Antonio Augusto de Lima Baptista

Desenvolvimento de uma plataforma baseada em IoT para aquisição de dados aplicados à agronomia

São Luís

2018

Antonio Augusto de Lima Baptista

Desenvolvimento de uma plataforma baseada em IoT para aquisição de dados aplicados à agronomia

Monografia apresentada a Coordenação do Curso de Engenharia da Computação da Universidade Federal do Maranhão, como requisito para a obtenção do grau de Bacharel em Engenharia da Computação.

Universidade Federal do Maranhão – UFMA

Coordenação de Engenharia da Computação

Orientador: Prof. Dr. Rafael Fernandes Lopes

São Luís

2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Baptista, Antonio Augusto de Lima.
Desenvolvimento de uma plataforma baseada em IoT para aquisição de dados aplicados à agronomia / Antonio Augusto de Lima Baptista. – 2018.
71 f.

Orientador(a): Rafael Fernandes Lopes.
Monografia (Graduação) – Curso de Engenharia da Computação, Universidade Federal do Maranhão, São Luís, 2018.

1. Automação. 2. IoT. 3. LARAVEL. 4. REST. I. Lopes, Rafael Fernandes. II. Título.

Antonio Augusto de Lima Baptista

Desenvolvimento de um Sistema para Análise de Dados e Determinação do Fluxo de Seiva

Monografia apresentada a Coordenação do Curso de Engenharia da Computação da Universidade Federal do Maranhão, como requisito para a obtenção do grau de Bacharel em Engenharia da Computação.

Trabalho aprovado em 11 de julho de 2018:

Prof. Dr. Rafael Fernandes Lopes
Orientador
Coordenação de Engenharia da Computação

Prof. Dr. Shigeaki Leite de Lima
Professor Convidado
Coordenação de Engenharia da Computação

Prof. Dr. Luis Claudio de O. Silva
Professor Convidado
Coordenação de Engenharia da Computação

São Luís
2018

Este trabalho é dedicado primeiramente à Deus por sempre me dá o que preciso para melhorar e não o que quero, por sempre guiar meus passos por bons caminhos e me carregar nas horas de imensa tristeza e cansaço.

Aos meus pais Antonio Fernandes Baptista, Maria Elisabete de Lima e grande amigo Francisco Medeiros, "In Memorian", pelas suas existências em minha vida e com toda a contribuição ao longo de minha voda, mesmo com sacrifício próprio, pois sem eles este trabalho e muitos dos meus sonhos não se realizariam."

À toda minha família que abriu mão de tantos momentos comigo, em especial meu filho João Matheus, pois mesmo sem saber, sempre foi motivação principal de minha vida.

À namorada e companheira Letícia Ramos, pois foi por ela que dei meu primeiro passo dentro deste curso e por conta de todo seu apoio e dedicação o estou terminando.

Agradecimentos

Agradeço à todos os amigos de curso que sempre me incentivaram e contribuíram direta ou indiretamente para o desenvolvimento e conclusão deste trabalho.

À grande amiga Rayanne Silveira por toda ajuda prestada na produção da estação e programação Web deste sistema.

Ao Professor Dr. Shigeaki Leite de Lima por todos os ensinamentos, paciência e prestreza na produção do protótipo da estação e por sua sincera amizade ao longo desses anos de curso.

Ao Professor Dr. Rafael Fernandes Lopes, pela orientação e confiança, por grande desprendimento em sempre ajudar e pela amizade sincera.

Ao Instituto de Energia Elétrica da Universidade Federal do Maranhão por disponibilizar instalações e ferramentas para prototipação da estação SAFS.

Ao Curso de Pós Graduação de Agroecologia da Universidade Estadual do Maranhão pela confiança e oportunidade de demonstrar conhecimento e adquirir experiência valiosa para futuros desafios na carreira profissional.

Ao Professor Dr. Fabrício Oliveira Reis por sempre me receber nas dependências da Universidade Estadual do Maranhão com muita paciência e por todas as dúvidas sanadas sobre os requisitos e levantamento teórico da problemática deste trabalho.

Ao Professor Dr. Luís Cláudio De Oliveira Silva por ensinamentos valiosos sobre sistemas embarcados e por despertar a minha atenção pela arquitetura REST facilitando a nova abordagem do trabalho após mudança de requisitos por parte do cliente.

À Primeira-Tenente da Aeronáutica Ariadne Ruy, Adjunta-chefe da seção de redes operacionais do Centro de Lançamento de Alcântara por sempre confiar em meu julgamento e por acreditar nos meus estudos flexibilizando por muitas vezes os horários de trabalho e permitindo a permanência e continuação do curso até o fim.

Ao Primeiro-Tenente da Aeronáutica Mateus Mattos Moller, chefe da Seção de

redes Operacionais, por mesmo com pouco tempo de convívio, ter acreditado e continuado com a política da Tenente Ruy me permitindo concluir este trabalho e o curso.

Ao meu amigo Franklin Maria por sempre me incentivar e apoiar em meus estudos e por ser um exemplo que tento seguir desde de criança.

À todos os amigos do Centro de Lançamento de Alcântara que contribuíram de forma direta ou indireta para realização deste trabalho.

"*A educação é a arma mais poderosa que você pode usar para mudar o mundo.*"

- Nelson Mandela

Resumo

Nos tempos atuais existem inúmeras possibilidades de automação das mais variadas tarefas no cotidiano, rotinas simples às complexas vêm sendo automatizadas através de processos tecnológicos e integração com a grande rede, a Internet das Coisas se desenvolve em ritmo acelerado e mediante isto alavanca a busca por conhecimento nas áreas de eletrônica, elétrica, automação e Tecnologia da Informação. No presente trabalho está proposta uma solução para uma demanda do núcleo de Pós-Graduação em Agroecologia da Universidade Estadual do Maranhão, demanda a qual necessitou um levantamento bibliográfico sobre campos específicos de tecnologia e a posterior opção por montar uma solução de custo acessível unindo tecnologias já estabelecidas e em ascensão na comunidade. O protótipo proposto une *Hardware* e *Software* em uma estação de monitoramento de variação de temperatura com utilização de dois sensores e integração com um servidor web baseado em REST, se utilizando de uma arquitetura orientada à serviços e o *framework* Laravel baseado em MVC para desenvolvimento PHP. Nesse contexto, o trabalho se baseia na discussão teórica da problemática e implementação de funcionalidade de dispositivos já existentes no mercado, com diferenciais como custo e melhor acessibilidade à informação.

Palavras-chave: IoT, automação, REST, LARAVEL

Abstract

In the current times we are inundated with the possibilities of automation of the most varied tasks in the daily routine, simple to complex routines increasingly being automated through technological processes and integration with the great network, the Internet of Things (IoT) develops in a fast pace and along with this it leverages the search for knowledge in the areas of electronics, electrical, automation and IT. In the present work, a solution is proposed for a request from the core of the Postgraduate in Agroecology of the State University of Maranhão, which requires a bibliographical survey on specific fields of technology and the subsequent option to assemble an affordable solution linking technologies already established and rising in the community. The proposed prototype links Hardware and Software to a temperature-sensitive monitoring station using two sensors and integration with a REST-based web server using a service-oriented architecture and MVC-based Laravel framework for PHP development. In this context, the work is based on the theoretical discussion of the problematic and implementation of functionality of existing devices in the market, with differentials such as cost and better accessibility to information.

Keywords: IoT, Automated, REST, Laravel

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SAFS *Sistema de Análise de Fluxo de Seiva*

IoT *Internet Of Thingsr*

HTTP *Hypertext Transfer Protocol*

SOA *Service Oriented Architecture*

SOAP *Simple Object Access Protocol*

REST *Representational State Transfer*

JSON *JavaScript Object Notationr*

API *Application Programming Interface*

XML *Extensible Markup Language*

URL *Uniform Resource Locator*

URI *Uniform Resource Identifier*

PHP *PHP: Hypertext Preprocessor*

MVC *Model-View-Controller*

DBA *Database administrator*

GPIO *General Purpose Input/Output*

IEE *Instituto de energia elétrica*

IDE *Integrated Development Environment*

RF *Requisito Funcional*

RNF *Requisito Não Funcional*

Sumário

Capítulo 1

Introdução

Dentro do universo da *Internet of Things* (IoT), um tipo de paradigma lógico seguido por dispositivos que se interligam através da Internet, se tem várias tecnologias que culminam em uma considerável variedade de funcionalidades e posterior utilidades na vida cotidiana. o leque de funcionalidades é tão grande que o paradigma está se expandindo para outras áreas e importantes de ramos profissionais e acadêmicos.

