sensores. O terceiro capítulo mostra os principais pontos do desenvolvimento do trabalho como: requisitos, especificação, implementação e resultados obtidos. Por último, o quarto capítulo relata as conclusões e sugere extensões para o desenvolvimento de trabalhos futuros.

1.4 RELEVÂNCIA DO TRABALHO

O trabalho proposto se mostra relevante pelo fato da possibilidade de ampliar a sustentabilidade das lavouras agrícolas, além disso, agrego relevância social, pois se pretende com o trabalho atingir além de grandes produtores, os produtores com menor poder aquisitivo. No campo tecnológico a relevância desta proposta é percebida pela utilização do módulo ESP8266, além da confecção dos dispositivos responsáveis pelo monitoramento de umidade e temperatura.

Atualmente, muitos mecanismos de irrigação, segundo Fancelli e Dourado Neto (2004), consistem em uma distribuição de água constante, sem saber da real necessidade da mesma. A proposta deve tornar possível identificar quais as áreas que necessitam receber irrigação, tornando assim a tecnologia a favor da sustentabilidade.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 expõe um panorama sobre como a população do planeta já sofre as consequências da falta de água. A seção 2.2 fala sobre a sustentabilidade e a necessidade da irrigação na lavoura. A seção 2.3 oferece uma visão geral sobre Internet das Coisas (IoT). A seção 2.4 apresenta informações sobre microcontroladores e traz detalhes do módulo ESP8266. A seção 2.5 fala sobre sensores e a seção 2.6 sobre painéis solares justificando sua utilidade neste trabalho. Por fim, a seção 2.7 traz detalhes sobre redes de comunicação e a seção 2.8 um breve estudo de trabalhos correlatos.

2.1 A FALTA DE ÁGUA NO PLANETA

A água é o recurso natural mais abundante do planeta. De maneira quase onipresente, ela está no dia a dia dos sete bilhões de pessoas que habitam o planeta. Além de matar a sede, a água está nos alimentos, nas roupas, nos carros. Mas o recurso mais fundamental para a sobrevivência dos seres humanos enfrenta uma crise de abastecimento. Estima-se que cerca de 40% da população global viva hoje sob a situação de estresse hídrico. Essas pessoas habitam regiões onde a oferta anual é inferior a 1700 metros cúbicos de água por habitante, limite mínimo considerado seguro pela Organização das Nações Unidas (SEGALA, 2012).

A Organização das Nações Unidas (ONU) prevê que em 2050 mais de 45% da população mundial não poderá contar com a porção mínima individual de água para necessidades básicas. Segundo dados estatísticos existem hoje 1,1 bilhão de pessoas praticamente sem acesso à água doce (JACOBI, 2006). Daqui a 25 anos, Índia, China e África do Sul deverão entrar na estatística. “Nesses lugares, as reservas deverão se esgotar completamente”, alerta o autor do estudo, o geólogo Igor Shiklomanov (1998, apud ANGELO; MELLO; VOMERO, 2000, p. 48-54), do Instituto Hidrológico Estatal de São Petersburgo, Rússia.

O precário abastecimento d'água desses lugares vai falir por vários motivos. “Nos últimos cinquenta anos, a população mundial triplicou e o consumo de água aumentou seis vezes”, sintetiza o ecólogo paulista José Galizia Tundisi, do Instituto Internacional de Ecologia. Com a população cresce também a agricultura, a atividade humana que mais consome o líquido. “Os países em desenvolvimento vão aumentar seu uso de água em até 200% em 25 anos”, disse Shiklomanov (1998, apud ANGELO; MELLO; VOMERO, 2000, p. 48-54).

2.2 SUSTENTABILIDADE NA LAVOURA

Sete bilhões de habitantes sobre a terra, marca já atingida em 2011, segundo as Nações Unidas. E as projeções apontam que a humanidade atingirá a nove bilhões de pessoas em 2050, ou seja, dois bilhões a mais que a população atual. Os desafios para a agricultura: alimentar uma crescente população, com aumento de renda e urbanização acelerada. Para superar este desafio, é premente aumentar a produção de alimentos, com segurança, qualidade e uso sustentável da base de recursos naturais. Inovação agropecuária é componente crítico do processo de desenvolvimento sustentável e condição para melhoria da alimentação e nutrição no mundo (LOPES; CONTINI, 2012).

Os desafios no horizonte são enormes. Tecnologias mais eficientes serão necessárias para permitir o atendimento das necessidades básicas de alimentos para a sociedade brasileira, além da produção de excedentes exportáveis para o mundo, constituindo em oportunidade de negócios e responsabilidade social, nacional e mundial. Ao mesmo tempo, estas mesmas tecnologias deverão incorporar práticas para a preservação dos recursos naturais, como solo, água, florestas e biodiversidade. Acrescente a esperada contribuição para o mais recente desafio do aquecimento global e seus potenciais efeitos sobre a produção agrícola (LOPES; CONTINI, 2012).

A finalidade básica da irrigação é proporcionar água às culturas de maneira a atender às exigências hídricas durante todo seu ciclo, possibilitando altas produtividades e produtos de boa qualidade. Sendo que a quantidade de água necessária às culturas é função da espécie cultivada, da produtividade desejada, do local de cultivo, do estádio de desenvolvimento da cultura, do tipo de solo e da época de plantio (BERNARDO, 1997).

2.3 INTERNET DAS COISAS

Internet das coisas (IoT) é um termo utilizado que se refere a uma revolução tecnológica que tem o objetivo de conectar todos equipamentos eletrônicos que utilizamos no dia a dia a rede mundial de computadores. O intuito, é que através de dispositivos que se comuniquem uns com os outros e retornem benefícios aos usuários, o mundo físico e o digital se tornem um só. A ideia por trás da internet das coisas nasce de uma nova dimensão de conexão propiciada pela Internet – além de possibilitar a comunicação a qualquer tempo e em qualquer lugar, agora também considera a comunicação de qualquer coisa (DINIZ, 2006). A IoT é uma infraestrutura de rede dinâmica e global com capacidades de autoconfiguração, baseada em protocolos de comunicação padronizados e interoperáveis. Nela “coisas” físicas e

virtuais têm identidades, atributos físicos e personalidades virtuais (TEIXEIRA; PEREIRA; VIEIRA et al., 2014).

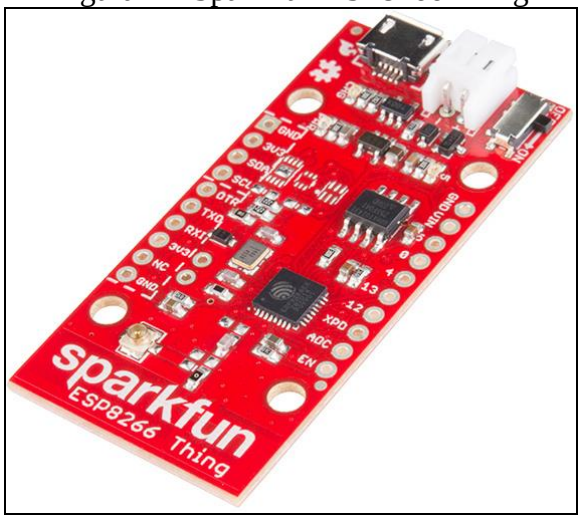
2.4 MICROCONTROLADORES

Microcontrolador é um circuito integrado capaz de executar programas. Um dos mais conhecidos é o Arduino. Por conta de sua arquitetura aberta permite ser utilizado em inúmeros tipos de projetos (VICENZI, 2015). Dentre as principais características estão à frequência de *clock* de poucos Megahertz (MHz) ou talvez menos. Os microcontroladores operam a uma frequência muito baixa se comparados com os microprocessadores atuais. São adequados para a maioria das aplicações usuais como, por exemplo, controlar uma máquina de lavar roupas ou uma esteira de chão de fábrica. O seu consumo em geral é relativamente pequeno, normalmente na casa dos miliwatts e possuem geralmente habilidade para entrar em modo de espera (*sleep* ou *wait*) aguardando por uma interrupção ou evento externo, como por exemplo, o acionamento de uma tecla, ou um sinal que chega via uma interface de dados. O consumo destes microcontroladores em modo de espera pode chegar à casa dos nanowatts, tornando-os ideais para aplicações onde a exigência de baixo consumo de energia é um fator decisivo para o sucesso do projeto (MICROCONTROLADORES, 2017).

Hoje se vive num mundo onde as coisas estão cada vez mais móveis e conectadas, principalmente no que tange à internet. E dentre os inúmeros módulos que surgiram recentemente para explorar a onda da Internet das Coisas (IoT), o de maior destaque é o ESP8266. Esse módulo foi originalmente criado pela Espressif, e possui um conjunto de alto desempenho, alta interação *wireless*, projetado para espaços pequenos com restrição de consumo de energia para plataformas móveis. Ele fornece a capacidade de incorporar Wireless Fidelity (Wi-Fi) dentro de outros sistemas, podendo funcionar como aplicativo independente, com menor custo e com um mínimo de espaço, além disso, possui uma grande facilidade para se integrar a outras soluções, além de comunicação serial Universal Asynchronous Receiver/Transmitter (UART) e até Wi-Fi (ASSUNÇÃO, 2017).

A Figura 1 ilustra o modelo SparkFun ESP8266 Thing, fabricado pela SparkFun e o Quadro 1 traz algumas de suas características. O Thing foi desenvolvido para suprir algumas necessidades que o ESP8266 original carecia, assim ele vem equipado com carregador LiPo, fonte de alimentação e outros circuitos de apoio necessários (SPARKFUN, 2017a).

Figura 1 – SparkFun ESP8266 Thing



Fonte: SparkFun (2017a).

Quadro 1 – Características do ESP8266

Alimentação: 3.3V
Carregador de bateria LiPo integrado
Suporta rede Wi-Fi
Processador integrado de 32bits de baixo consumo
Conector para antena externa
Dimensões: 55 x 26 x 7mm

Fonte: SparkFun (2017a).

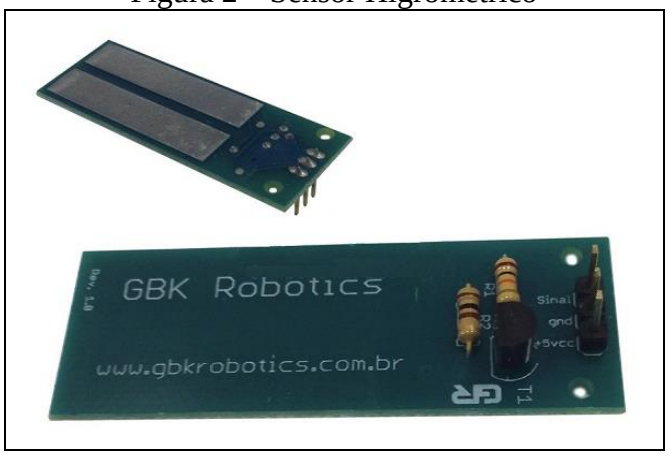
2.5 SENSORES

Sensores servem para informar um circuito eletrônico a respeito de um evento que ocorra externamente, sobre a qual ele deva atuar, ou a partir do qual ele deva comandar uma determinada ação. Há uma série de características relacionadas aos sensores que devem ser levadas em consideração na hora da seleção do sensor mais indicado para uma aplicação. Existem diversos tipos de sensores como exemplo os sensores analógicos e os sensores térmicos. O sensor analógico pode assumir qualquer valor no seu sinal de saída ao longo do tempo, desde que esteja dentro da sua faixa de operação. Essas variáveis são mensuradas por elementos sensíveis com circuitos eletrônicos não digitais (WENDLING, 2010).

O Sensor higrométrico é capaz de medir a umidade do solo em determinado local, atuando em conjunto com placas microcontroladoras. Desenvolvido de uma forma mais compacta ele é composto por um sensor que através de duas sondas realiza a medição da umidade por meio da aferição da corrente entre as sondas. Muito aplicado em projetos eletrônicos e de automação residencial, o sensor em conjunto com a placa é capaz de medir a umidade do solo, e quando atingir determinado índice a placa pode acionar um equipamento para irrigação da área ou até mesmo pode emitir sinais sonoros ou luminosos (PROESI, 2017).

A Figura 2 apresenta o sensor higrométrico utilizado no projeto.

Figura 2 – Sensor Higrométrico



Fonte: Proesi (2017).

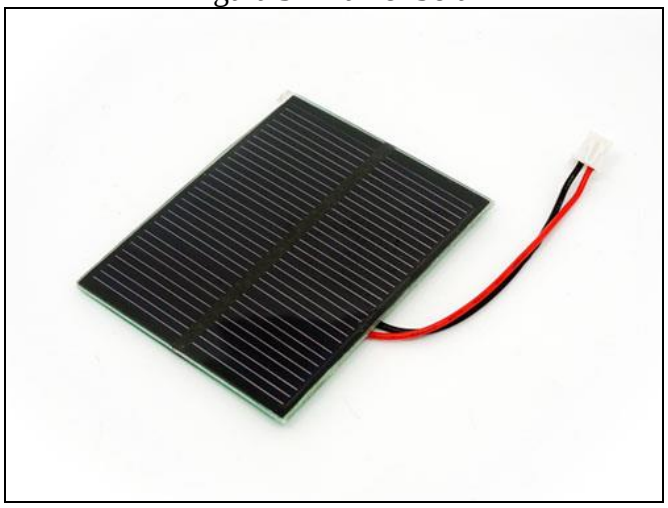
2.6 PAINEL SOLAR

Segundo Pozzebom (2013), painéis solares são, basicamente, dispositivos utilizados para converter a energia da luz do Sol em energia elétrica. O dispositivo também é conhecido como “Painel Solar Fotovoltaico”. A composição de um painel solar consiste em células fotovoltaicas, estas com a propriedade de ter sensibilidade de absorver a energia solar e gerar a eletricidade em duas camadas opostas.

Segundo Seeedstudio (2017), muitas pessoas gostam de energia solar porque é simples, limpa e renovável. Especialmente quando você está construindo projetos ao ar livre, a energia solar é uma das melhores soluções para o fornecimento de energia. Dependendo do projeto que você vai construir, há muitos tamanhos diferentes de painel solar para escolher.

A Figura 3 mostra o painel solar utilizado no projeto.

Figura 3 – Painel Solar



Fonte: Seeedstudio (2017).

2.7 REDES DE COMUNICAÇÃO

Nos últimos anos, houve um aumento na mobilidade. Devido a esse novo estilo de vida, os meios tradicionais de acesso à Internet se tornaram inadequados. Conexões sem fio surgiram para remover as restrições impostas pelas conexões cabeadas (GAST, 2005).

A expressão Wi-Fi significa Wireless Fidelity e é o nome da aliança reguladora dos padrões e protocolos usados na comunicação sem fio, bem como o nome deste meio de comunicação. A Wi-Fi Alliance é uma organização sem fins lucrativos, responsável por certificar a interoperabilidade dos dispositivos baseado no padrão 802.11 e os membros desta aliança são as empresas responsáveis por produzir os dispositivos 802.11 (DAVIS, 2004).