Nos últimos anos o mercado de IoT no Brasil teve um crescimento considerável e o país desponta como um dos primeiros em crescimento do segmento na América Latina(ZABADAL; CASTRO, 2017) , a conectividade das "coisas", ou seja, qualquer dispositivo apto à se comunicar com outro através de algum protocolo de comunicação é o ponto chave que impulsiona este mercado. Dentro deste contexto há inúmeras aplicações nos mais diversos ramos para implementação do paradigma, entre as áreas: Indústria e linhas de produção, lojas, transporte e outros serviços públicos, logística, hospitais, agricultura e pecuária, automação residencial, etc...

Inovações tecnológicas estão sendo testadas e aos poucos implantadas na realidade das praticas agrícolas. O monitoramento das reais condições físico-climáticas no campo encontra atualmente uma vasta gama de sensores e atuadores, que provem as condições viáveis para a manutenção em tempo real das condições ideais de produtividade (SANTOS et al., 2016), e também aplicadas ao meio de pesquisa acadêmica, em especial devido a dificuldade de conexão à Internet ou outro tipo de transmissão de dados.

Nesta realidade e no contínuo aprimoramento das tecnologias e processos industriais e científicos, a heterogeneidade de sistemas é um fator relevante e deve ser um aspecto analisado visando tanto a implantação como manutenção e expansão de funcionalidades de um novo sistema. A Arquitetura Orientada à Serviços (*Service-Oriented Architecture - SOA*) propõe uma solução ao problema, se mostrando um paradigma que visa a organização de funcionalidades distintas e desintegradas, inclusive podendo ser produtos de

proprietários distintos. Basicamente o paradigma opera de forma a estabelecer um par solicitante/provedor de serviços e na comunicação entre ambos através de mensagens padronizadas.

Dentro do paradigma há uma implementação de destaque para o uso em IoT, a arquitetura *Representational State Transfer* (REST) se mostra adequada e eficiente para sistemas com recursos limitados no tocante a memória e processamento, se utilizando do protocolo *Hyper Text Transfer Protocol* (HTTP) e de mensagens do tipo *JavaScript Object Notation* (JSON) tornando possível uma comunicação mais leve, um requisito crucial para aplicações do nicho (NUNES et al.,).

1.1 Problema

No Brasil há um consumo de água na agropecuária cerca de três vezes maior que o consumo humano. Com a escassez deste recurso natural, este consumo consideravelmente alto se torna um fator de risco e que deve ser estudado a fim de obtenção de soluções tecnológicas para minimizar qualquer tipo de desperdício (SILVA, 2008).

Um dos principais aspectos a ser estudado é a irrigação da área do plantio, e para racionalizar tal técnica se faz necessário um estudo do gasto hídrico da planta para mensurar o grau de racionalização da água. Segundo REIS, Campostrini e SOUSA (2009):

> *As técnicas de irrigação, bem como o uso de espécies adaptadas às condições de deficiência hídrica,têm sido cada vez mais importantes para o sucesso da produção vegetal. De fato, a adequada irrigação em plantas cultivadas pode melhorar a eficiência no uso da água, refletindo na qualidade do produto agrícola e no custo de produção.*

Recentes trabalhos relatam que o fluxo de seiva é parâmetro indicativo do *status* hídrico da planta e ferramenta promissora para o manejo da irrigação. Alguns trabalhos foram utilizados e indicam o fluxo de seiva como indicativo de irrigação (NICOLAS et al., 2005).

Nesse contexto, se faz necessário o desenvolvimento de um sistema que busque facilitar a análise dos dados necessários para o estudo do fluxo de seiva e informações derivadas. Inferindo dessa forma no melhor uso das informações em pesquisas ou outros sistemas de automação que venham a utilizar os dados para outra aplicação prática da área.

Dentre os métodos existentes, este sistema optou pela aquisição da variação de temperatura como indicativo. Além de projetar e fabricar um protótipo para receber os dados advindos de sensores que podem ser desenvolvidos em trabalhos futuros ou adquiridos

separadamente mediante alguns fabricantes. O sistema proposto busca solucionar algumas limitações de aquisição dos dados pelo pesquisador, propondo então uma interface que possibilita a consulta das informações obtidas de forma organizada em um *notebook* ou *smartphone* com conexão com a Internet ou rede local.

1.2 Justificativa

No contexto previamente apresentado, a partir das necessidades impostas pelo grupo de pós-graduação da Universidade Estadual do Maranhão (UEMA), em que diversas pesquisas necessitam de um equipamento que disponibilize os dados desse tipo de medição (Fluxo de Seiva). Foi observado o alto custo do sistema disponível no mercado, uma vez que o equipamento pretendido para aquisição pela Universidade Federal do maranhão foi o *FLOW-32 1k* fabricado pela *Dinamax, USA*, custando cerca de R$ 5.430,00 mais o valor de frete e desembargo alfandegário. Além disso, se faz necessário a aquisição dos sensores tipo sonda que custam aproximadamente de R$ 330,00 cada, sem considerar os impostos de importação.

Com o principal intuito de fornecer uma solução que atenda os requisitos do cliente, em especial o custo da solução e experimentar a utilização de um protocolo de comunicação que vem sendo amplamente utilizado em IoT, foi concebido a ideia do Sistema para Análise do Fluxo de Seiva (SAFS). O sistema proposto se trata da integração entre uma estação *hardware* para aquisição de dados e envio a um sistema web através da Internet. O usuário poderá ter acesso aos dados em qualquer lugar através de *smartphone*, computador ou *laptop*, se utilizando de um sistema supervisor.

Outro aspecto a ser explorado é melhorar o acesso as informações através de uma aplicação própria e melhor elaborada que as já existentes embarcadas em dispositivos concorrentes, ou mesmo a ausência destas, deixando como única opção para o usuário o *download* dos dados via alguma interface serial ou USB.

A possibilidade de expansão de sensores por um sistema embarcado ou das próprias estações, a possibilidade de agregação de novos módulos e sensores de diferentes tipos, a facilidade de disponibilidade de informação e o baixo custo destes componentes devem resolver os principais problemas enfrentados pelo cliente, bem como a dificuldade na obtenção de dados para análise e o alto custo dos equipamentos semelhantes no mercado.

Embora existam outras formas de realizar a comunicação entre os dispositivos e o sistema embarcado, porém um dos objetivos deste trabalho é testar a abordagem utilizando a arquitetura REST para resolução do problema, mostrando como uma forma objetiva, simples e compacta de lidar com a comunicação inerente ao IoT.

Nesse aspecto, uma das justificativas desse projeto é a produção de um dispositivo

de baixo custo que pode colaborar para a disseminação do uso de tecnologias IoT na área da agronomia. Além de favorecer o incentivo a pesquisa acadêmica sobre protocolos de comunicação e disseminação do uso destas tecnologias para melhorar o acesso à dispositivos de alto custo para aquisição de dados. Além do baixo custo, é um dispositivo portátil que pode ser alimentado diretamente pela porta *USB* do computador e não necessita de estrutura cliente-servidor na mesma localidade da estação, ou seja, a mobilidade propiciada e necessária aos requisitos do cliente.

Outra justificativa para o estudo proposto é a escalabilidade do sistema. O dispositivo idealizado deverá ter a capacidade de incorporar novas funcionalidades com adição de novos módulos, por exemplo, sensores de umidade no solo e fornecimento da informação com uma mínima adição ao código embarcado e do sistema web. Portanto, as funcionalidades básicas dos dispositivos existentes no mercado poderão ser incorporadas e aprimoradas no sistema proposto. Com isso, é esperado contribuir para a divulgação dos métodos computacionais desenvolvidos, agregando tanto caráter tecnológico quanto científico ao sistema.