2.8 TRABALHOS CORRELATOS

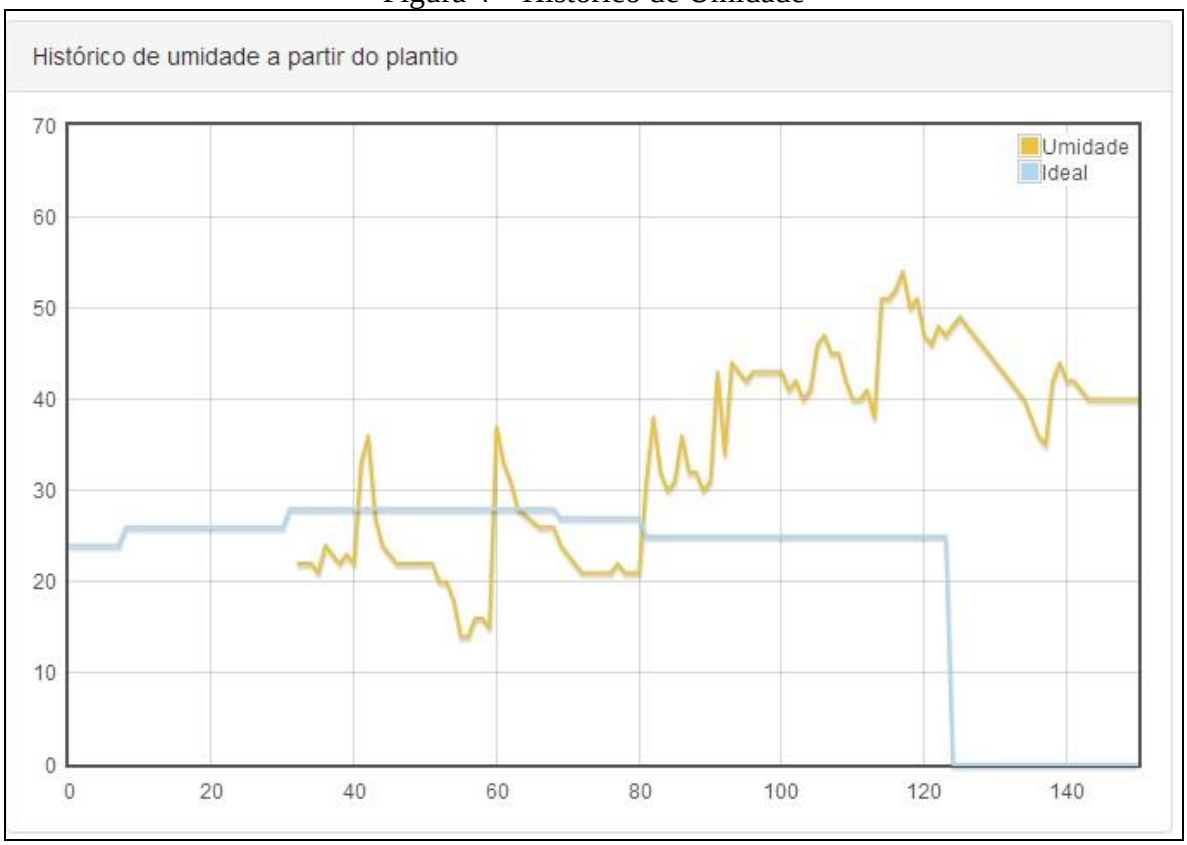
Foram selecionados dois trabalhos correlatos ao proposto. A seção 2.7.1 apresenta o trabalho de Pasieka (2014) que desenvolveu uma aplicação web para monitoramento dos dados coletados por uma rede de sensores sem fio em ambiente agrícola. Na seção 2.7.2 será abordado o Sistema Irriga, tecnologia desenvolvida pela UFSM que oferece um conjunto de serviços de manejo e monitoramento da irrigação, e por fim, no item 2.7.3, é demonstrada uma tabela comparativa em relação aos trabalhos correlatos.

2.8.1 Aplicação web para monitoramento dos dados coletados por rede de sensores sem fio em ambiente agrícola.

O trabalho de Pasieka (2014) tinha como objetivo construir uma aplicação web para que agricultores e agrônomos pudessem monitorar dados coletados por meio de sensores em uma área agrícola. Para alcançar seu objetivo Pasieka (2014) utilizou uma Rede de sensores sem fio (RSSF) para coletar os dados e armazenou num arquivo em formato Comma-separated values (CSV) em um computador interligado com a RSSF em conjunto com uma aplicação para realizar a leitura do arquivo CSV armazenando no banco de dados e uma aplicação web para monitorar on-line esses dados coletados possibilitando a tomada de decisão sobre quando e quanto irrigar uma determinada área com uma respectiva cultura.

Com a combinação do estudo, foi possível analisar os dados coletados e assim possibilitar as decisões de quando e quanto irrigar o solo. Na Figura 4 o exemplo demonstra a umidade coletada em alguns estádios.

Figura 4 – Histórico de Umidade



Fonte: Pasieka (2014).

2.8.2 Sistema Irriga

A proposta do Sistema Irriga é oferecer um conjunto de serviços de manejo e monitoramento da irrigação prático, eficiente e fácil de operar. Ele estima a necessidade diária de água de irrigação a ser aplicada em cada cultura no período de 24 e 48 horas. O sistema se ajusta adequadamente ao manejo da irrigação de culturas para atender a demanda climática e disponibilidade de água no solo para as plantas. Leva em conta a influência de todos os fatores do balanço hídrico, especialmente a demanda hídrica da cultura.

O servidor central do Sistema Irriga possui modelos e rotinas de programação para processar as informações de solo, planta, clima e sistema de irrigação de cada área irrigada. Recomenda com precisão quando e quanto de água aplicar em cada irrigação. De acordo com o Sistema Irriga (2017), ao seguir as recomendações do sistema, o produtor aumenta a produtividade da água e das culturas. Economiza água e energia e contribui de forma significativa para uma sustentabilidade ambiental.

2.8.3 Comparação entre os trabalhos correlatos

O Quadro 2 apresenta de forma comparativa algumas características em relação aos trabalhos correlatos.

Quadro 2 – Características dos trabalhos correlatos

Características	Aplicação web para monitoramento dos dados coletados por rede de sensores sem fio em ambiente agrícola	Sistema Irriga
Faz leitura de umidade solo	Sim	Sim
Possui uma rede de dispositivos para monitorar a lavoura	Sim	Sim
Possui uma aplicação que transmite as informações para o agricultor	Sim	Sim
Informa o agricultor a necessidade de irrigar a lavoura	Sim	Sim
Calcula o volume de água necessário para irrigação	Não	Sim

Fonte: elaborado pelo autor.

As aplicações são semelhantes na sua proposta, apresentando praticamente as mesmas características. Um diferencial significativo do Sistema Irriga é o fato dele poder determinar a quantidade de água que deve ser aplicada na próxima irrigação. Esta informação agrega muito valor ao produto, pois a economia de água é determinante tanto para um melhor resultado financeiro como para uma agricultura sustentável.

3 DESENVOLVIMENTO

As seções deste capítulo abordam os aspectos referentes à construção do projeto, demonstrando sucintamente as ferramentas e etapas utilizadas no seu desenvolvimento. Na seção 3.1 estão os requisitos funcionais e não funcionais e na seção 3.2 é apresentada a especificação detalhando o funcionamento do aplicativo. A seção 3.3 entre em detalhes da implementação, trazendo trechos de código fonte e na seção 3.4 é feita uma análise dos resultados obtidos.

3.1 REQUISITOS

Os requisitos do projeto são:

a) os dispositivos devem ser capazes de fazer leitura da umidade do solo (Requisito Funcional – RF);
b) o módulo ESP8266 deve armazenar em sua memória as leituras do solo (RF);
c) a aplicação deve demonstrar ao agricultor quando é necessário a irrigação da plantação (RF);
d) os dispositivos devem ser capazes de transmitir os registros das últimas leituras do solo para aplicação em caso de falta de conexão Wi-Fi (RF);
e) a comunicação dos dispositivos com a aplicação deve ser via Wi-Fi 802.11 (Requisito Não Funcional – RNF);
f) os dispositivos devem ser programados na IDE do Arduino (RNF);
g) a aplicação que exibe as informações ao agricultor deve ser programada em Hypertext Preprocessor (PHP) - (RNF);
h) a aplicação deve armazenar os dados no banco de dados MySQL (RNF);
i) para construção dos dispositivos utilizar a placa ESP8266 Thing (RNF).

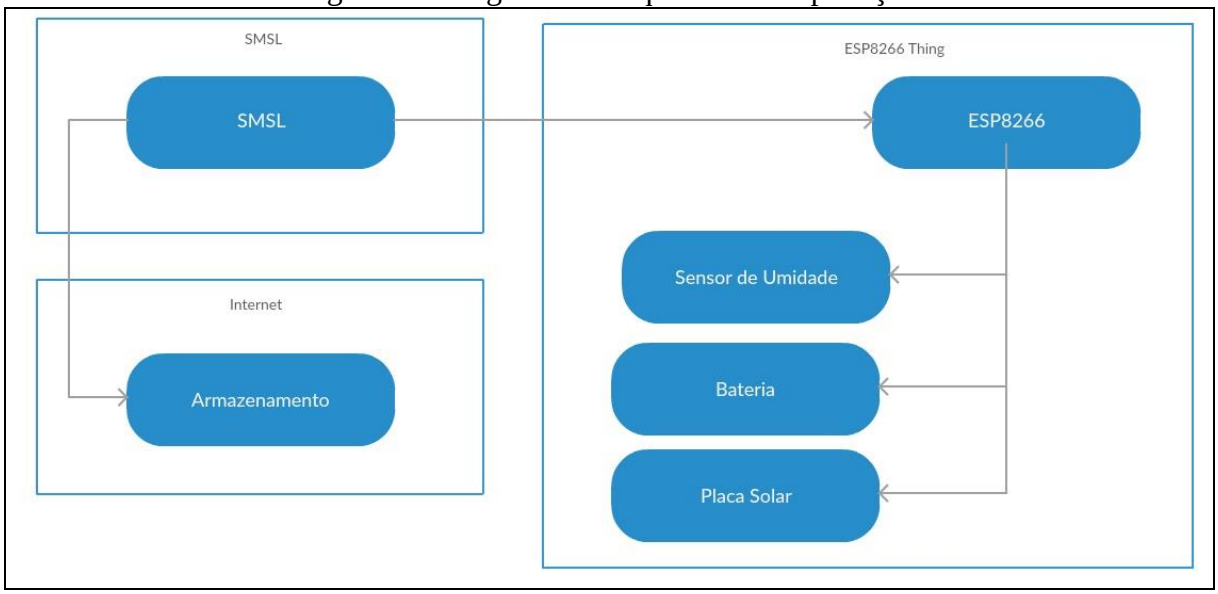
3.2 ESPECIFICAÇÃO

Para especificação foram elaborados o diagrama de arquitetura da aplicação, o diagrama de atividades e o diagrama esquemático. Os diagramas foram criados utilizando as ferramentas Creately para gerar o diagrama de arquitetura e de atividades e o Fritzing para gerar o diagrama esquemático.

3.2.1 Diagrama de arquitetura da aplicação

A Figura 5 exibe as principais características da arquitetura da aplicação.

Figura 5 – Diagrama de arquitetura da aplicação



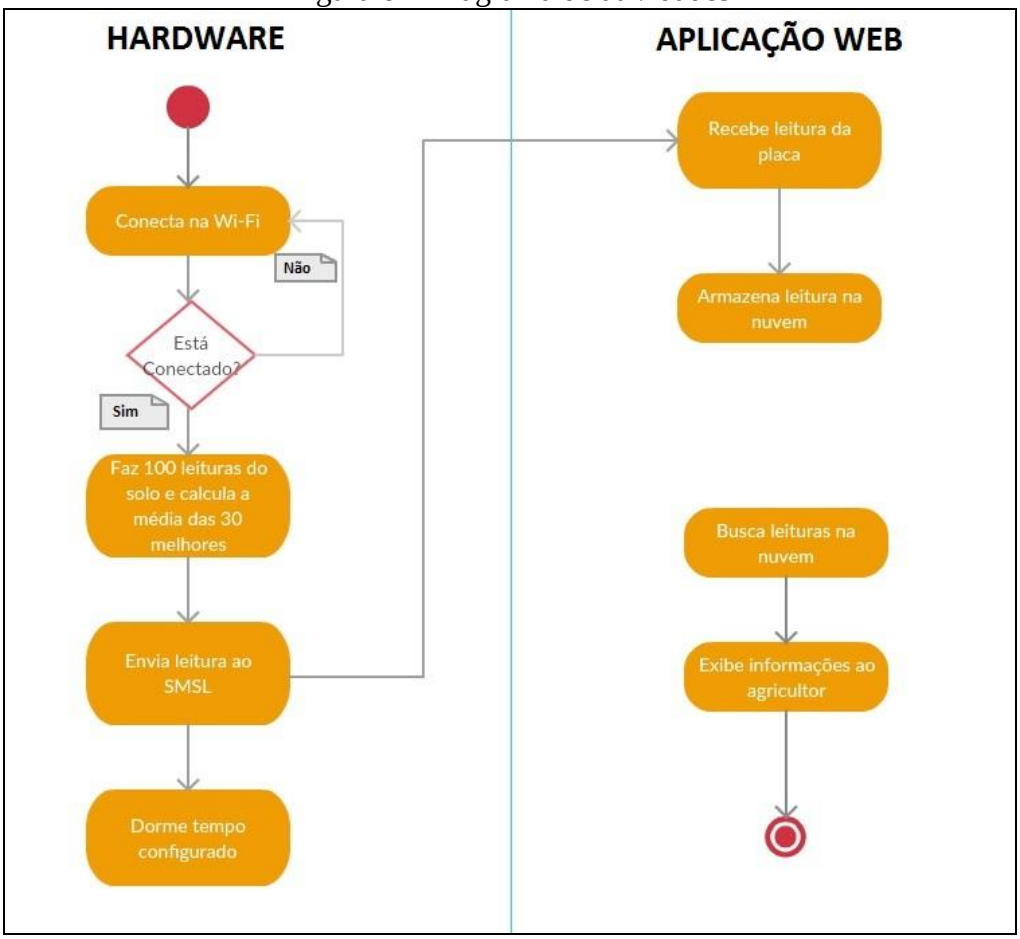
Fonte: elaborado pelo autor.

O projeto está composto por um ESP8266 Thing conectado num sensor higrométrico, uma bateria e um painel solar. A função do ESP8266 é coletar os dados do solo através do sensor e enviar para aplicação web através de conexão Wi-Fi por meio de protocolo Hypertext Transfer Protocol (HTTP). Todo equipamento é alimentado por uma bateria que é recarregada por energia solar através do painel solar. Já a aplicação tem a responsabilidade de armazenar os dados na nuvem e exibir as informações ao agricultor.

3.2.2 Diagrama de atividades

Na sequência é demonstrado na Figura 6 o diagrama de atividades realizadas pela aplicação.

Figura 6 – Diagrama de atividades



Fonte: elaborado pelo autor.

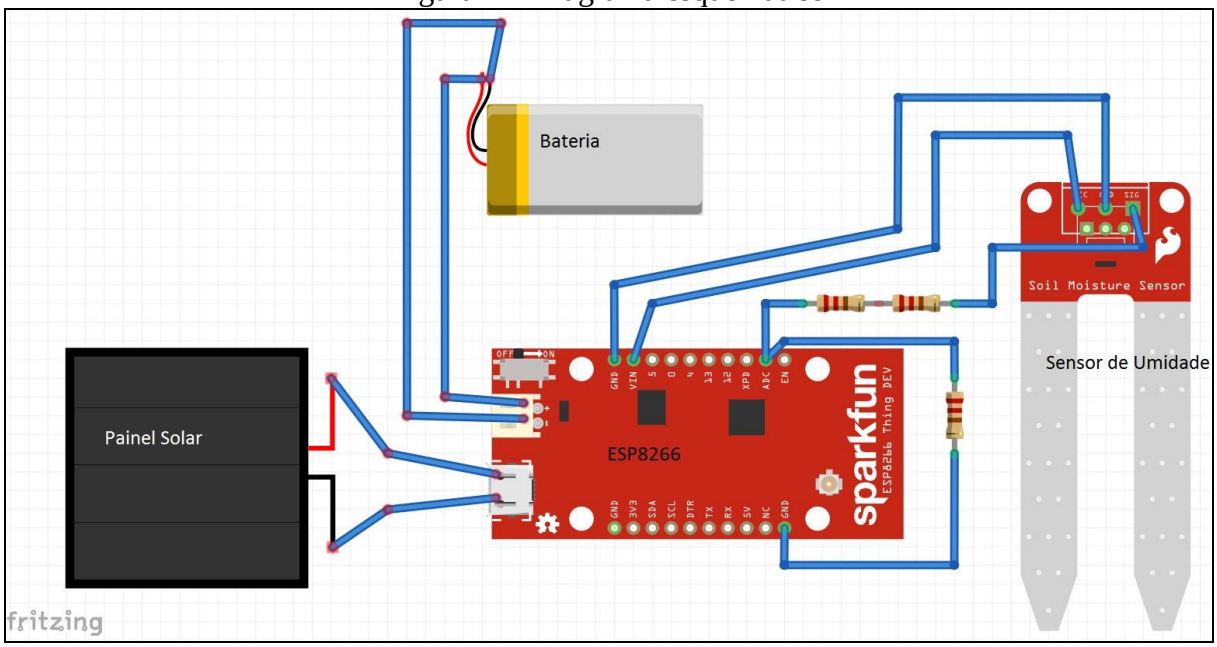
O diagrama mostra que o processo está dividido em duas partes: hardware e aplicação web. Cada parte tem uma função específica, sendo elas:

a) o hardware inicialmente tem a função de se conectar com a Wi-Fi e faz isso sucessivamente até obter sucesso. Vencendo esta etapa, ele inicia o processo de leitura do solo, tirando uma média das leituras para descartar picos que podem ocorrer na corrente elétrica. Em seguida ele envia as informações via protocolo Hypertext Transfer Protocol (HTTP) para aplicação web e por fim dorme o tempo que está configurado;
b) a aplicação web tem a responsabilidade de receber as leituras do solo do hardware e fazer a persistência dos dados no banco de dados, além disso também é responsável em buscar as últimas leituras e exibir em gráficos os resultados ao agricultor para que este possa tomar a decisão de irrigar a plantação.