1.3 Objetivos

O projeto a ser desenvolvido busca propor uma solução tecnológica baseada em sistemas embarcados e sistemas web para o problema específico. Para o êxito do projeto especificado, É necessário concluir os seguintes objetivos gerais e específicos.

Objetivo geral

- Projetar o dispositivo de baixo custo que forneça os recursos básicos para fundir a aquisição e leitura dos dados através de uma interface intuitiva, unindo praticidade aos requisitos do cliente.

Objetivos Específicos:

- Pesquisar contexto e levantamento teórico da problemática e seus requisitos por meio de levantamento bibliográfico;
- Pesquisar sobre tecnologias IoT que possam ser agregadas ao projeto;
- Desenvolver e testar a comunicação entre a estação e o Servidor web.
- Desenvolver uma aplicação Web que receba, organize e facilite o tratamento dos dados obtidos através das leituras dos sensores conectados à(s) estação(ões).
- Desenvolver o protótipo do dispositivo;

1.4 Estrutura do trabalho

- Capítulo 2: Apresenta uma fundamentação teórica para melhor compreensão do contexto acerca da aplicabilidade do sistema, além de uma descrição superficial do mesmo. Também descreve e apresenta conceitos sobre a arquitetura escolhida e os componentes.
- Capítulo 3: Será abordado um breve estudo de mercado, descrevendo alguns dos dispositivos já existentes, focando nos recursos chaves solicitados pelo cliente: As leituras térmicas, interface amigável, mobilidade, escalabilidade e principalmente custo reduzido.
- Capítulo 4: Será apresentado o desenvolvimento da plataforma que possui um servidor web em comunicação com um dispositivo remoto que funciona como estação receptora dos sensores. Este sistema é responsável pela interface ao usuário final e gerenciamento do sistema como um todo. Neste capítulo serão descritos os requisitos e diagrama desse módulo levantados durante a fase de desenvolvimento.
- Capítulo 5: Serão enumerados e explanados os materiais e métodos empregados no desenvolvimento do protótipo do sistema, listando as funcionalidades de cada recurso tecnológico utilizado no processo, focando especialmente no estudo da arquitetura escolhida e do *hardware* utilizado.
- Capítulo 6: Serão listadas as funcionalidades do sistema como um todo e explanadas as formas de interação usuário-sistema.
- Capítulo 7: Serão demonstrados os resultados obtidos na experimentação parcial do sistema e avaliação do desempenho com uma discussão acerca dos resultados.
- Capítulo 8: Serão abordadas as conclusões acerca dos resultados obtidos e observações de todo o processo.

Capítulo 2

Fundamentação Teórica

2.1 Contexto da aplicação

Cerca de 98 % do volume de água absorvido pelas plantas é perdido por transpiração, todavia, esse fluxo de água é necessário para a manutenção da turgescência dos tecidos, regulação da temperatura da folha e para o transporte e absorção de nutrientes, sendo, portanto, necessário que a taxa seja mantida dentro dos limites ótimos para cada cultura (REICHARDT; TIMM, 2004).

Fatores inerentes ao clima, ao solo e as plantas interferem no processo de transpiração dos vegetais. A redução da umidade do ar aliada à elevação da temperatura contribui para o aumento da taxa transpiratória, bem como a forte irradiação, pois aumenta a diferença da pressão de vapor entre a folha e a atmosfera, na qual, a condutância estomática é reduzida em casos extremos, contribuindo para a queda da transpiração e consequentemente da fotossíntese. O mesmo ocorre durante o período de um dia, quando a transpiração decresce próximo ao meio dia, período em que as temperaturas do ar são mais altas e a umidade relativa do ar mais baixa. No entanto, durante a estação chuvosa, os valores de umidade do ar são mais elevados, não apresentando variações significativas ao longo do dia (LUTTGE, 1997; MORAES; PRADO, 1998; JÚNIOR et al., 2007).

Convencionalmente as irrigações são baseadas levando em consideração o teor de água no solo, no entanto, diversos métodos têm sido utilizados buscando aumentar a precisão das medições do consumo de água pelas plantas, em especial aqueles que permitem estimar diretamente a transpiração através da mensuração do fluxo de seiva (SILVA, 2008).

Relembrando, a determinação direta da transpiração vegetal pode ser mensurada por meio de técnicas termométricas (SILVA, 2008), ao analisar as mais variadas técnicas é possível notar que todas as técnicas seguem uma entre três grandes tendências: métodos de pulso de calor (MARSHALL, 1958), método de balanço de calor (SAKURATANI, 1981)

e o método de dissipação térmica (GRANIER, 1985).

Segundo SILVA (2008), o primeiro método trabalha no sentido de rastreio de um pulso de calor por um curto espaço de tempo através do fluxo de seiva, o segundo estuda o transporte do calor para fora da fonte aquecedora controlada e finalmente o terceiro trabalha na linha de dissipação de calor por uma relação empírica.

Seguindo o estudo do dispositivo previamente escolhido pelo cliente, da fabricante *Dinamax, USA*, um estudo superficial das três técnicas mostradas no parágrafo anterior e as próprias entrevistas com os professores e alunos do núcleo de pós-graduação de Agronomia da UEMA, o método de dissipação de calor proposto por Granier é mais indicado pela simplicidade do modelo matemático, custo de futura confecção do sensor resultando em menor complexidade de implementação e custo no sistema final.

2.2 Arquitetura Orientada a Serviços

Desde que surgiu, em um artigo publicado pela equipe de desenvolvimento de serviços web da IBM, a Arquitetura Orientada a Serviços (*Services Oriented Architecture - SOA*) tem sido alvo de grande interesse pela comunidade científica e recebeu grande aceitação na área de desenvolvimento de software (ATTORRE, 2015). A arquitetura trata basicamente de um tipo de implementação cliente-servidor com o panorama de comunicação baseado em mensagens de requisição e resposta entre os *Web Services* culminando no fornecimento de serviços ao solicitante, o intermédio entre o consumidor e o provedor do recurso, sendo realizado através de um terceiro elo: os registros de serviços. Figura 2.1.

Figura 2.1: Relacionamento entre papéis em SOA

Fonte: Adaptado de (GUEDES, 2013)

Na arquitetura SOA a abstração dos recursos computacionais são encarados como serviços a serem acessados e consumidos pelos usuários ou mesmo outros serviços, viabilizando a integração de diferentes componentes dentro de uma mesma organização ou mesmo em um aglomerado de organizações através da Internet.

Ainda na Figura 2.1, os papéis se relacionam basicamente através de três operações conforme exemplificado por Guedes (2013):

- **Publicação:** disponibilização de um serviço por um provedor. Nesta publicação são informados a localização e metadados do recurso, facilitando a busca do serviço. Desta forma, a identificação física do serviço, a chamada *Uniform Resoure Locator* (URL), em que o serviço pode ser encontrado e executado. As informações acerca dos metadados descrevem a funcionalidade do serviço de forma a permitir aos consumidores a análise recurso *versus* necessidade;
- **Busca:** o consumidor de serviços busca pelo recurso fornecendo parâmetros, através do acesso e análise dos metadados ele pode acessar a localização do recurso caso este atenda a necessidade. A busca pelo serviço pode ser projetada no desenvolvimento ou pela aplicação em tempo de execução;
- **Conexão:** para execução do serviço se faz necessário a conexão entre consumidor e provedor do serviço, as regras especificadas para que este procedimento ocorra são especificadas em uma abstração chamada "contrato de serviço", após atendidas o servidor executa a solicitação e retorna para o consumidor o resultado da operação, exemplificando a descrição *Request/Response* da arquitetura.

2.2.1 Web-Services

No universo de sistemas distribuídos e com o crescimento de disponibilização de serviços na Internet, se fez necessária uma padronização de comunicação inter-sistemas, de forma que a troca de informações entre sistemas locais de empresas pudessem se comunicar e utilizar informações advindas de sistemas exclusivos da web (voltados unicamente para navegadores), ou seja, voltados para *desktops*, *laptops* e *Smartphones*. Para viabilizar este cenário e esta comunicação necessária, se tornasse possível independente das especificações individuais de cada um como sistemas operacionais, linguagens de desenvolvimento utilizadas ou protocolos internos de aplicação, essa padronização se fez presente com a criação e aperfeiçoamento das *Application Programming Interface* (API) e do protocolo HTTP, utilizado na transmissão de dados, sendo excelente para a posterior disseminação da arquitetura SOA e REST. Figura 2.2.

Figura 2.2: Funcionalidade dos *Web Services*

Fonte: Adaptado de (CODENUCLEAR, 2018)

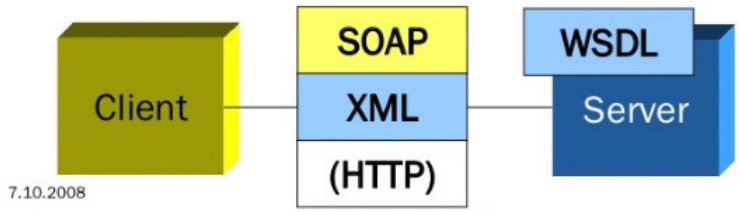
Entre outras mais antigas temos como opções de *Web Services* o protocolo *Simple Object Data Protocol* (SOAP) e o *RESTful*, que serão explicados nas próximas seções além de uma comparação entre ambos para utilização no sistema proposto.

2.2.2 Arquitetura SOAP

O protocolo SOAP trabalha com quatro principais funcionalidades: o formato das convenções para encapsulamento de dados e as orientações de rota na forma de um envelope, um transporte ou protocolo obrigatoriamente, regras de codificação, e um mecanismo de *Remote Procedure Call* (RPC). O envelope cuida das convenções para descrição do conteúdo da mensagem e isso implica diretamente na forma como os dados são processados. Um protocolo de ligação proporciona um mecanismo genérico para o envio de um envelope SOAP por meio de um protocolo de baixo nível, tais como HTTP. Regras de codificação proporcionam uma convenção para mapear vários tipos de dados da aplicação, para uma representação baseada em *tag Extensible Markup Language* (XML). Finalmente, o mecanismo de Chamada Procedimento Remoto, que fornece uma forma para representar chamadas de procedimento remoto e os valores de retorno (FERREIRA[1]; MOTA[1], 2014).

Basicamente, em uma comunicação nesta arquitetura, a aplicação solicita um serviço ao provedor através de uma requisição HTTP pela rede, o provedor devolve em uma resposta HTTP um arquivo *Web Services Description Language* (WSDL), ou seja, um arquivo em formato XML que trata de informar ao solicitante as informações acerca da utilização do serviço invocado, sejam nomes de parâmetros e métodos, formatação de arquivo de entrada e saída e endereço do serviço. Figura 2.3.

Figura 2.3: Protocolo *SOAP*



Fonte: Adaptado de (PAUTASSO, 2008)

Importante citar que o SOAP suporta *WS-ReliableMessaging* que tem uma lógica sucesso/repetição embutido enquanto o REST usa novas tentativas no caso de falhas. O SOAP se sobressai na criação de aplicações com segurança mais exigente, como transações bancárias (FERREIRA, 2015).

2.2.3 Arquitetura REST e implementação RESTful

A denominação como protocolo não se mostra a mais adequada, segundo Fielding e Taylor (2000), o REST pode ser definido como um estilo arquitetural de implementação, uma especificação cliente-servidor e tem como denominação de implementação REST em *Web Services* como *RESTful*. O padrão arquitetural REST abstrai detalhes de implementação dos componentes e a sintaxe de protocolos visando a evidência dos componentes e os respectivos papéis, as restrições, interações e interpretação da dados significantes (FIELDING; TAYLOR, 2000).

O REST foi desenvolvido a partir da sugestão de Roy Fielding, um dos principais criadores do protocolo HTTP durante o desenvolvimento da versão 1.1 do protocolo, afim de resolver problemas de semântica nas requisições HTTP. Entre as sugestões, uma forma de utilização dos verbos HTTP de forma mais próxima à realidade foi implementada, a partir do uso sugerido dos verbos *GET*, *POST*, *DELETE*, *HEAD*, *PUT* e *OPTIONS* permitindo reproduzir qualquer tipo de interação entre serviços em uma comunicação cliente-servidor.

Através do uso desses métodos e da arquitetura baseada em serviços, se torna bem mais simples expressar as ações desejadas, utilizando junto às URLs podendo ter expressões (fictícias) como:

- GET http://www.universidade.com.br/alunos
- DELETE http://www.universidade.com.br/alunos/fernandocctg
- POST http://www.universidade.com.br/alunos –data nome: lucasccg

É possível inferir apenas pela compreensão dos verbos e contexto das URLs que trata de uma operação de busca da lista de todos os alunos da universidade através do verbo GET, a deleção de um aluno chamado "fernandocctg"com o verbo DELETE e a

postagem de informações para um possível cadastro de novo aluno com o verbo POST.

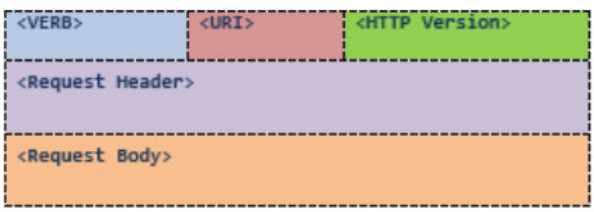
O REST se utiliza da troca de mensagens em formato XML ou JSON, segundo Ferreira[1] e Mota[1] (2014) o JSON é um formato mais leve, um objeto representado por uma concatenação de *Strings* no formato {texto: valor, texto: valor ... texto: valor}, possibilitando simplificar mesmo uma informação complexa. O JSON pode representar primitivas como *strings*, números, booleanos e *null*, além de estruturas de dados como *objects* e *arrays*.

Como sugerida durante atualização do HTTP, a arquitetura REST se utiliza de padrões já adotados pelo protocolo de transporte, economizando assim, o uso de recursos próprios. A mensagem REST é composta de Verbo, *Uniform Resource Identifier* (URI), HTTP *version*, *request header* e *request body*. Os verbos indicam as ações tomadas, como exemplificado nos exemplo dos alunos, logo o verbos existentes no REST são:

- POST - Envia dados para processamento do recurso identificado;
- GET - Solicita dados específicos de um determinado recurso;
- PUT - Funciona como edição de um recurso identificado;
- DELETE - Apaga o recurso especificado;
- OPTION - Solicita os métodos
- HEAD - Solicita apenas o cabeçalho do recurso identificado.

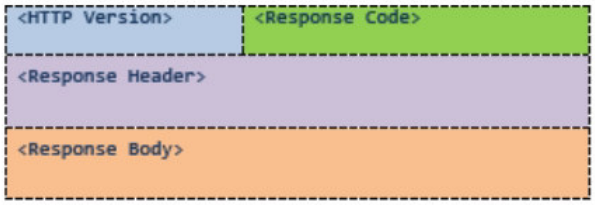
Novamente retomando o exemplo dos alunos, após o uso do verbo temos a URI indicando o local do recurso a ser manipulado, o endereço físico a ser buscado pelo servidor, seguidos da versão do protocolo de transporte (HTTP *version*), meta-dados e conteúdo da mensagem de requisição. Em uma mensagem de resposta REST os campos são o *Response code*, contendo o estado do pedido, *Response header*, os meta-dados sobre a mensagem de resposta e o conteúdo da mensagem propriamente dito como *Response body*. O formato da mensagem REST de requisição e resposta são visualizados nas Figuras 2.4 e 2.5 abaixo:

Figura 2.4: Formato de mensagem de requisição REST



Fonte: Adaptado de (VAQQAS, 2014)

Figura 2.5: Formato de mensagem de resposta REST



Fonte: Adaptado de (VAQQAS, 2014)

O REST vem sendo largamente utilizado, com registros de suas implementações (RESTful) em grandes empresas como Amazon, Google e Yahoo entre outras. O motivo disto pode ser explicado pela preferência de simplificação de comunicação devido ao menor volume de dados e a escalabilidade de serviços (MENDES, 2014).