3.2.3 Diagrama esquemático

Na Figura 7 é especificado o diagrama esquemático do protótipo construído para captura de informações do solo.

Figura 7 – Diagrama esquemático



Fonte: elaborado pelo autor.

O diagrama mostra quatro principais componentes, sendo eles: ESP8266, sensor higrométrico, painel solar e bateria. O ESP8266 é alimentado diretamente pela bateria através do conector de bateria de LiPo que já vem integrado na placa. É utilizado uma bateria de 3.7 volts com 400mAh e para seu carregamento um painel solar de 0.5W que está conectado no ESP8266 através do conector de carregamento de bateria LiPo, que também vem integrado na placa. O sensor higrométrico é alimentado através dos pinos GND e 5V do ESP8266 e as leituras são feitas através do pino ADC do ESP8266 utilizando resistores para montar um *voltage divider*[1], que diminui a tensão de 5V para 1V (tensão máxima da porta analógica).

3.3 IMPLEMENTAÇÃO

Esta seção foi dividida em quatro seções. Na seção 3.3.1 são mostrados informações da construção do hardware enquanto na seção 3.3.2 técnicas e ferramentas utilizadas no projeto. A seção 3.3.3 traz detalhes do código fonte tanto do software embarcado como da aplicação web e por fim na seção 3.3.4 é apresentada a operacionalidade da implementação.

---

[1] *voltage divider* é um circuito simples que transforma uma certa tensão em uma menor. Usando apenas dois resistores em série em uma tensão de entrada, é possível criar uma tensão de saída que é uma fração da entrada (SPARKFUN, 2017b).

3.3.1 Construção do Hardware

Na construção do hardware que vai ficar na plantação, foi utilizado um ESP8266 Thing, um sensor higrométrico, um painel solar, uma antena Wi-Fi e uma bateria de íon de lítio, além de fios, conectores, resistores, *jumpers*, ferro de solda e estanho. Na Tabela 1 estão disponíveis as peças e os periféricos utilizados incluindo o valor de cada elemento.

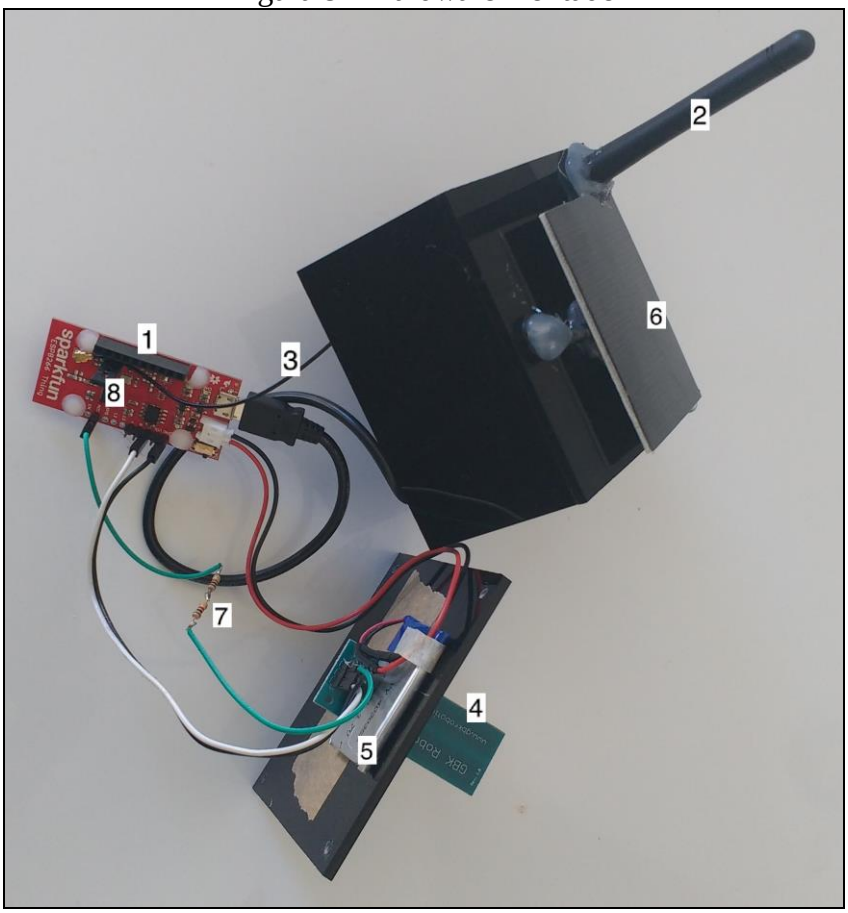
Tabela 1 – Relação dos componentes

Peça	**Valor (R$)**
SparkFun ESP8266 Thing	89,90
Antena Omni 3dBi 2.4GHz com Conector SMA	9,90
Cabo Pigtail U.FL com conector SMA 15cm	9,90
Sensor higrométrico	9,97
Lithium Ion Battery - 400mAh	16,03
0.5W Solar Panel 55x70	6,31
Resistor Carbono CR25 - 1/4W - 1K	0,02
Jumper Plástico MKBL02, com alça	0,15
Caixa de Acrílico	35,00
Ferro de Solda	38,06
Estanho/solda Cobix, tubinho 22 gramas	5,67

Fonte: elaborado pelo autor.

Na Figura 8 está o hardware depois de montado.

Figura 8 – Hardware montado



Fonte: elaborado pelo autor.

O item 1 é a placa ESP8266 Thing enquanto o item 2 é a antena que tem a responsabilidade de ampliar o raio de conexão Wi-Fi. O item 3 é o cabo que conecta a antena ao ESP8266, já o item 4 é o sensor higrométrico responsável em fazer a leitura do solo. O item 5 é a bateria que vai alimentar a placa e o item 6 é o painel solar que vai recarregar a bateria. O item 7 mostra alguns resistores que foram utilizados no circuito divisor de tensão que é detalhado na Figura 9 e por fim o item 8 é o jumper necessário para tornar possível a visualização da depuração no monitor serial do Arduino que está detalhado na Figura 10.

Na Figura 9 está disponível o cálculo do circuito divisor de tensão com resistores.

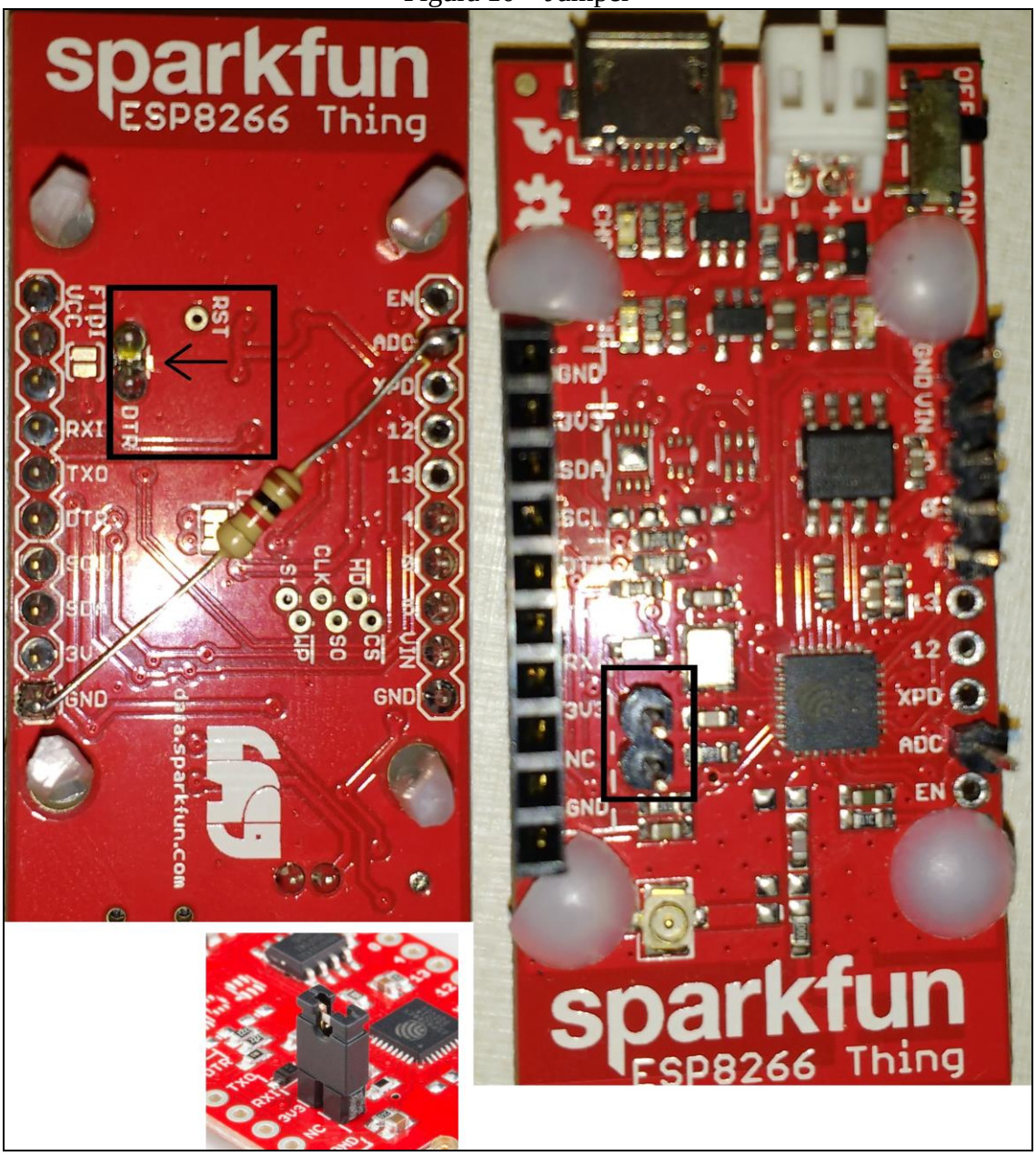
Figura 9 – Circuito divisor de tensão com resistores

Fonte: Arduino e Cia (2017).

A tensão máxima da porta analógica é 1V e o sensor envia 5V, então foi necessário calcular e adicionar um circuito divisor de tensão ao projeto. Para efetuar este cálculo foi utilizada a calculadora de divisão de tensão. Como tensão de entrada foi informado 5V, em R1 foi informado dois resistores em série para alcançar uma resistência de 2000 ohms e em R2 adicionado mais um resistor de 1000 ohms, chegando numa tensão de 1V.

A Figura 10 mostra detalhes do *jumper* necessário para tornar possível a visualização da depuração no monitor serial do Arduino.

Figura 10 – Jumper

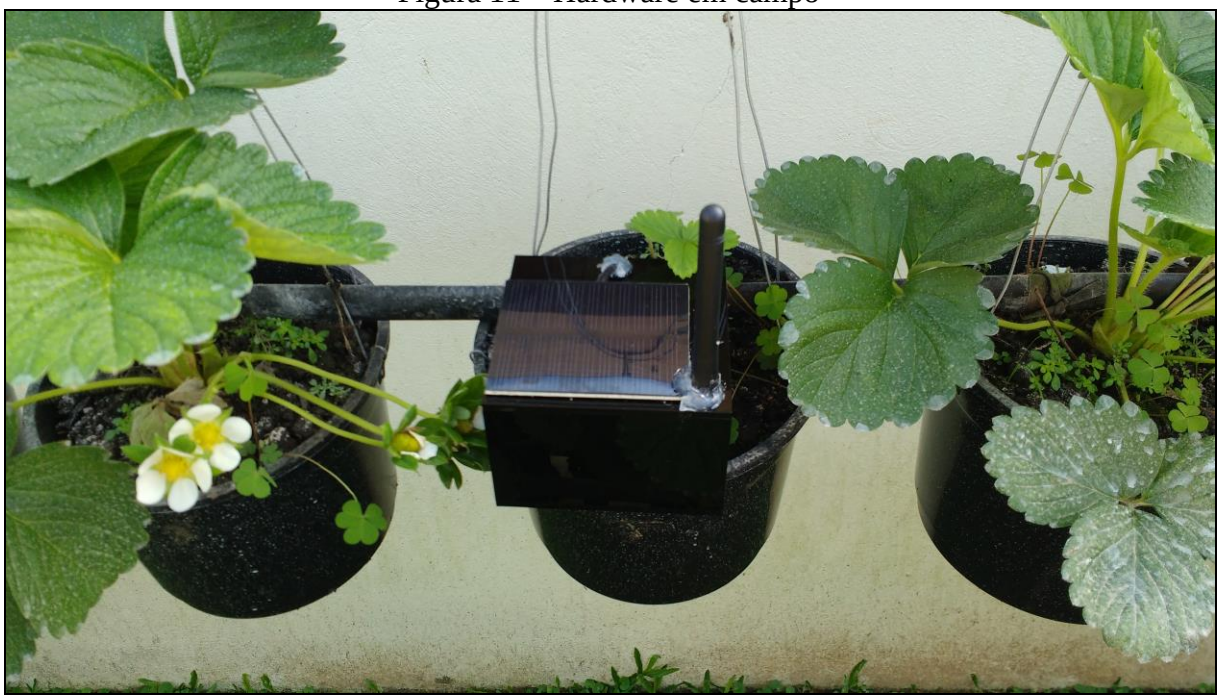


Fonte: elaborado pelo autor.

Segundo SparkFun (2017c), quando é iniciado o monitor serial do Arduino o ESP8266 entra no modo de programação ao invés de rodar o programa, o que impede de visualizar qualquer saída no console. Porém existe uma maneira de contornar isso, sendo necessário cortar a trilha DTR como demonstrado na imagem acima e adicionar um *jumper* sobre o DTR na parte inferior da placa. Sempre que o jumper estiver presente, a placa pode ser programada e a remoção do jumper vai habilitar o modo de terminal serial, tornando possível a visualização no monitor serial.

A Figura 11 apresenta o hardware em campo.

Figura 11 – Hardware em campo



Fonte: elaborado pelo autor.

3.3.2 Técnicas e ferramentas utilizadas

O trabalho foi desenvolvido na arquitetura cliente-servidor. A camada cliente é responsável por coletar os dados do solo e transmiti-los via Wi-Fi padrão 802.11, já a camada servidor processa e armazena os dados e permite ao usuário a consulta das informações para tomada de decisão. Para o desenvolvimento do cliente foi utilizada a Integrated Development Environment (IDE) Arduino e a linguagem de programação C e para o desenvolvimento do servidor foi utilizada a IDE JetBrains PhpStorm com a linguagem de programação PHP. O código fonte foi gerenciado pelo sistema de controle de versão Git, por meio de linha de comando.

Para construção da interface servidor, foram utilizadas as bibliotecas/*frameworks*: Doctrine ORM e Silex Framework. O Doctrine é uma biblioteca de relacionamentos de objetos com o banco de dados com sintaxe e uso semelhantes ao Hibernate e possui uma abstração de banco de dados, permitindo facilmente trabalhar com diversos bancos. O Silex é um micro *framework* PHP que permite uma rápida construção de aplicações web. O *framework* é baseado nas bibliotecas do Symfony Framework, garantindo acesso a funcionalidades comuns como: validação de dados que pode ser feita através de Annotations ou Extensible Markup Language (XML), serialização de dados que permite transformar dados em formatos XML e JavaScript Object Notation (JSON) para objetos e *arrays* PHP onde o

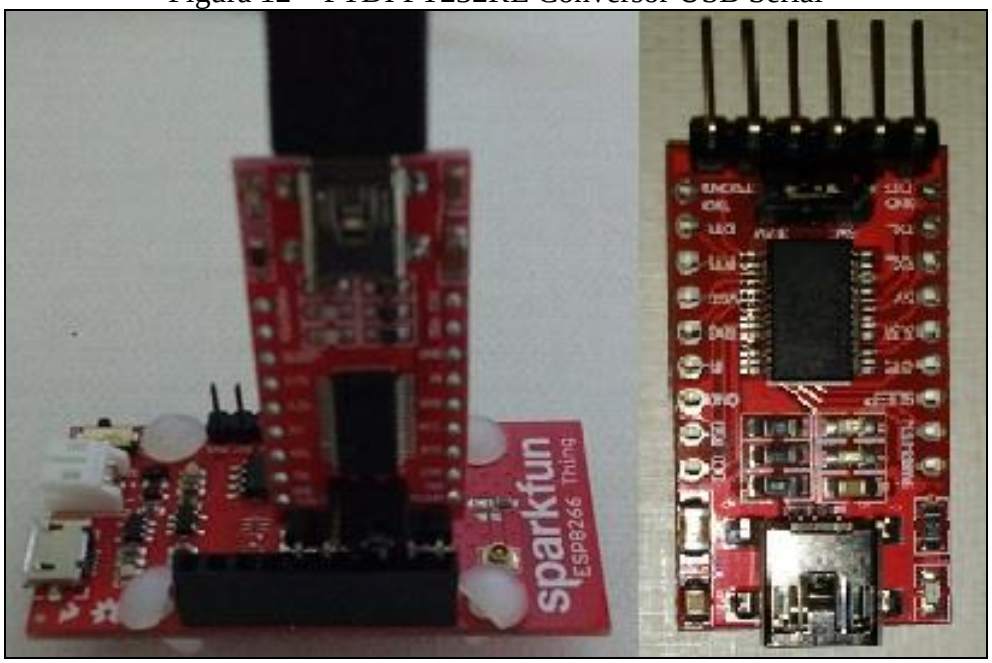
processo inverso também é possível e segurança da aplicação que permite implementar controles de acesso por Uniform Resource Locator (URL).

Além destas ferramentas foi utilizada uma técnica de programação chamada Front-Controller. Nesta técnica, por padrão as requisições web fornecem diretamente o recurso solicitado pelo cliente. Com um arquivo `.htaccess` e o componente de roteamento de requisições, as requisições são direcionadas para uma única página PHP, a qual irá fazer o carregamento da aplicação e realizar os processamentos posteriores. A principal vantagem desta técnica é forçar todas as requisições a cumprirem com algum requisito, como por exemplo, solicitar que o usuário esteja autenticado através de HTTP *Basic*. Outra vantagem é garantir um fluxo claro na aplicação, permitindo que seja feito todo o carregamento da aplicação em um ponto central, deixando os demais controladores apenas com as regras que realmente precisem se preocupar.

Uma segunda técnica utilizada na construção foi o padrão de projetos Model View Controller (MVC). Este padrão nos permite dar uma clara organização ao projeto, deixando cada arquivo em sua respectiva localidade, e permitindo que os componentes tenham suas responsabilidades bem definidas. Para finalizar, foi utilizada a técnica de Thin-Controller & Fat-Model. Está técnica consiste em trazer o máximo de informações de negócio possíveis para dentro do objeto, deixando as classes controladoras com uma lógica mais simples. A vantagem disso é que facilmente conseguimos reutilizar os objetos em demais locais do projeto, sem necessidade de se preocupar se está realizando todas as validações necessárias ou interações com o banco de dados.

Para escrever o código do servidor foi utilizado o *framework* do Arduino e as bibliotecas: FS.h para fazer o gerenciamento de arquivos, ESP8266WiFi.h para utilizar a conexão Wi-Fi do ESP8266, ESP8266HTTPClient.h para utilizar requisições HTTP e StringSplitter.h para trabalhar com variáveis do tipo *String*. A compilação e transferência do código para o hardware era feita através da IDE do Arduino. A conexão entre o computador e o ESP8266 foi feita através de cabo Universal Serial Bus (USB) e uma placa auxiliar Future Technology Devices International (FTDI) FT232RL Conversor USB Serial que é demonstrada na Figura 12. Para depuração do código era utilizada e IDE Arduino e foi necessário cortar a trilha do DTR da placa para soldar um *jumper* que era removido para tornar possível a visualização da depuração no Monitor Serial do Arduino.

Figura 12 – FTDI FT232RL Conversor USB Serial

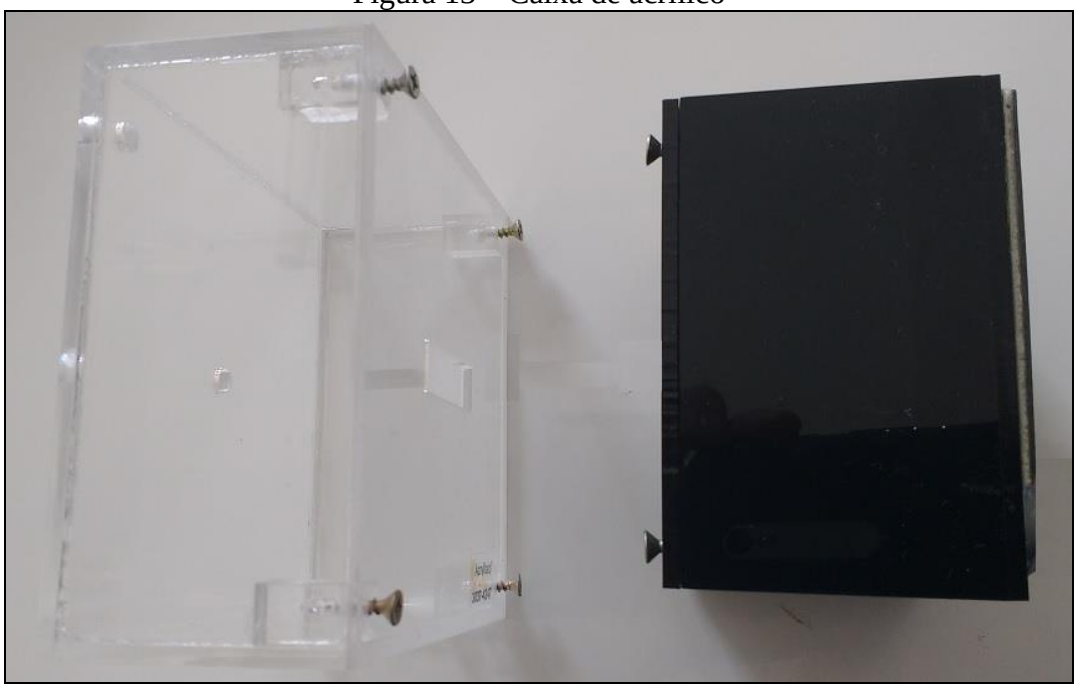


Fonte: Filipeflop (2017).

Para programar o ESP8266 Thing é necessário um conversor USB para Serial. Neste projeto foi utilizado o FT232RL que possui seis pinos que coincidem exatamente com o que o ESP8266 oferece. Neste caso bastou conectar o conversor serial a placa e na porta USB e iniciar sua programação.

Para acondicionar o hardware em campo, foi necessário construir uma caixa de acrílico como mostra a Figura 13.

Figura 13 – Caixa de acrílico



Fonte: elaborado pelo autor.

Inicialmente como protótipo, foi construída uma caixa transparente com dimensões maiores como mostra o lado esquerdo da Figura 13, mas ela foi aperfeiçoada com dimensões menores, além de utilizar a cor preta para bloquear os raios solares afim de evitar aquecimento do hardware. Depois que o hardware foi acondicionado da caixa de acrílico, ela foi lacrada utilizando cola quente, para evitar infiltração.

3.3.3 Código Fonte

Nesta seção serão apresentados os principais trechos de código da aplicação, divididos entre hardware e aplicação web.

3.3.3.1 Hardware

Ao iniciar seu funcionamento o hardware passa por uma função denominada `setup`, conforme Quadro 3.

Quadro 3 – Fase inicial do hardware

```
40 void setup() {
41   pinMode(pino_alimentacao_sensor, OUTPUT);
42   Serial.begin(9600);
43   os_timer_setfn(&mTimer, tCallback, NULL);
44   os_timer_arm(&mTimer, 1000, true);
45   //Abre o sistema de arquivos (mount)
46   openFS();
47   bool conectou = false;
48   if (SPIFFS.exists(configFile)) {
49     readConfigFile();
50
51     int str_len = ssidConfigFile.length();
52     char ssid_array[str_len];
53     memset(ssid_array, '\0', sizeof(ssid_array));
54     ssidConfigFile.toCharArray(ssid_array, str_len);
55
56     str_len = passConfigFile.length();
57     char pass_array[str_len];
58     memset(pass_array, '\0', sizeof(pass_array));
59     passConfigFile.toCharArray(pass_array, str_len);
60
61     if (!conectaWiFi(ssid_array, pass_array)) {
62       conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
63       while (!conectou) {
64         conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
65       }
66     }
67   } else {
68     conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
69     while (!conectou) {
70       conectou = conectaWiFi(SSID_REDE, SENHA_REDE);
71     }
72   }
73 }
```

Fonte: elaborado pelo autor.

Nesta função, nas linhas 43 e 44 é inicializado o relógio interno que será utilizado para saber o horário correto de leitura do solo, na linha 46 o sistema de arquivos que será utilizado para gravar as configurações e as leituras e por fim entre as linhas 48 e 72 é iniciado um processo de tentativa de conexão Wi-Fi que tenta conectar na rede do arquivo de configuração que contém a rede que o agricultor define na página de configuração da aplicação web, quando este existir ou tenta conectar na rede padrão.

Para conexão Wi-Fi foi criada a função `conectaWiFi`, disponível no Quadro 4.

Quadro 4 – Conectando na Wi-Fi

```
137 bool conectaWiFi(char* ssid, char* pass) {
138   client.stop();
139   Serial.println("Conectando-se a rede WiFi.");
140   Serial.println();
141   delay(1000);
142   WiFi.mode(WIFI_STA);
143   WiFi.begin(ssid, pass);
144   int tentativas = 0;
145   while (WiFi.status() != WL_CONNECTED && tentativas < 100) {
146     delay(500);
147     Serial.print(".");
148     tentativas++;
149   }
150   Serial.println("");
151   if (WiFi.status() == WL_CONNECTED) {
152     Serial.println("WiFi connectado com sucesso");
153     Serial.println("IP obtido: ");
154     Serial.println(WiFi.localIP());
155     if (SPIFFS.exists(configFile)) {
156       deleteFile(configFile);
157     }
158     createFile(configFile);
159     String configs = buscaConfig();
160     StringSplitter *splitter = new StringSplitter(configs, ';', 4);
161     writeFile(configFile, splitter->getItemAtIndex(0));
162     writeFile(configFile, splitter->getItemAtIndex(1));
163     writeFile(configFile, splitter->getItemAtIndex(2));
164     String datetime = splitter->getItemAtIndex(3);
165     splitter = new StringSplitter(datetime, ';', 2);
166     String data = splitter->getItemAtIndex(0); //data
167     String hora = splitter->getItemAtIndex(1); //hora
168     splitter = new StringSplitter(data, '-', 3);
169     ano = splitter->getItemAtIndex(0).toInt();
170     mes = splitter->getItemAtIndex(1).toInt();
171     noInterrupts();
172     dia = splitter->getItemAtIndex(2).toInt();
173     splitter = new StringSplitter(hora, ':', 3);
174     horas = splitter->getItemAtIndex(0).toInt();
175     minutos = splitter->getItemAtIndex(1).toInt();
176     segundos = splitter->getItemAtIndex(2).toInt();
177     interrupts();
178     return true;
179   }
180   return false;
181 }
```

Fonte: elaborado pelo autor.

Esta função solicita como parâmetros o nome e a senha da rede em que se deseja conectar e retorna verdadeiro ou falso para informar se houve ou não sucesso na conexão. Na linha 145 são executadas cem tentativas de conexão na rede que foi passada como parâmetro. Se esta conexão for possível, então entre a linha 155 e 163 o arquivo de configurações é atualizado com os dados da página de configuração da aplicação web e também entre a linha 164 e 176 a data e o horário da placa.

Após essa etapa de inicialização que é executada somente uma vez, o hardware entra na função `loop` disponível no Quadro 5.

Quadro 5 – Função principal

```
75 void loop() {
76
77   // só manda energia para o sensor quando vai utilizar ele, para ser mais eficiente.
78   digitalWrite(pino_alimentacao_sensor, HIGH);
79
80   // faz 100 leituras da tensão que o pino analógico esta recebendo. Esta tensão pode variar de 0v até 1v
81   // sendo 0v considerado sem umidade e 1v totalmente umido.
82   int reads[100];
83   for (int i = 0; i < 100; i++) {
84     reads[i] = analogRead(pino_sinal_analogico);
85   }
86
87   // ordena o array de leituras
88   for (int i = 0; i < 100; i++) {
89     for (int j = 1; j < (100 - i); j++) {
90       if (reads[j - 1] > reads[j]) {
91         int temp = reads[j - 1];
92         reads[j - 1] = reads[j];
93         reads[j] = temp;
94       }
95     }
96   }
97
98   // descarta as 35 maiores e as 35 menores leituras
99   // soma as 30 melhores leituras e divide pela quantidade de leituras para ter o valor mais exato da umidade.
100  int analogValue = 0;
101  for (int i = 35; i < 65; i++) {
102    analogValue += reads[i];
103  }
104  analogValue = analogValue / 30;
105
106  char leituraSolo[90];
107  sprintf(leituraSolo, "{\"valorLido\": %d,\"dispositivo\": %d, \"dataLeitura\": \"%d-%02d-%02d %02d:%02d:%02d\"}",
108  if (SPIFFS.exists(leiturasFile)) {
109    writeFile(leiturasFile, leituraSolo);
110  } else {
111    createFile(leiturasFile);
112    writeFile(leiturasFile, leituraSolo);
113  }
114
115  readConfigFile();
116
117  int str_len = ssidConfigFile.length();
118  char ssid_array[str_len];
119  memset(ssid_array, '\0', sizeof(ssid_array));
120  ssidConfigFile.toCharArray(ssid_array, str_len);
121
122  str_len = passConfigFile.length();
123  char pass_array[str_len];
124  memset(pass_array, '\0', sizeof(pass_array));
125  passConfigFile.toCharArray(pass_array, str_len);
126
127  if (conectaWiFi(ssid_array, pass_array)) {
128    enviaInformacao();
129  }
130
131  // desliga alimentação do pino
132  digitalWrite(pino_alimentacao_sensor, LOW);
133
134  delay(tempoConfigFile.toInt());
135 }
```

Fonte: elaborado pelo autor.