Importante salientar que o REST não possui nenhum estado do cliente, toda mensagem é auto-suficiente, possuindo contexto para seu processamento, qualquer estado é mantido do lado do cliente, mesmo *tokens* de segurança são enviados em cada mensagem de *Request/Response*. Sendo possível considerar a ausência de estados em termos de endereçamento. Este contexto diz que cada informação interessante deve ser exposta na forma de um recurso e ter URI própria, no caso da ausência de estados, cada estado possível deverá ser expressado na forma de um recurso e ter própria URI (RICHARDSON; RUBY, 2008).

2.2.4 Web Services - Padrões de comunicação

Diante do exposto, embora de forma superficial, sobre os padrões apresentados, é possível inferir a preferência pela implementação REST baseado nas vantagens e desvantagens de cada paradigma considerando o contexto da aplicação desenvolvida neste trabalho.

O SOAP possui grande quantidade textual formatado em XML, suporte a mais recursos de segurança com o uso de WS-*Security*, maior compatibilidade nativa com linguagens de programação, sendo principalmente indicada em aplicações onde aquele contexto seja gerenciado com segurança (*Stateful*).

O REST é focado na orientação a recursos com o uso de Verbos e URIs, não necessita de suporta nativo da linguagem, pois utiliza de mensagens XML ou JSON mais simples, e possui característica *Stateless*, ou seja, mensagens com contexto auto-suficiente.

Considerando o levantamento teórico e principalmente as comparações realizadas por (NUNES et al.,), o REST se mostra mais adequado a implementação do projeto deste trabalho principalmente pelos dispositivos envolvidos, de certa forma limitados no aspecto de processamento e da aplicação possuir indicação de uso em áreas com largura de banda limitada. Os recursos de segurança adicionais e maior complexidade de conexão não são

tão relevantes e acabam por ser um empecilho para utilização do SOAP no contexto do projeto.

2.3 MVC

Segundo Franco, Piedade e Rêgo (2014), o *model-view-controller* (MVC) é definido como uma arquitetura de projeto. A ideia é subdividir as responsabilidades da aplicação em camadas, aumentando a modularização do *software* com a seguinte divisão:

- *Model* - cuida do modelo, da persistência e acesso dos dados.
- *View* - são interfaces gráficas, as telas da aplicação em que ocorre toda interação com o usuário, e através dos quais é realizado a manipulação e visualização de dados.
- *Controller* - gerencia as requisições de entrada do sistema, além de comandar as *Views* a serem visualizadas e as alterações na camada de modelos.

A maior vantagem do uso deste modelo é a sua modularização, com a aplicação de profissionais independentes nas camadas, no que um DBA poderia cuidar da camada de modelos, enquanto um programador poderia cuidaria de *scripts* CSS e HTML nas *Views* e outro profissional na modelagem de regras do negócio no *Controller*.

2.3.1 LARAVEL

O LARAVEL é um *framework* livre e de código aberto, utilizado para desenvolvimento rápido com PHP, sendo conhecido especialmente por prezar pela organização e legibilidade do código. Atualmente possui a liderança na preferência da comunidade PHP e vem sendo muito utilizado em outros grupos. Para ilustração é possível observar um comparativo em uma pesquisa no *Google Trends* com os principais concorrentes. Figura 2.6.

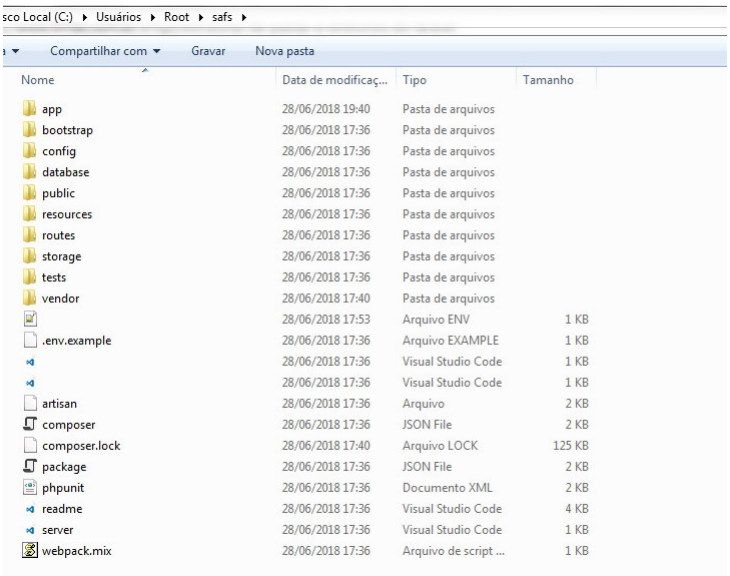
Figura 2.6: Popularidade atual do LARAVEL

Fonte: Adaptado pelo autor

Segundo a análise de (FRANCO; PIEDADE; RÊGO, 2014), o *framework* LARAVEL é recomendado para projetos de pequenos e médio porte pois possui uma aceitável complexidade de instalação, uma documentação estabelecida, abrangente e didática e uma comunidade crescente para suporte.

Ao utilizar um *framework* baseado em MVC, o projeto agrega a característica de subdivisão em diretórios de forma a separar os componentes do sistema das classes básicas do MVC (FRANCO; PIEDADE; RÊGO, 2014). Figura 2.7.

Figura 2.7: Estrutura de Diretórios do Laravel 5



Fonte: Produzido pelo autor

Ainda na na Figura 2.7, é ilustrado a estrutura inicial de diretórios do LARAVEL 5, cada diretório pode ser descrito no conteúdo como:

- APP - neste diretório são armazenados os códigos de controle do sistema, as *control-*

lers, *Models* e rotas.

- *BOOTSTRAP* - neste diretório são armazenados os arquivos de inicialização do *framework*, no caso o LARAVEL.
- CONFIG - são armazenados os arquivos de configuração do sistema.
- *DATABASE* - neste diretório são armazenadas as *migrations*, ou seja, estruturas das tabelas do banco de dados do sistema, além de funcionarem como um tipo de *backup* destas estruturas.
- *PUBLIC* - é o diretório público do projeto, em que ficarão todos os arquivos públicos como *htaccess*, folhas de estilo, *script*, e imagens.
- *RESOURCES* - neste diretório são armazenados arquivos importantes para as *views* do sistema, existem subpastas que armazenam arquivos de idioma e a própria pasta *Views*.
- *ROUTES* - neste diretório existem todas as definições de rotas do projeto, sejam dos comandos em *prompt* pelo *Artisan*, console ou mesmo do HTTP.
- *STORAGE* - diretório responsável por armazenar arquivos de *caches* e *logs* do sistema.
- *TESTS* - diretório que armazena os testes de integração do sistema.

Além dos diretórios já enumerados, a pasta-raiz do projeto possui alguns arquivos importantes, a seguir é realizado uma descrição sobre os que mais se destacam a fim de tornar mais clara a arquitetura de trabalho do *framework* LARAVEL.

- .ENV - arquivo em que todas as variáveis de ambiente são definidas, ou seja, permitem a configuração personalizada de cada ambiente, de forma a proteger dados importantes de cada integrante de uma equipe de desenvolvimento por exemplo.
- *artisan* - Arquivo que contém as configurações para execução de comando *Artisan* no *Prompt*.
- composer.JSON e composer.lock - arquivos de conFiguração do *Composer*, definem as dependências PHP e informações básicas do projeto, a diferença entre os dois é que o arquivo ".lock"não pode ser editado.

2.4 WAMP

O *WampServer* pode ser definido com um pacote de aplicações para instalação de um completo ambiente de desenvolvimento web em plataforma *Windows*, de forma intuitiva são instalados entre outras coisas um servidor APACHE, MySQL, PHP e a ferramenta PHPmyadmin, itens necessários para começar o desenvolvimento de qualquer aplicação web. A utilização do WAMP permite economizar tempo com instalação e configuração de cada componente, tornando o processo intuitivo, rápido e eficiente.