Na função `loop` na linha 78 é alimentado o sensor para que o consumo de energia seja mais eficiente. Após essa instrução entre as linhas 82 e 85 é iniciado um processo de cem leituras da tensão que o pino analógico está recebendo. Esta tensão pode variar de 0v até 1v,

sendo 0v considerado sem umidade e 1v totalmente úmido. Em seguida entre as linhas 88 e 96 é feita uma ordenação destas leituras de forma crescente para que entre as linhas 100 e 103 sejam descartadas as 35 maiores e as 35 menores leituras e por fim na linha 104 é feito uma média das 30 leituras restantes para chegar ao valor mais aproximado de umidade do solo. No próximo passo é gravar no sistema de arquivos no arquivo leituras.txt o valor lido, instrução que é executada entre as linhas 108 e 113 e depois é feita a chamada da função `conectaWiFi` na linha 127. Caso a tentativa de conexão obtenha sucesso, na linha 128 é feita a chamada da função `enviaInformação` disponível no Quadro 6. Na linha 134 esta a última instrução, onde a placa dorme o tempo que foi definido na página de configuração da aplicação.

Quadro 6 – Enviando informação para aplicação

```
192 // envia todas as leituras que estão armazenadas. normalmente só vai ter a ultima leitura.
193 void enviaInformacao() {
194   File xFile = SPIFFS.open(leiturasFile, "r");
195   bool enviado = false;
196   while (xFile.available()) {
197     String leitura = xFile.readStringUntil('\n');
198     Serial.println(leitura);
199     client.connect(host, httpPort);
200     if (client.connected()) {
201       client.println("POST /web/index.php/inserir HTTP/1.1");
202       client.println("Host: smsl.jeanhertel.com.br");
203       client.println("Authorization: Basic YWRtaW46Zm9v");
204       client.println("Connection: close");
205       client.println("Content-Type: application/json");
206       client.print("Content-Length: ");
207       client.println(leitura.length());
208       client.println();
209       client.println(leitura);
210       Serial.println("Informacoes enviadas para o SMSL!");
211       enviado = true;
212     } else {
213       Serial.println("Desconectado do SMSL");
214       return;
215     }
216     client.stop();
217   }
218   xFile.close();
219   if (enviado) {
220     deleteFile(leiturasFile);
221   }
222 }
```

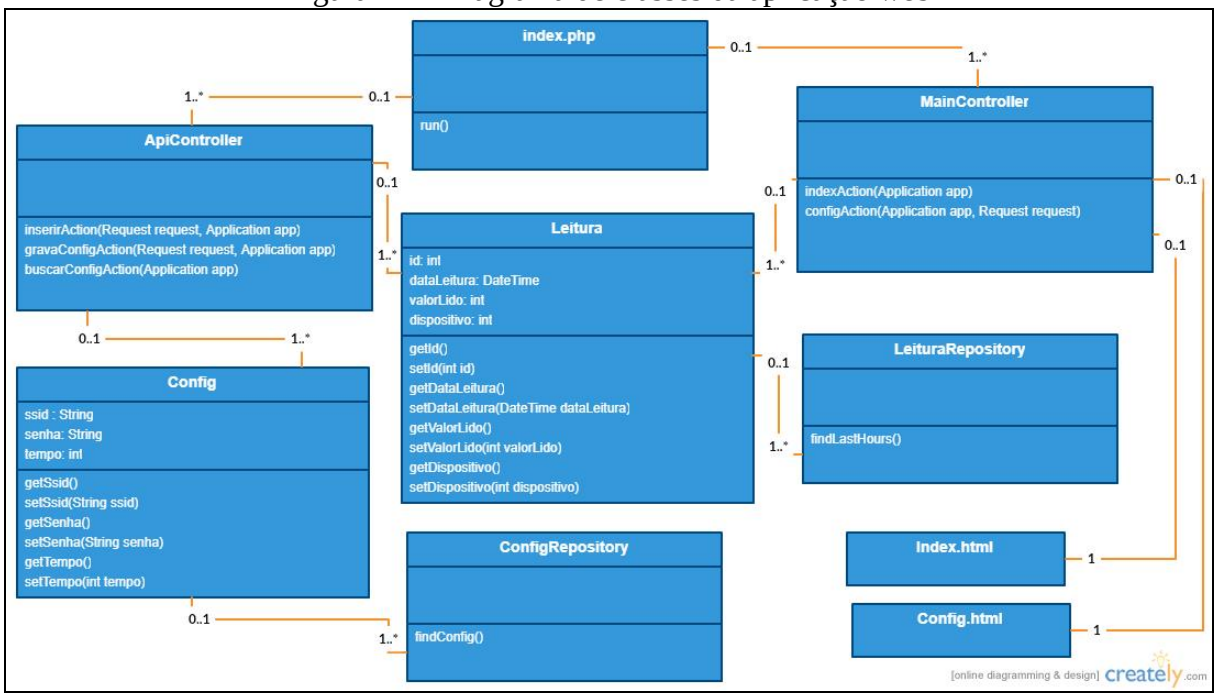
Fonte: elaborado pelo autor.

O objetivo desta função é percorrer o arquivo de leituras leituras.txt entre as linhas 194 e 196 e para cada leitura encontrada entre as linhas 201 e 209 enviar uma requisição HTTP para a aplicação web informando o valor da leitura. O formato dos pacotes enviados é {"valorLido": 0,"dispositivo": 2, "dataLeitura": "2017-06-18 10:35:13"}.

3.3.3.2 Aplicação Web

Para auxiliar na compreensão do código fonte da aplicação web, foi elaborado na ferramenta Creately o diagrama de classes disponível Figura 14.

Figura 14 – Diagrama de classes da aplicação web



Fonte: elaborado pelo autor.

Na aplicação web, tudo começa pela classe `index.php`. Qualquer requisição HTTP feita para aplicação vai para o `index.php` e ele é responsável por encaminhar para sua respectiva chamada conforme Quadro 7.

Quadro 7 – Classe index.php

```
1  <?php
2
3  require "../setup.php";
4
5  $app->get('/', "App\\Controller\\MainController::indexAction")->bind( routeName: 'index');
6  $app->get('/config', "App\\Controller\\MainController::configAction")->bind( routeName: "config_view");
7  $app->post( pattern: '/inserir',  to: "App\\Controller\\ApiController::inserirAction");
8  $app->post( pattern: '/gravar',  to: "App\\Controller\\ApiController::gravaConfigAction")->bind( routeName: 'gravar_config');
9  $app->get('/buscar', "App\\Controller\\ApiController::buscarConfigAction")->bind( routeName: 'buscar_config');
10
11 $app->run();
```

Fonte: elaborado pelo autor.

Foram escritas várias chamadas na classe `index.php`. A chamada `index` da linha 5 envia a requisição para a função `indexAction` da classe `MainController` que é responsável em carregar o HTML da página principal da aplicação, já a chamada `config_view` da linha 6 envia a requisição para função `configAction` da classe `MainController` que tem responsabilidade de carregar a página de configuração da aplicação. A chamada `inserirAction` da linha 7 é responsável por registrar as leituras do hardware no banco de dados. Também está disponível a chamada `gravar_config` na linha 8, que registra as

configurações da página de configurações e por fim a chamada `buscar_config` na linha 9, que retorna as configurações registradas no banco de dados para quem fizer sua requisição.

Em seguida, são demonstrados detalhes de duas classes importantes da aplicação, que são o `ApiController.php` e o `MainController.php`.

No Quadro 8 estão detalhes do `ApiController.php`.

Quadro 8 – Classe ApiController

```
1   <?php
2
3   namespace App\Controller;
4
5   use ...
14
15  class ApiController
16  {
17      public function inserirAction(Request $request, Application $app)
18      {
19          $bodyRequest = $request->getContent();
20
21          /** @var Leitura $leitura */
22          $leitura = $app['serializer']->deserialize($bodyRequest, type: Leitura::class, format: 'json', ['groups' => ['insercao']]);
23
24          /** Ajusta a data que pode estar incorreta */
25          if(!($leitura->getDataLeitura() instanceof \DateTime)) {
26              $data = $leitura->getDataLeitura();
27              $dataReal = \DateTime::createFromFormat( format: 'Y-m-d H:i:s', $data);
28              $leitura->setDataLeitura($dataReal);
29          }
30
31          $app['orm.em']->persist($leitura);
32          $app['orm.em']->flush();
33
34          return new Response( content: "Salvo com sucesso");
35      }
36
37      public function gravaConfigAction(Request $request, Application $app)
38      {
39          /** @var ConfigRepository $configRepository */
40          $configRepository = $app['orm.em']->getRepository( entityName: 'App\\Entity\\Config');
41          $config = $configRepository->findConfig();
42
43          $config->setSsid($request->request->get( key: "ssid"));
44
45          $senhaNova = $request->request->get( key: "senha");
46          if($senhaNova != '') {
47              $config->setSenha($senhaNova);
48          }
49
50          $config->setTempo($request->request->getInt( key: "tempo"));
51
52          $app['orm.em']->flush();
53
54          $sessao = $request->getSession();
55          $sessao->set('mensagem', "Salvo com sucesso!");
56
57          return new RedirectResponse($app['url_generator']->generate( name: 'config_view'));
58      }
59
60      public function buscarConfigAction(Application $app) {
61          /** @var ConfigRepository $configRepository */
62          $configRepository = $app['orm.em']->getRepository( entityName: 'App\\Entity\\Config');
63          $config = $configRepository->findConfig();
64
65          $data = new DateTime( time: "now", new DateTimeZone( timezone: "America/Sao_Paulo"));
66
67          $retorno = $config->getSsid() . ";" . $config->getSenha() . ";" . $config->getTempo() . ";" . $data->format( format: "Y-m-d;H
68
69          return new Response($retorno);
70      }
71  }
```

Fonte: elaborado pelo autor.

Nesta classe estão disponíveis as funções `inserirAction` na linha 17 que recebe dados de leitura do solo e faz a persistência da informação no banco de dados nas linhas 31 e 32, a função `gravaConfigAction` na linha 37 que recebe que recebe as configurações da

página de configurações e faz a persistência no banco de dados nas linhas 54 e 55 e por fim a função `buscarConfigAction` na linha 60 que retorna as configurações do banco de dados na linha 69 para quem solicitar.

No Quadro 9 estão detalhes da classe `MainController.php`.

Quadro 9 – Classe MainController

```
1  <?php
2
3  namespace App\Controller;
4
5  use ...
11
12 class MainController
13 {
14     public function indexAction(Application $app)
15     {
16         $leituraRepository = $app['orm.em']->getRepository( entityName: 'App\\Entity\\Leitura');
17
18         $dispositivos[] = $leituraRepository->findLastHours(1);
19         $dispositivos[] = $leituraRepository->findLastHours(2);
20         $dispositivos[] = $leituraRepository->findLastHours(3);
21
22         return new Response($app['twig']->render( name: 'index.html.twig', [
23             'dispositivo1' => $dispositivos[0],
24             'dispositivo2' => $dispositivos[1],
25             'dispositivo3' => $dispositivos[2]
26         ]));
27     }
28
29     public function configAction(Application $app, Request $request)
30     {
31         /** @var ConfigRepository $configRepository */
32         $configRepository = $app['orm.em']->getRepository( entityName: 'App\\Entity\\Config');
33         $config = $configRepository->findConfig();
34
35         $mensagem = "";
36
37         $sessao = $request->getSession();
38         if($sessao->has( name: 'mensagem')) {
39             $mensagem = $sessao->get( name: 'mensagem');
40             $sessao->remove( name: 'mensagem');
41         }
42
43         return new Response($app['twig']->render( name: 'config.html.twig', [
44             'config' => $config,
45             'mensagem' => $mensagem
46         ]));
47     }
48
49 }
```

Fonte: elaborado pelo autor.

Esta classe é composta por duas funções, sendo elas o `indexAction` na linha 14 que tem a responsabilidade de buscar as últimas leituras do solo entre as linhas 18 e 20 e fazer a chamada do HTML da página principal da aplicação na linha 22 e o `configAction` na linha 29 que faz a leituras das configurações na linha 33 e faz a chamada do HTML da página de configurações na linha 43.

No Quadro 10 estão as classes `Config.php` e `Leitura.php` que são responsáveis em modelar os elementos que serão persistidos no banco de dados.

Quadro 10 – Classe Config.php e Leitura.php

```php
<?php

namespace App\Entity;

use ...

/**
 * @ORM\Entity(repositoryClass="App\Repository\Co
 * @ORM\Table(name="config")
 */
class Config
{
    /**
     * @var \String
     *
     * @ORM\Id()
     * @ORM\Column(name="ssid", type="string")
     */
    private $ssid;

    /**
     * @var \String
     *
     * @ORM\Column(name="senha", type="string")
     */
    private $senha;

    /**
     * @var \int
     *
     * @ORM\Column(name="tempo", type="integer")
     */
    private $tempo;
```

```php
<?php

namespace App\Entity;

use ...

/**
 * @ORM\Entity(repositoryClass="App\Repository\LeituraRepository"
 * @ORM\Table(name="leitura")
 */
class Leitura
{
    /**
     * @var int
     *
     * @ORM\Id()
     * @ORM\GeneratedValue(strategy="AUTO")
     * @ORM\Column(name="id", type="integer", nullable=false)
     */
    private $id;

    /**
     * @var \DateTime
     *
     * @ORM\Column(name="data_leitura", type="datetime", nullable
     *
     * @Groups({"insercao"})
     */
    private $dataLeitura;

    /**
     * @var int
     *
     * @ORM\Column(name="valor_lido", type="integer", nullable=fa
     *
     * @Groups({"insercao"})
```

Fonte: elaborado pelo autor.

Nestas classes são modeladas as leituras do solo e as configurações, determinando quais seus atributos, como por exemplo o horário em que a leitura foi executada na linha 30 da classe Leitura. São elementos desta estrutura que serão persistidos no banco de dados.

O Quadro 11 mostra as classes de consulta ao banco de dados.

Quadro 11 – Classe ConfigRepository.php e LeituraRepository.php

```php
<?php

namespace App\Repository;

use App\Entity\Config;

class ConfigRepository extends \Doctrine\ORM\EntityRe
{
    /**
     * @return Config
     */
    public function findConfig()
    {
        $qb = $this->createQueryBuilder( alias: 'a');

        return $qb
            ->setFirstResult(0)
            ->setMaxResults(1)
            ->getQuery()
            ->getOneOrNullResult();
    }
}
```

```php
<?php

namespace App\Repository;

class LeituraRepository extends \Doctrine\ORM\EntityRepositor
{
    public function findLastHours($dispositivo)
    {
        $retorno = $this
            ->createQueryBuilder( alias: 'a')
            ->where( predicates: 'a.dispositivo = :dispositivo')
            ->setParameter( key: 'dispositivo', $dispositivo)
            ->setFirstResult(0)
            ->setMaxResults(24)
            ->orderBy( sort: 'a.dataLeitura', order: 'DESC')
            ->getQuery()
            ->getArrayResult();

        return array_reverse($retorno);
    }
}
```

Fonte: elaborado pelo autor.

Nestas classes estão disponíveis os métodos que fazem consultas ao banco de dados. Na classe `ConfigRepository` o método `findConfig` na linha 12 retorna as configurações da página de configurações do SMSL, já na classe LeituraRepository o método `findLastHours` na linha 7 busca as ultimas 24 leituras realizadas pelo dispositivo solicitado.

As páginas HTML foram construídas utilizando uma biblioteca denominada PURE.CSS (2017). Juntamente com esta biblioteca, foi utilizado um arquivo denominado `customizacao.css` para adicionar os demais detalhes das telas e para geração dos gráficos foi utilizado a biblioteca CHART.JS (2017). Por fim, para garantir a dinâmica das páginas foi utilizada uma biblioteca chamada TWIG (2017) que permite misturar o código HTML com a linguagem de *template*, mantendo o código bastante simples e legível.

No Quadro 12 e Quadro 13 é detalhada a página principal da aplicação: `index.html.twig`.