Capítulo 3

Soluções existentes

A funcionalidade do projeto proposto (SAFS) possui alguns equivalentes no mercado, embora existam algumas diferenças maiores ou menores, em sua maioria se baseiam na aquisição de dados via um sensor por dissipação de calor proposto por Granier (GRANIER, 1987) e recebimento dos dados por um *datalogger*. A diferença desta proposta trata de como os dados são tratados, armazenados e extraídos pelo usuário, além de características como invólucro e a alimentação.

É necessário informar, que alguns sensores implementam um segundo método de aferição, diferente do Granier que propõe um aquecimento constante na resistência. Este método alternativo consiste num pulso intercalado de calor em um dos sensores e uma medição de temperatura em períodos com maior intervalo (COHEN; FUCHS; GREEN, 1981). Relembrando, o principal diferencial proposto pelo SAFS é o custo otimizado aliado à facilidade de acesso aos dados e escalabilidade do sistema. Porém, se faz necessário um estudo dos equipamentos já disponíveis no mercado, assim como algumas das tecnologias utilizadas que poderão ser implementadas futuramente no SAFS.

3.1 Dispositivos Existentes

3.1.1 ES-SYS-2

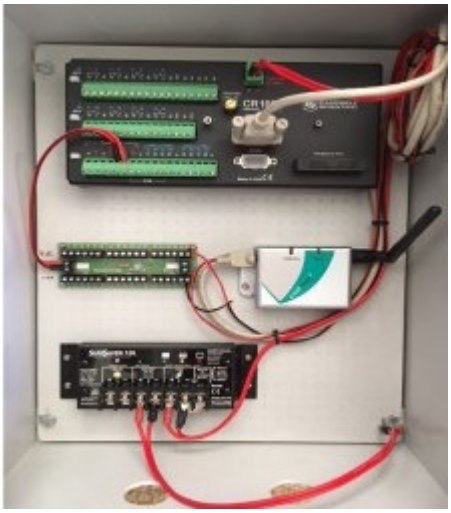
A empresa *Edaphic Scientific*, especializada em pesquisa ambiente e dispositivos de monitoramento para uso científico possui o próprio sistema de *SAP FLOW*, o ES-SYS-2, conforme ilustrado na Figura 3.1. Esse sistema trabalha acoplando um *datalogger* energizado por uma bateria recarregada por um painel solar, além de contar com um módulo GSM para transmissão de dados e pode receber vários tipos de sensores.

Figura 3.1: Sistema *ES-SYS-2*

Fonte: Adaptado de (SCIENTIFIC, 2018)

No caso específico de um sistema *SAP FLOW*, a *Edaphic Scientific* possui sensores que podem ser acoplados ao sistema maior ou sondas com *dataloggers* dedicados, com módulos de comunicação e alimentação de menor porte. Os sensores para este fim se baseiam no método de pulso intercalado de calor proposto por (COHEN; FUCHS; GREEN, 1981) , argumentando economia energética e consequentemente nos recursos destinados à alimentação do sistema. Os sensores podem vir com três ou duas agulhas, sendo uma a sonda aquecedora e a(s) outra(s) os termopares conforme ilustrado. Figuras 3.2 e 3.3.

Figura 3.2: Estrutura *ES-SYS-2*



Fonte: Adaptado de (SCIENTIFIC, 2018)

O sistema da *Edaphic Scientific* é composto por um *datalogger* de maior ou menor porte, como o *CR300* da *Campbell Scientific* acoplado com o subsistema de alimentação que compreende uma bateria 12 *Volts*, o módulo de comunicação GSM e a placa solar de recarga, além dos sensores *SAP FLOW*.

Figura 3.3: Sensor *SF3* em experimento (vendido separadamente)

Fonte: Adaptado de (SCIENTIFIC, 2018)

3.1.2 SFM1 Sap Flow System

Desenvolvido pela *ICT International*, é ilustrado na Figura 3.4. O *SFM1* se baseia na arquitetura na portabilidade e agregação de módulos juntos ao sensor, reunindo de forma integrada o *datalogger* portátil com software integrado, módulo *wireless* de coleta de dados 2.4 Ghz, Bateria interna de Lithium de 960mAH e os sensores propriamente ditos. Acompanham o conjunto stand-alone o módulo de comunicação *wireless* (GSM/3g/4g) e um painel solar de 11 ou 20 watts, à escolha do cliente.

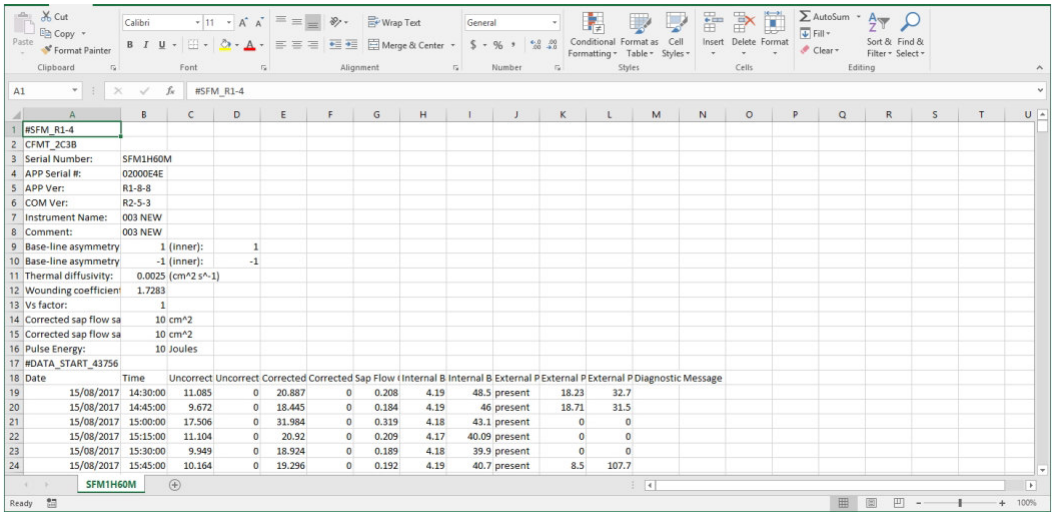
O sistema possui grande vantagem em portabilidade e usabilidade, porém, na falta de alimentação externa o sistema se mantém em atividade por um período de aproximadamente 36 horas, utilizando o método de pulso de calor intercalado (BURGESS, 2017).

Figura 3.4: Dispositivo *SFM1 Sap Flow System*

Fonte: Adaptado de (INTERNATIONAL, 2018)

Como o sistema da Edaphic Scientific, o SFM1 disponibiliza os dados em formatos *.csv* e *.bin*, que podem ser interpretados pelo SFT *Sap Flow Tool Software* (Figura 3.5), propiciando o acesso e a análise pelos usuários. Importante mencionar que o sistema SFT aceita sensores de outros fabricantes, porém o conjunto com os sensores da própria empresa possibilitam cálculos adicionais sobre os dados.

Figura 3.5: Relatório *SFT Sap Flow Software*



	A	B	C	D	E	F	G	H	I	J	K	L
1	#SFM_R1-4											
2	CFMT_2C3B											
3	Serial Number:	SFM1H60M										
4	APP Serial #:	02000E4E										
5	APP Ver:	R1-8-8										
6	COM Ver:	R2-5-3										
7	Instrument Name:	003 NEW										
8	Comment:	003 NEW										
9	Base-line asymmetry	1	(inner):	1								
10	Base-line asymmetry	-1	(inner):	-1								
11	Thermal diffusivity:	0.0025	(cm^2 s^-1)									
12	Wounding coefficien	1.7283										
13	Vs factor:	1										
14	Corrected sap flow sa	10	cm^2									
15	Corrected sap flow sa	10	cm^2									
16	Pulse Energy:	10	Joules									
17	#DATA_START_43756											
18	Date	Time	Uncorrect	Uncorrect	Corrected	Corrected	Sap Flow (	Internal B	Internal B	External P	External P	External P
19	15/08/2017	14:30:00	11.085	0	20.887	0	0.208	4.19	48.5	present	18.23	32.7
20	15/08/2017	14:45:00	9.672	0	18.445	0	0.184	4.19	46	present	18.71	31.5
21	15/08/2017	15:00:00	17.506	0	31.984	0	0.319	4.18	43.1	present	0	0
22	15/08/2017	15:15:00	11.104	0	20.92	0	0.209	4.17	40.09	present	0	0
23	15/08/2017	15:30:00	9.949	0	18.924	0	0.189	4.18	39.9	present	0	0
24	15/08/2017	15:45:00	10.164	0	19.296	0	0.192	4.19	40.7	present	8.5	107.7

Fonte: Adaptado de (BURGESS, 2017).