Quadro 12 – Início do index.html.twig

```
1  <!doctype html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6
7      <title>SMSL</title>
8
9      <link rel="stylesheet" type="text/css" href="/web/css/pure-min.css" />
10     <link rel="stylesheet" type="text/css" href="/web/css/customizacao.css" />
11
12     <script type="text/javascript" src="/web/js/Chart.min.js"></script>
13
14 </head>
15 <body>
16
17 <div class="header-container" style="...">
18     <div class="container">
19         <div class="header-logo">
20             <div class="pure-menu pure-menu-horizontal">
21                 <a href="{{ path('config_view') }}">Configuração</a>
22             </div>
23         </div>
24     </div>
25
26     <div class="container">
27         <div class="pure-u-1-1 text-center">
28             <h1 class="header-title">Sistema de Monitoramento de Solo e Lavoura</h1>
29         </div>
30     </div>
31 </div>
32
33 <div class="container content text-center">
34
35 {% if dispositivo1|length > 0 %}
36     <div style="...">
37         <canvas id="meuCanvas1" width="600" height="400"></canvas>
38
39         {% set ultimoDispositivo1 = dispositivo1|last %}
40         <div>
41             <div style="...">
42                 <div>
43                     <span class="umidade umidade-baixa {% if ultimoDispositivo1.valorLido >= 0 and ultimoDispositivo1.valorLido < 350
44                     <span class="umidade umidade-moderada {% if ultimoDispositivo1.valorLido >= 350 and ultimoDispositivo1.valorLido
45                     <span class="umidade umidade-alta {% if ultimoDispositivo1.valorLido >= 700 %}active{% endif %}">Alta</span>
46                 </div>
47             </div>
48             <div style="..."></div>
49         </div>
50     </div>
51 {% endif %}
52
```

Fonte: elaborado pelo autor.

O `index.html.twig` possui cabeçalho padrão HTML entre as linhas 3 e 14 com a inclusão dos arquivos CSS na linha 9/10 e Javascript na linha 12 necessários. No seu corpo foi adicionada uma divisão na linha 17 para exibir o título da página, juntamente com uma imagem ilustrativa e depois é verificado na linha 35 com linguagem de *template* se cada dispositivo tem dados disponíveis. Para cada dispositivo com informações a disposição é criado um *container* para o gráfico com a *tag canvas*. Por fim é avaliada com linguagem de *template* entre as linhas 43 e 45 a posição que deve ser desenhada o indicador de umidade (baixa, moderada e alta).

Quadro 13 – Corpo do index.html.twig

```
<script type="text/javascript">
    var ctx1 = document.getElementById("meuCanvas1");
    var ctx2 = document.getElementById("meuCanvas2");
    var ctx3 = document.getElementById("meuCanvas3");

    var globalOptions = {
            scales: {
                yAxes: [{
                    ticks: {
                        min: 1,
                        max: 1024,
                        stepSize: 350,
                        callback: function(value, index, values) {
                            if(value <= 350) {
                                return 'Baixa';
                            }

                            if(value <= 700) {
                                return 'Média';
                            }

                            if(value <= 1020) {
                                return 'Alta';
                            }

                            return '';
                        }
                    }
                }]
            }
        };

    var data1 = {
        labels: [
            {% for dados in dispositivo1 %}
            "{{ dados.dataLeitura|date('d/m H:i') }}",
            {% endfor %}
        ],
        datasets: [
            {
                label: "Dispositivo 1",
                borderWidth: 1,
                backgroundColor: '#0083db',
                data: [
                    {% for dados in dispositivo1 %}{{ dados.valorLido }},{% endfor %}
                ]
            }
        ]
    };

    {% if dispositivo1|length > 0 %}
    var myChart = new Chart(ctx1, {
        type: 'line',
        data: data1,
        options: globalOptions
    });
    {% endif %}
```

Fonte: elaborado pelo autor.

Em seguida na linha 96 é inicializado um bloco de javascript para desenhar os gráficos de cada dispositivo. É definida uma variável de opções padrões chamada `globalOptions`, e definidas algumas variáveis com os dados do gráfico, utilizando os comandos de *template*

para realizar *loops* e variável de *template* para realizar a impressão de valores para o HTML. Por fim na linha 146 é verificado se existem dados para o dispositivo com linguagem de *template* e construído o gráfico.

3.3.4 Operacionalidade da implementação

Para melhor distinção entre as partes do sistema, este capítulo será dividido em duas partes. Primeiramente serão apresentados detalhes de operação do hardware e em seguida informações sobre a aplicação web.

3.3.4.1 Hardware

O hardware está programado para dois tipos de conexão, aquela que está configurada na página de configuração do SMSL e uma configuração padrão. Quando o dispositivo é ligado ele procura no seu sistema de arquivos um arquivo denominado config.txt, onde estão registradas as configurações definidas no SMSL. Se este arquivo não for encontrado, significa que este dispositivo nunca se conectou a internet, então ele entra em modo de conexão padrão e busca infinitamente se conectar a uma rede denominada admin, utilizando a senha admin123. Quando esta é encontrada o dispositivo faz a leitura das configurações no SMSL, grava no config.txt e então passa a utilizar esta rede. Se o arquivo config.txt for encontrado, significa que este dispositivo já teve uma conexão com a internet e então tenta conectar na rede que está no arquivo de configuração. Após cem tentativas sem sucesso, o dispositivo entra em modo de conexão padrão e segue o processo explicado anteriormente.

O hardware já está encapsulado numa caixa de acrílico, basta posicioná-lo na plantação no local que se deseja avaliar a umidade do solo. O equipamento é resistente à água, então não há necessidade de removê-lo da plantação em períodos de chuva. A função do hardware é: processar uma leitura do solo, enviar para a aplicação e dormir o período configurado. No projeto foi utilizado uma bateria de 400mAh que oferece uma autonomia de operação de nove horas sem luz solar.

3.3.4.2 Aplicação Web

A aplicação web está dividida em duas páginas, sendo elas a página principal e a página de configuração. Na Figura 15 é possível observar a página inicial da aplicação.

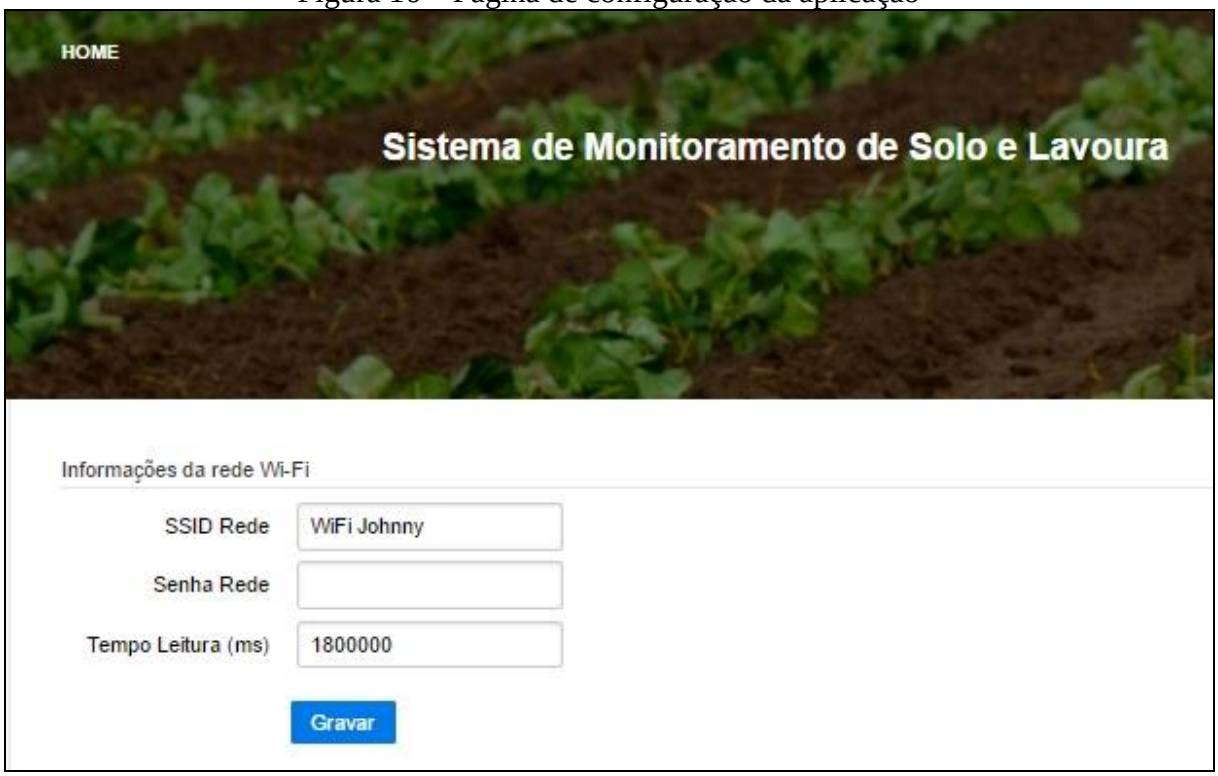
Figura 15 – Página inicial da aplicação

Fonte: elaborado pelo autor.

Na parte superior da Figura 15 está disponível um menu denominado Configuração que concede acesso a página de configuração do hardware que será comentado mais adiante. Além do menu, existem gráficos na página que exibem as leituras feitas pelo hardware. Estes gráficos exibem verticalmente uma escala que determina se as leituras estão apontando solo úmido, solo com umidade moderada ou solo seco e horizontalmente estão disponíveis a data e a hora que as leituras foram executadas pelo hardware. Após o gráfico, existe um elemento que nos indica qual o estado atual do solo. Se a última leitura executada pelo hardware atingir valores inferiores a 350 ele indica solo seco, valores entre 351 e 700 solos com umidade moderada e valores acima de 701, solo úmido.

Na Figura 16 está a página de configuração.

Figura 16 – Página de configuração da aplicação



Fonte: elaborado pelo autor.

Na parte superior da figura está disponível o menu que leva a página inicial da aplicação. Ao centro estão disponíveis alguns campos, onde o usuário informa qual rede/senha que deseja que o hardware utilize para enviar as informações do solo, além do tempo de intervalo entre cada leitura.

3.4 ANÁLISE DOS RESULTADOS

O desenvolvimento deste projeto permitiu o monitoramento da umidade do solo numa plantação através de uma página na internet. Para confirmar a veracidade dos dados coletados pelo sensor higrométrico foi executado um teste com um vaso com terra e um copo com água da seguinte forma: foi configurado o sensor para fazer leituras com intervalos de um minuto e colocado dentro do vaso com terra, onde a cada 1 minuto foi adicionado água. Conforme a água foi adicionada no vaso o valor exibido no gráfico do sensor foi crescendo, provando que a umidade aumentou. Na Figura 17 é possível visualizar o teste executado e na Tabela 2 os valores capturados pelo sensor.

Figura 17 – Teste do sensor higrométrico

Fonte: elaborado pelo autor.

Tabela 2 – Valores capturados pelo sensor

Hora	**Valor**	**Solo**
09:22	212	Seco
09:23	460	Moderado
09:24	680	Moderado
09:25	820	Úmido

Fonte: elaborado pelo autor.

A tabela acima mostra que conforme foi adicionada água no vaso o sensor passou a capturar valores maiores indicando que a terra ficou mais úmida, provando assim que o sensor é eficiente e pode ser utilizado para o fim proposto.

Inicialmente foi utilizado outro sensor para fazer a leitura da umidade do solo, mas durante a fase de testes o sensor não era capaz de perceber que o solo estava sendo irrigado, pois ele só faz leituras digitais, ou seja, só traz a informação se o solo está ou não úmido. Como este não é o objetivo do trabalho, o sensor foi substituído por um sensor que faz leituras analógicas, capaz de perceber uma a variação da umidade no solo, sensor este que atendeu a necessidade do projeto.

Durante a fase de implementação foram encontrados alguns problemas em relação à conexão Wi-Fi, pois esta poderia não estar disponível para o hardware, podendo ocasionar a perda da leitura do solo e também não ser possível determinar o horário correto da leitura. Para contornar o problema do relógio, foram criadas variáveis que armazenam a hora e a data e foi criada uma interrupção denominada `tCallback` que a cada um segundo é responsável por atualizar o relógio. Além disso, sempre que uma nova conexão com a Wi-Fi for realizada, o relógio é atualizado, inclusive a primeira conexão. Para contornar o problema de uma possível perda da leitura por falta de conexão Wi-Fi, foi utilizado o sistema de arquivos do ESP8266. As leituras do solo são registradas num arquivo denominado leituras.txt e em seguida é feito a tentativa de conexão com a Wi-Fi. Independente da conexão se efetivar ou não, a leitura esta a salvo no arquivo, evitando perda de dados.

Uma limitação da aplicação Web é o seu uso em Smartphones. A aplicação não foi totalmente otimizada para estes aparelhos, sendo assim a experiência do usuário pode ser insatisfatória dependendo do tamanho e resolução da tela. Porém este item não descarta seu uso em tais aparelhos como visto na operacionalidade da implementação.

Na Figura 18 é possível visualizar as leituras que um sensor efetuou a cada 30 minutos na plantação entre os dias 25/06/2017 e 26/05/2017.

Figura 18 – Leituras de um sensor

Fonte: elaborado pelo autor.

4 CONCLUSÕES

A tecnologia veio para facilitar a vida do homem e que se pode aplicá-la nos mais variados problemas, aperfeiçoando ou até mesmo substituindo nossas atividades diárias. Durante muito tempo a vida das pessoas no campo foi de trabalho pesado, muitas vezes não por opção, mas por falta de recursos ou por não estarem aptas a utilizarem ferramentas modernas. Este trabalho apresentou o desenvolvimento de um protótipo de sistema de monitoramento de solo e lavoura batizado de SMSL. Os principais objetivos eram desenvolver um hardware utilizando o microcontrolador ESP8266 Thing para capturar dados referentes à umidade do solo para transmitir para uma aplicação e desenvolver uma aplicação web para exibir ao agricultor os dados coletados pelo hardware para tomada de decisão. Conforme demonstrado, foi concebido um protótipo e uma aplicação que demonstram a viabilidade do projeto num ambiente com conexão Wi-Fi disponível.

As ferramentas utilizadas para a especificação e desenvolvimento atenderam bem as necessidades surgidas e em relação ao hardware e componentes utilizados, o microcontrolador ESP8266 Thing utilizado supriu todas as necessidades do projeto, como uma conexão Wi-Fi de qualidade e o sensor higrométrico obteve resultados satisfatórios e supriu todas as necessidades.

Conforme apresentado nos trabalhos correlatos, existem aplicações similares já em operação, mas que infelizmente são sistemas proprietários. O Sistema Irriga é operado pela Universidade Federal de Santa Maria/Rio Grande do Sul (UFSM/RS) e possui grande poder de crescimento pelo seu âmbito de aplicação social, além de se ajustar as necessidade hídricas das culturas com a demanda climática, pois leva em consideração a influência de todos os fatores do balanço hídrico, contribuindo diretamente para que a água seja utilizada racionalmente.

O trabalho desenvolvido é uma implementação que trabalha de forma similar ao Sistema Irriga, onde as informações são obtidas de um hardware específico. A sua grande vantagem em relação aos demais, é a possibilidade de adoção por qualquer agricultor do Brasil ou do mundo que possui conexão Wi-Fi em sua lavoura. Apesar de ser abordada apenas em plantação de morangos, a aplicação desenvolvida pode ser utilizada em outras plantações sem qualquer configuração adicional. Os resultados obtidos foram satisfatórios, sendo que o trabalho alcançou os objetivos propostos.

4.1 EXTENSÕES

No decorrer do desenvolvimento do trabalho, foram observados alguns aspectos que podem ser aprimorados.