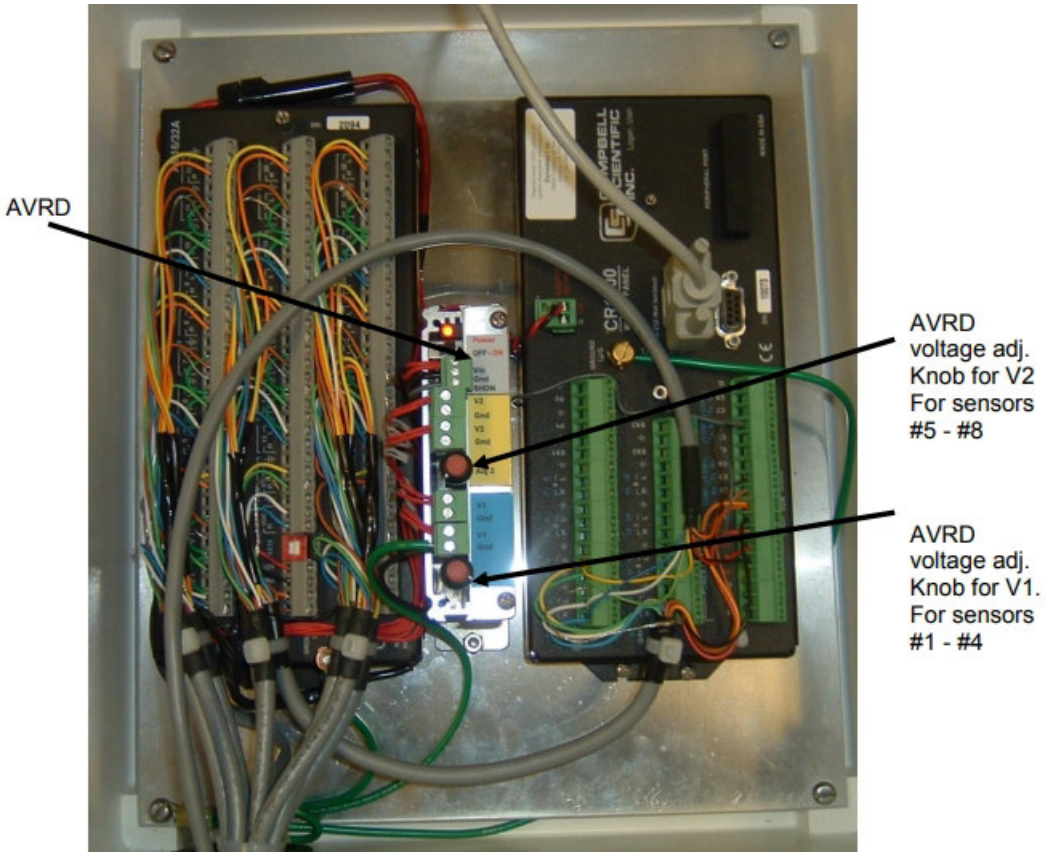
3.1.3 Dynagage Flow32-1k Sap Flow System

O Sistema fabricado pela Dynamax possui mais de dez anos de mercado e foi o cogitado para aquisição pelo núcleo de pós-graduação da Universidade Estadual do Maranhão.

O FLow32-1k possui características um pouco diferentes, pois possui arquitetura segmentada, na qual a estação principal é composta de um *datalogger* robusto com *software* de cálculo embarcado, modelo CR1000 da *Campbell Scientific*, um regulador de voltagem ajustável e um armazenamento de memória de 4 MB. O módulo de alimentação do sistema básico é composto por uma bateria 7 Ah de 12 *Volts* e módulo carregador 110/220 *Volts*, com adição opcional de painel solar com regulador. A conexão com o computador do usuário é realizada através de conexão serial no padrão RS232 e existe a possibilidade de download dos dados através do padrão USB.

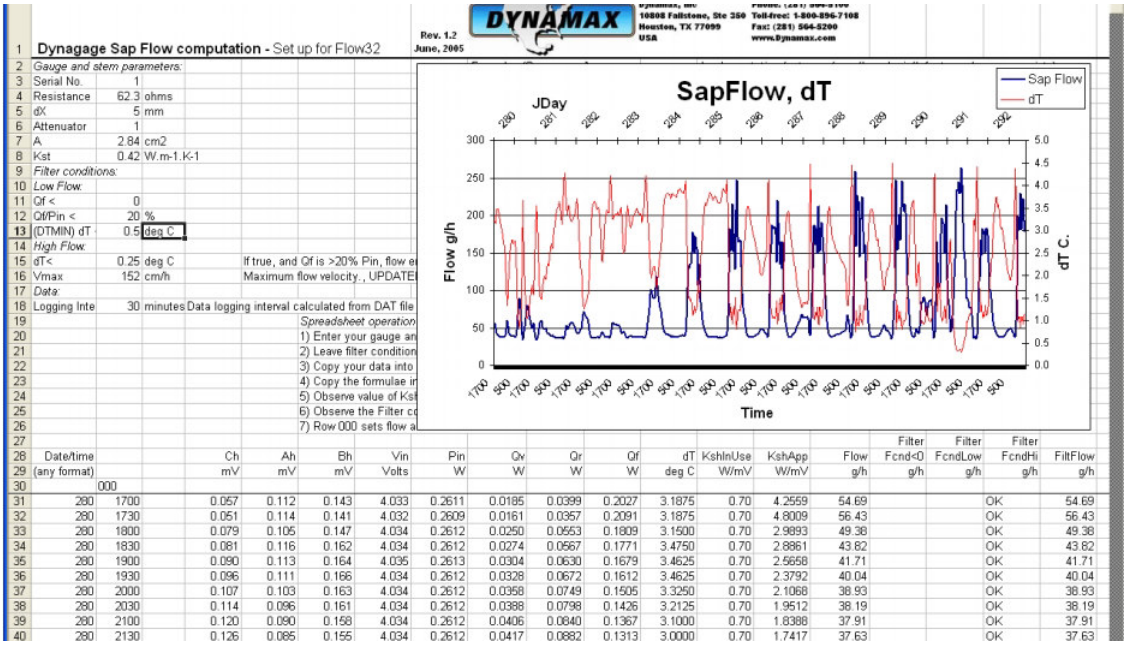
O sistema suporta até 8 sensores Dynagages podendo ser extensível até 32 com a aquisição de módulos de expansão cada qual suportando a adição de mais 8 sensores. O cálculo integrado ao dispositivo tem diferenças pelas variantes do tipo de planta e clima local, obrigando o pesquisador a utilizar uma planilha disponibilizada pelo fabricante para inserção de dados retirados do *logger*. Figura 3.7.

Figura 3.6: Sistema *Flow32-1k*



Fonte: Adaptado de (DYNAMAX, 2017).