A utilização da biblioteca Deep Sleep do Arduino. Em alguns dispositivos alimentados por bateria, ou por qualquer outra fonte de energia limitada, é imprescindível que haja o menor consumo possível. A economia de energia possibilita que este dispositivo dure por dias ou até mesmo anos sem a necessidade de nenhuma fonte adicional de energia. O dispositivo construído no projeto tem autonomia de operação de nove horas sem luz solar, tempo que não foi considerado adequado. Outra solução seria trocar a bateria por uma bateria com maior capacidade, pois neste projeto foi usando uma bateria de 400mAh que tem pouca capacidade.

O segundo aspecto a melhorar seria incluir a possibilidade de ativar os irrigadores automaticamente através de atuadores, pois hoje esta atividade é feita manualmente pelo agricultor ao visualizar um determinado ponto com pouca umidade.

O terceiro aspecto a melhorar seria um sistema de notificação para aplicativos Android e iOs. Este novo sistema iria permitir ao usuário final configurar um limite mínimo e máximo da umidade, e depois de feito isso, o sistema o alertaria automaticamente através de e-mail, SMS ou mesmo um aplicativo de celular quando a umidade fugisse do limite estabelecido.

Um quarto aspecto a melhorar seria implementar um algoritmo que calculasse o volume de água necessário de cada irrigação a ser aplicada na plantação, evitando assim ao máximo o desperdício de água.

Outra questão que pode ser trabalhada é a possibilidade da aplicação web exibir a previsão do tempo para os próximos dias, pois pode influenciar na decisão do agricultor de irrigar ou não a plantação.

REFERÊNCIAS

ANGELO, Claudio, MELLO, Mariana, VOMERO, Maria Fernanda. A era da falta d'água. Uma previsão catastrófica marca o colapso da água no mundo para o ano 2025. Foi dada a largada para a corrida em busca de soluções. Veja o que se pode fazer para não entrarmos pelo cano. **Superinteressante**, São Paulo, ano 14, n. 7, p. 48-54, jul. 2000.

ARDUINO E CIA. **Calculadora Online - Divisor de tensão com resistores**. 2017. Disponível em: <http://www.arduinoecia.com.br/p/calculador-divisor-de-tensao-function.html>. Acesso em: 09 jun. 2017.

ARDUINO. **Arduino**. 2017. Disponível em: <https://www.arduino.cc/en/Main/Software>. Acesso em: 18 jun. 2017.

ASSUNÇÃO, Arthur. **ESP8266 – Introdução E Primeiros Passos**, 2016. Disponível em: <http://www.decom.ufop.br/imobilis/esp8266-introducao-e-primeiros-passos/>. Acesso em: 20 mai. 2017.

BALBINO, Luiz Carlos et al. Agricultura sustentável por meio da integração lavoura-pecuária-floresta (*i*LPF). **International Plant Nutrition Institute – Brasil – Informações Agronômicas**, São Paulo, n. 138, p. 1-18, jun. 2012. Disponível em:<http://www.ipni.net/PUBLICATION/IA-BRASIL.NSF/0/67E9CCA96D48CF6685257A84004F5D7D/$FILE/IA-2012-138.pdf> Acesso em: 20 mai. 2017.

BERNARDO, S. Impacto ambiental da irrigação no Brasil. In: SILVA, D. D.; PRUSKI, F. F. (ed.). **Recursos hídricos e desenvolvimento sustentável da agricultura**. Viçosa: MMA, SRH, ABEAS, UFV, 1997.

CHART.JS. **Chart.js**. 2017. Disponível em: <http://www.chartjs.org>. Acesso em: 18 jun. 2017.

COMPOSER. **Composer**. 2017. Disponível em: <https://getcomposer.org/Composer-Setup.exe>. Acesso em: 18 jun. 2017.

DAVIS, Harold. **Absolute Beginner's Guide to Wi-Fi Wireless Networking**. Indianapolis: Que Publishing, 2004.

DINIZ, Eduardo H. Era digital. **GVexecutivo**, São Paulo, n. 5, p. 59, fev. 2006. Disponível em: <http://bibliotecadigital.fgv.br/ojs/index.php/gvexecutivo/issue/viewIssue/1888/723>. Acesso em: 21 mai. 2017.

FANCELLI, Antonio Luiz; DOURADO NETO, Durval. **Produção de milho**. 2. ed. Guaíba: Agropecuária, 2004.

FILIPEFLOP. **Placa FTDI FT232RL Conversor USB Serial**. 2017. Disponível em: <http://www.filipeflop.com/pd-14690c-placa-ftdi-ft232rl-conversor-usb-serial.html>. Acesso em: 01 jun. 2017.

GAST, Matthew. **802.11 Wireless Networks:** The Definitive Guide. 2nd. Sebastopol, CA: O'Reilly Media Inc., 2005.

JACOBI, Pedro. **A água na terra está se esgotando? É verdade que no futuro próximo teremos uma guerra pela água**. 2006. Disponível em: <http://www.geologo.com.br/aguahisteria.asp>. Acesso em: 21 mai. 2017.

LOPES, Maurício Antônio; CONTINI, Elísio. Agricultura, sustentabilidade e tecnologia. **Especial EMBRAPA – Agroanalysis**, São Paulo, fev. 2012. Disponível em: <http://bibliotecadigital.fgv.br/ojs/index.php/agroanalysis/article/viewFile/24791/23560>. Acesso em: 03 mai. 2017.

MICROCONTROLADORES. 2017. Disponível em: <http://robolivre.org/conteudo/microcontroladores>. Acesso em: 03 abr. 2017.

PASIEKA, Tiago José. **Aplicação web para monitoramento dos dados coletados por redes de sensores sem fio em ambiente agrícola**. 2014. 70 f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Sociedade Educacional Três de Maio, Rio Grande do Sul, 2014. Disponível em:<http://www.escavador.com/sobre/2262493/tiago-jose-pasieka> Acesso em: 03 mai. 2017.

PHP. **PHP: Hypertext Preprocessor**. 2017. Disponível em: <http://windows.php.net/download/>. Acesso em: 18 jun. 2017.

POZZEBOM, Rafaela. **O que são painéis solares?** 2013. Disponível em: <https://www.oficinadanet.com.br/artigo/ciencia/o-que-sao-os-paineis-solares>. Acesso em: 31 mai. 2017.

PROESI. **P23 - Módulo sensor de umidade de solo**. 2017. Disponível em: <http://proesi.com.br/catalog/product/view/id/8421/s/p23-modulo-sensor-de-umidade-de-solo/>. Acesso em: 06 jun. 2017.

PURE.CSS. **Pure.css**. 2017. Disponível em: <http://purecss.io>. Acesso em: 18 jun. 2017.

SEEEDSTUDIO. **0.5W Solar Panel 55x70**. 2017. Disponível em: <https://www.seeedstudio.com/05W-Solar-Panel-55x70-p-632.html>. Acesso em: 31 mai. 2017.

SEGALA, Mariana. Água: a escassez na abundância. Hoje, 40% da população do planeta já sofre as consequências da falta de água. Além do aumento da sede no mundo, a falta de recursos hídricos tem graves implicações econômicas e políticas para as nações. **Guia Exame Sustentabilidade**, dez. 2012. Disponível em: <http://planetasustentavel.abril.com.br/noticia/ambiente/populacao-falta-agua-recursos-hidricos-graves-problemas-economicos-politicos-723513.shtml>. Acesso em: 03 abr. 2017.

SISTEMA IRRIGA. **Irrigação**. 2017. Disponível em: <https://www.sistemairriga.com.br/index.php>. Acesso em: 10 abr. 2017.

SPARKFUN. **SparkFun ESP8266 Thing**. 2017a. Disponível em: <https://www.sparkfun.com/products/13231>. Acesso em: 30 mai. 2017.

TEIXEIRA, Fernando A.; PEREIRA, Fernando; VIEIRA, Gustavo. **Siot – Defendendo a Internet das Coisas contra Exploits**. Minas Gerais: Universidade Federal de Minas Gerais, 2014. Disponível em: <sbrc2014.ufsc.br/anais/files/trilha/ST14-1.pdf>. Acesso em: 08 abr. 2017.

TWIG. **Twig**. 2017. Disponível em: <https://twig.sensiolabs.org/>. Acesso em: 18 jun. 2017.

VICENZI, Alexandre. **Bustracker**: sistema de rastreamento para transporte coletivo. 2015. 61 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2015. Disponível em: <https://raw.githubusercontent.com/alexandrevicenzi/tcc/master/monografia/tcc_bcc_2015_2_avicenzi_AlexandreVicenzi-VF.pdf>. Acesso em: 20 abr. 2017.

WENDLING, Marcelo. **Sensores**. São Paulo: Universidade Estadual Paulista, 2010. Disponível em: <http://www2.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf>. Acesso em: 03 abr. 2017.

WOLF, Luís F. **Agricultura ecológica**: agricultura sustentável e sistemas ecológicos de cultivo (agricultura química x agricultura ecológica). 2017. Disponível em: <http://www.agrisustentavel.com/doc/tipos.htm>. Acesso em: 10 mai. 2017.

_____. **Voltage Dividers**. [S.l]. 2017b. Disponível em: <https://learn.sparkfun.com/tutorials/voltage-dividers>. Acesso em: 09 jun. 2017.

_____. **Using the Serial Monitor**. 2017c. Disponível em: <https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/using-the-arduino-addon>. Acesso em: 09 jun. 2017.

ANEXO A – Configuração do ambiente de desenvolvimento Web

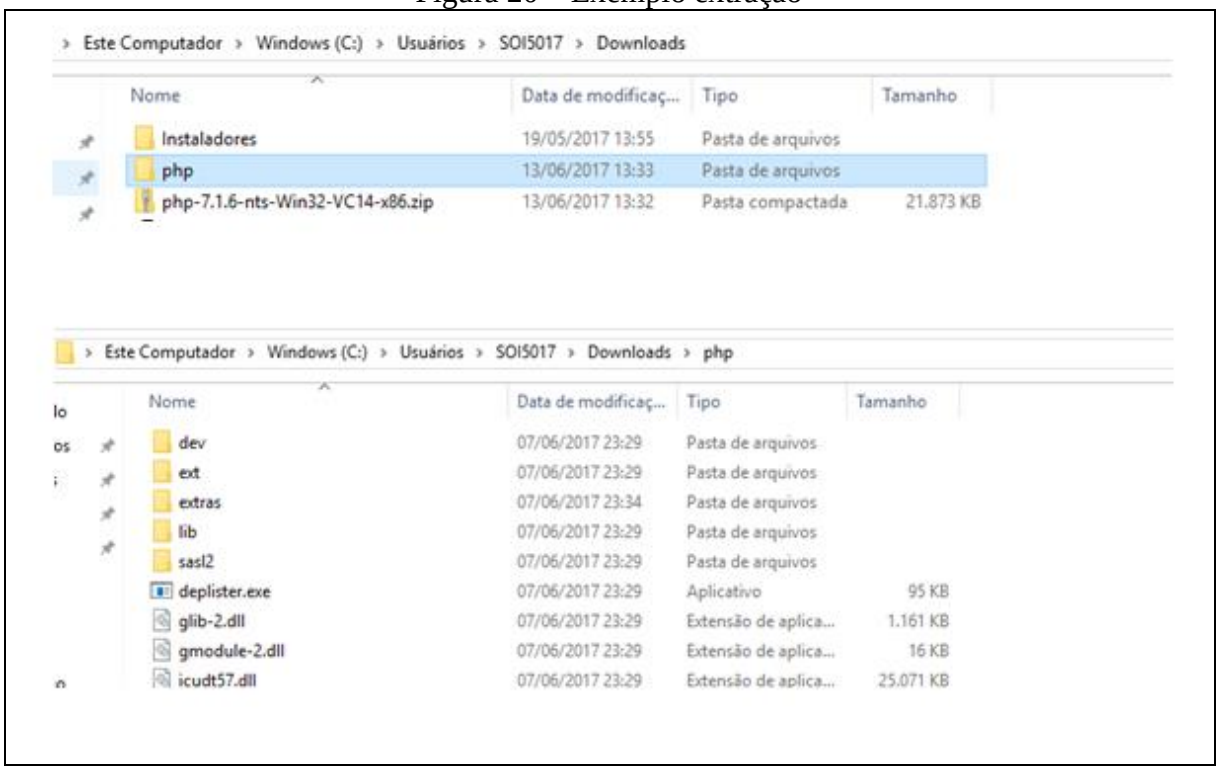
Para configurar o ambiente de desenvolvimento web é necessário fazer download e instalar a ultima versão do PHP (2017). A versão recomendada é: PHP 7.1 VC14 x86 Non Thread Safe no formato *zip*, como mostra a Figura 19.

Figura 19 – Versão recomendada PHP

Fonte: elaborador pelo autor.

Após realizar o download, extraia os arquivos em algum diretório conforme exemplo da Figura 20.

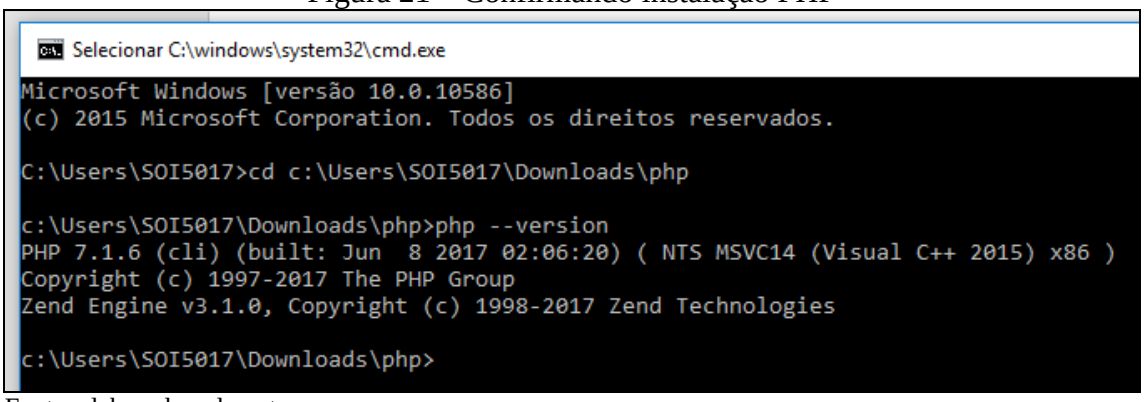
Figura 20 – Exemplo extração



Fonte: elaborado pelo autor.

Após extrair os arquivos é possível testar o resultado abrindo um prompt de comando na pasta e confirmar a versão do PHP com o comando `php -version`, como mostra a Figura 21.

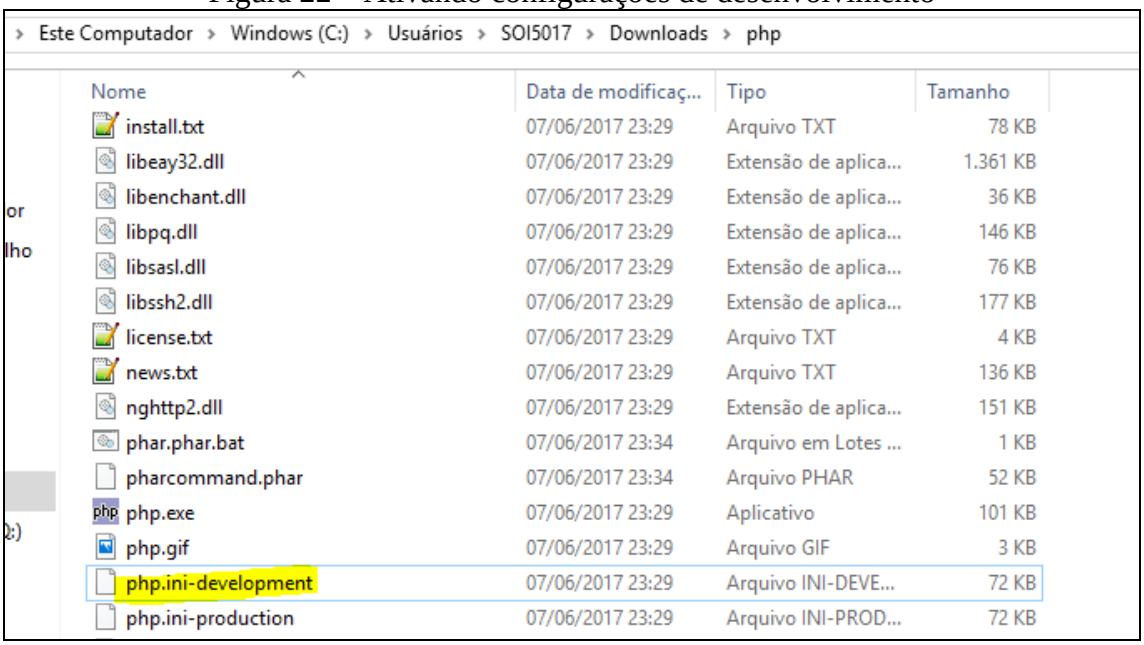
Figura 21 – Confirmando instalação PHP



Fonte: elaborado pelo autor.

Em seguida, ative as configurações de desenvolvimento do PHP, alterando o nome do arquivo `php.ini-development` para `php.ini`, como mostra a Figura 22.

Figura 22 – Ativando configurações de desenvolvimento



Fonte: elaborado pelo autor.

Abra o arquivo `php.ini` e localize a linha `extension_dir`, mudando ela para o caminho de instalação do seu PHP, como mostra o Quadro 14.

Quadro 14 – Ajuste no php.ini

```
719  ; http://php.net/include-path
720
721  ; The root of the PHP pages, used only if nonempty.
722  ; if PHP was not compiled with FORCE_REDIRECT, you SHOU
723  ; if you are running php as a CGI under any web server
724  ; see documentation for security issues.  The alternate
725  ; cgi.force_redirect configuration below
726  ; http://php.net/doc-root
727  doc_root =
728
729  ; The directory under which PHP opens the script using
730  ; if nonempty.
731  ; http://php.net/user-dir
732  user_dir =
733
734  ; Directory in which the loadable extensions (modules)
735  ; http://php.net/extension-dir
736  ; extension_dir = "./"
737  ; On windows:
738  extension_dir = "C:\Users\SOI5017\Downloads\php\ext"
```

Fonte: elaborado pelo autor.

Após isso, localize e habilite as extensões conforme Quadro 15. Para habilitar uma extensão, basta remover o caractere ; do início da linha.

Quadro 15 – Extensões que devem ser habilitadas

```
887  ; Note that ODBC support is built in, so no dll is nee
888  ; Note that many DLL files are located in the extensio
889  ; extension folders as well as the separate PECL DLL d
890  ; Be sure to appropriately set the extension_dir direc
891  ;
892  ;extension=php_bz2.dll
893  extension=php_curl.dll
894  extension=php_fileinfo.dll
895  ;extension=php_ftp.dll
896  ;extension=php_gd2.dll
897  ;extension=php_gettext.dll
898  ;extension=php_gmp.dll
899  extension=php_intl.dll
900  ;extension=php_imap.dll
901  ;extension=php_interbase.dll
902  ;extension=php_ldap.dll
903  extension=php_mbstring.dll
904  ;extension=php_exif.dll      ; Must be after mbstring
905  ;extension=php_mysqli.dll
906  ;extension=php_oci8_12c.dll  ; Use with Oracle Databas
907  extension=php_openssl.dll
908  ;extension=php_pdo_firebird.dll
909  extension=php_pdo_mysql.dll
910  ;extension=php_pdo_oci.dll
911  ;extension=php_pdo_odbc.dll
912  ;extension=php_pdo_pgsql.dll
913  ;extension=php_pdo_sqlite.dll
914  ;extension=php_pgsql.dll
915  ;extension=php_shmop.dll
```

Fonte: elaborado pelo autor.

Por fim, faça o download e instale o COMPOSER (2017). Para testar o ambiente, crie um arquivo chamado `info.php` em qualquer diretório, abra o arquivo e adicione as linhas conforme o Quadro 16 e salve o arquivo.

Quadro 16 – Arquivo de teste

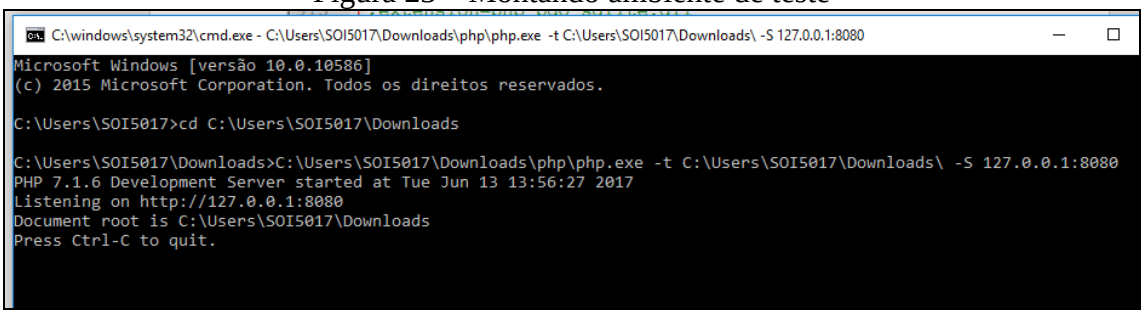
```
<?php

phpinfo();
```

Fonte: elaborador pelo autor.

Após isso, abra um prompt de comando e digite o conteúdo da Figura 23.

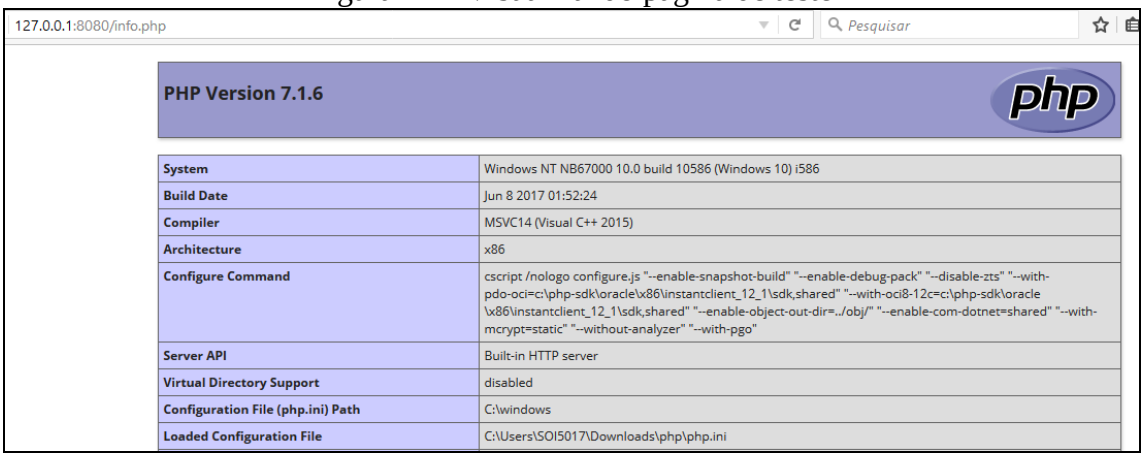
Figura 23 – Montando ambiente de teste



Fonte: elaborado pelo autor.

Por fim, abra o navegador de internet e visite o site como mostra a Figura 24.

Figura 24 – Visualizando pagina de teste



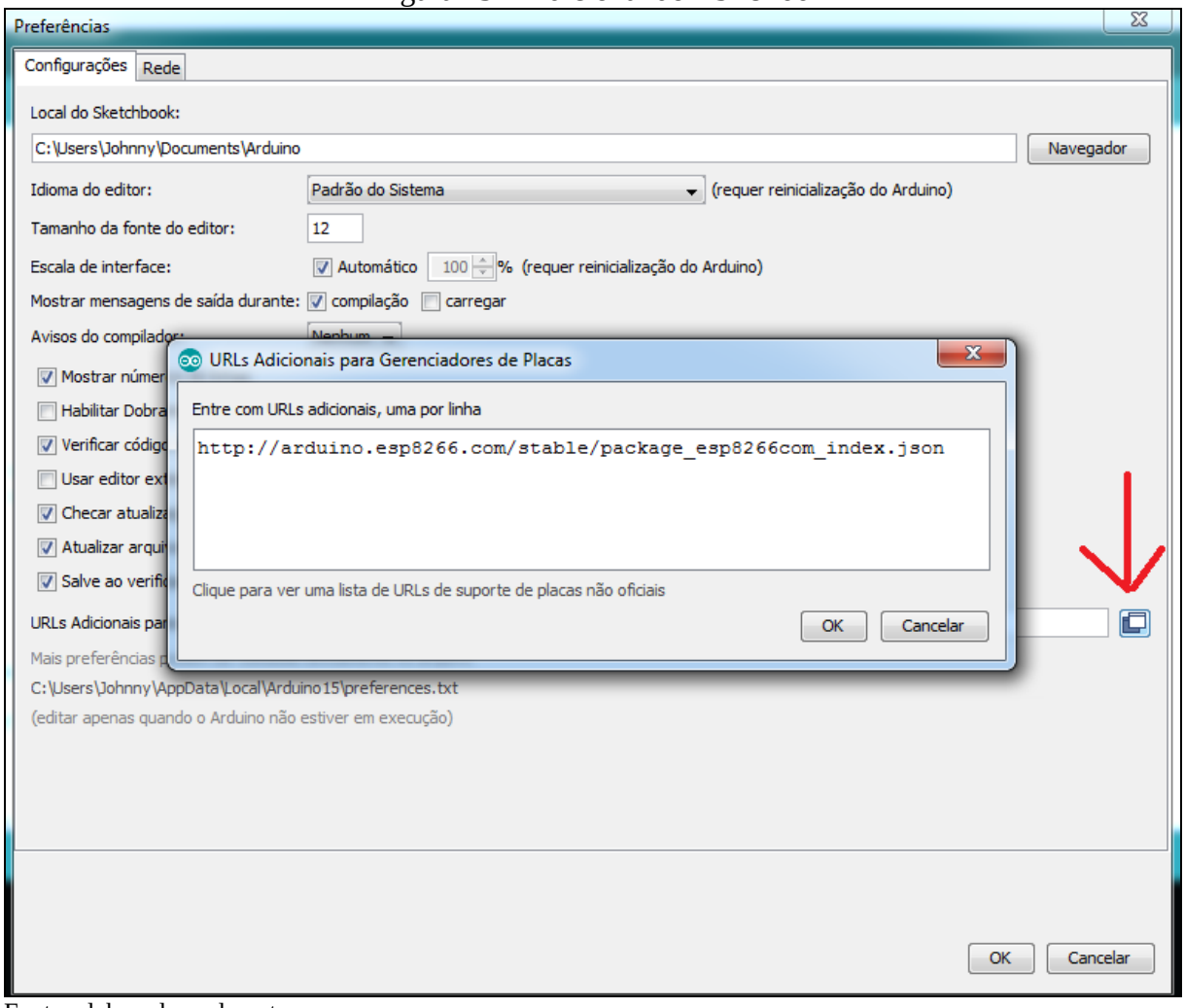
Fonte: elaborado pelo autor.

ANEXO B – Configuração do ambiente de desenvolvimento do Hardware

Para configurar o ambiente de desenvolvimento do microcontrolador ESP8266 Thing o fabricante da placa recomenda a utilização de um *addon* para a IDE do Arduino. Inicialmente faça o download da IDE do ARDUINO (2017) na versão mais recente.

Em seguida é necessário atualizar o gerenciador de placas com a URL do ESP8266 Thing. Abra a IDE do Arduino, vá em Arquivo>Preferências e na parte inferior da janela insira a URL na caixa de texto: URLs Adicionais para Gerenciadores de Placas, como mostra a Figura 25.

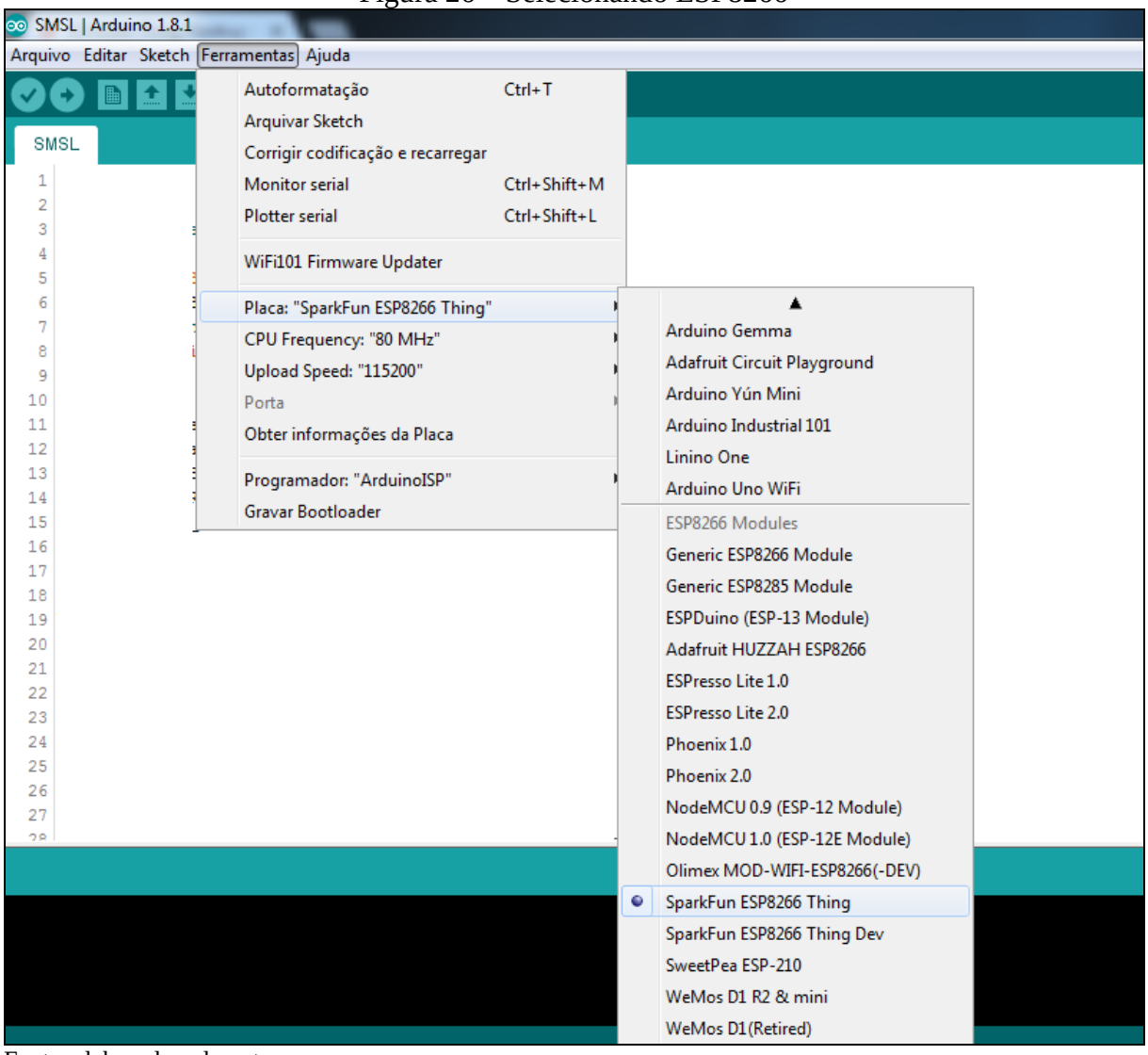
Figura 25 – Adicionando ESP8266



Fonte: elaborador pelo autor.

Clique em ok e em seguida navegue pelo menu até o gerenciador de placas: Ferramentas>Placa>Gerenciador de Placas. Procure pela placa ESP8266 Thing, selecione e clique em instalar. Com o addon do ESP8266 instalado, tudo que resta é selecionar a placa no menu Ferramentas>Placas, como mostra a Figura 26.

Figura 26 – Selecionando ESP8266



Fonte: elaborado pelo autor.