Figura 3.7: Planilha para cálculo *Sap Flow*



DYNAMAX

Rev. 1.2
June, 2005

Dynagage Sap Flow computation - Set up for Flow32

Gauge and stem parameters:		
Serial No.	1	
Resistance	62.3	ohms
dX	5	mm
Attenuator	1	
A	2.84	cm2
Kst	0.42	W.m-1.K-1
Filter conditions:		
Low Flow:		
Qf <	0	
Qf/Pin <	20	%
(DTMIN) dT	0.5	deg C
High Flow:		
dT<	0.25	deg C
Vmax	152	cm/h
Data:		
Logging Inte	30	minutes

If true, and Qf is >20% Pin, flow e
Maximum flow velocity., UPDATE
Data logging interval calculated from DAT file

Spreadsheet operation
1) Enter your gauge an
2) Leave filter condition
3) Copy your data into
4) Copy the formulae in
5) Observe value of Ks
6) Observe the Filter c
7) Row 000 sets flow a

Date/time		Ch	Ah	Bh	Vin	Pin	Qv	Qr	Qf	dT	KshInUse	KshApp	Flow	Filter Fcnd<0	Filter FcndLow	Filter FcndHi	FiltFlow
(any format)		mV	mV	mV	Volts	W	W	W	W	deg C	W/mV	W/mV	g/h	g/h	g/h	g/h	g/h
	000																
280	1700	0.057	0.112	0.143	4.033	0.2611	0.0185	0.0399	0.2027	3.1875	0.70	4.2559	54.69			OK	54.69
280	1730	0.051	0.114	0.141	4.032	0.2609	0.0161	0.0357	0.2091	3.1875	0.70	4.8009	56.43			OK	56.43
280	1800	0.079	0.105	0.147	4.034	0.2612	0.0250	0.0553	0.1809	3.1500	0.70	2.9893	49.38			OK	49.38
280	1830	0.081	0.116	0.162	4.034	0.2612	0.0274	0.0567	0.1771	3.4750	0.70	2.8861	43.82			OK	43.82
280	1900	0.090	0.113	0.164	4.035	0.2613	0.0304	0.0630	0.1679	3.4625	0.70	2.5658	41.71			OK	41.71
280	1930	0.096	0.111	0.166	4.034	0.2612	0.0328	0.0672	0.1612	3.4625	0.70	2.3792	40.04			OK	40.04
280	2000	0.107	0.103	0.163	4.034	0.2612	0.0358	0.0749	0.1505	3.3250	0.70	2.1068	38.93			OK	38.93
280	2030	0.114	0.096	0.161	4.034	0.2612	0.0388	0.0798	0.1426	3.2125	0.70	1.9512	38.19			OK	38.19
280	2100	0.120	0.090	0.158	4.034	0.2612	0.0406	0.0840	0.1367	3.1000	0.70	1.8388	37.91			OK	37.91
280	2130	0.126	0.085	0.155	4.034	0.2612	0.0417	0.0882	0.1313	3.0000	0.70	1.7417	37.63			OK	37.63

Fonte: Adaptado de (DYNAMAX, 2017).

3.2 Conclusões acerca dos requisitos básicos do equipamento

O dispositivo proposto, assim como os demais dispositivos apresentados neste capítulo, consistem em análise de dados térmicos, logo o primeiro requisito básico que deve ser contemplado é um recurso para aquisição desses dados. Conforme visto na fundamentação teórica, os sensores utilizados são do tipo sonda em formato de agulhas, basicamente formadas por junções termopares em que uma dessas agulhas também trabalha com uma resistência acoplada, exercendo também a função de aquecedor, o qual promove uma diferença de temperatura entre os dois sensores que será analisada pelos pesquisadores para cálculo e utilização do fluxo de seiva nas pesquisas científicas ou como indicador para outras aplicações.

Dessa forma, o aspecto de coleta de dados advindos das leituras dos sensores sempre passa por um dispositivo *Datalogger* em que são armazenados e tratados para visualização dos usuários. No sistema proposto este aspecto será abordado de forma diferente excluindo o dispositivo *datalogger* porém mantendo a funcionalidade com outro recurso tecnológico.

A comunicação entre sensores, *loggers* e usuário final é claramente o ponto-chave da tecnologia, a natureza atípica da aplicação deve permitir alta escalabilidade (para uma futura e numerosa rede de sensores *wired* ou *wireless*), a utilização da provável pouca banda disponível para tráfego de dados, consumo energético otimizado e distribuição eficiente de informação.

Outro elemento importante a se observar nesse sistema é o módulo de alimentação utilizado: um *powerbank*, onde diferente das soluções empregadas nos produtos no mercado que embora eficientes demandam um aumento considerável no custo final do equipamento. No sistema proposto a utilização de uma alternativa mais barata e de fácil substituição será aplicada de forma a atender o requisito específico do cliente.

Os sistemas disponíveis no mercado apresentam conexão RS232, conectores USB ou cartões SD, o que possibilita a importação de informações do dispositivo. No protótipo proposto essa aquisição somente poderá ser realizada através de uso da comunicação na arquitetura extraindo diretamente da aplicação web..

Na Tabela 1 estão resumidas as principais características em comum dos dispositivos analisados e as que o protótipo deverá possuir para garantir recursos mínimos para atender as necessidades específicas do cliente.

A partir dessa análise foi possível estabelecer os requisitos mínimos de custo do protótipo, que irá tornar possível a experimentação inicial e homologação das tecnologias internas ao sistema. Importante salientar que, a ausência de GSM pode ser suprida com a instalação de módulo próprio disponível no mercado por um custo acessível e a de *datalogger* é um fator decisivo para barateamento do sistema.

Tabela 1: Especificações dos dispositivos do mercado e proposto

Requisitos	**Comparativo de Recursos**			
	ES-SYS-2	**SFM1**	**Flow32-1k**	**SAFS**
Aquisição de ΔT	X	X	X	X
Armazenamento	X	X	X	X
WiFi	-	X	-	X
Datalogger	X	X	X	-
GSM	X	X	X	-
Porta Auxiliar	DB9	USB	DB9 e USB	-
Bateria	12v	Interna	12v	PowerBank
Interface	X	X	X	X
Suporte Local	-	-	-	X
Custo	AUD$ 4.387,00	Não Fornecido	R$ 5.418,00	-

Fonte: Levantamento elaborado pelo autor.

A partir dos resultados deste levantamento se torna possível definir as alternativas tecnológicas responsáveis pelas diversas funções a serem integradas no sistema, a próxima seção trata da discussão dos aspectos a serem considerados antes das escolhas definitivas a compor o protótipo inicial.

3.3 Tecnologias existentes de Microcontroladores

Na seção anterior foram levantados especificações e funcionalidades de dispositivos para leitura de fluxo dados para aferição do fluxo de seiva disponíveis no mercado, todavia para compor o protótipo, será necessário o estudo e definição a partir de uma gama de tecnologias que se encaixem em cada aspecto e função do sistema. Este é o passo determinante para o protótipo da estação uma vez que a escolha da placa ou microcontrolador irá inferir na escolha de todos os demais componentes de *hardware* do protótipo.

Dentre as opções de placas microcontroladoras ou módulos, como são chamadas, duas foram selecionadas para análise com base nos requisitos do sistema idealizado. A estação servirá como uma *Thing* a se conectar com o sistema web e enviar e receber dados de sensores acoplados e informações derivadas geradas pelo sistema final.

Nesse contexto, um sistema operacional não se faz necessário pela necessidade mínima de processamento na estação e muito menos a necessidade de alguma aplicação mais elaborada embarcada. O suporte aos periféricos (no caso os sensores e algum tipo de *Display*) é considerado, inclusive, a situação de utilização de sensores analógicos ou digitais, quantidade de pinos disponíveis, suporte do dispositivo e custo.

3.3.1 Raspberry PI 3

A primeira placa analisada é a *Raspberry PI 3* desenvolvida pela fundação Raspberry PI, a placa agrega todo o hardware de processamento, memória e E/S em uma única plataforma, assemelhando-se a um computador. As características embora a coloquem dentro do nicho de *Single-Boards*, quando comparada a uma plataforma de microcontrolador como Arduino e um microcomputador tradicional, colocando a plataforma em uma categoria mais parecida com os *Smartphones* atuais.

A plataforma *Raspberry* conta com um processador de *clock* 1.2 Ghz de 64 Bits com quatro núcleos ARMv8, interfaces Wireless e Bluetooth em suas respectivas versões 802.11n e 4.1, quatro portas USB, 40 pinos GPIO, porta HDMI, porta Ethernet, conectores *jack* 3.5mm para áudio e vídeo, Interface de câmera e *display*(CSI e DSI), suporte à cartão de memória micro SD e processamento de vídeo 3D integrado (FOUNDATION, 2018).

Com essas especificações, é possível observar que a plataforma pode sozinha substituir um pequeno servidor de baixo-rendimento para aplicações simples e leves, porém a enorme gama de recursos integradas se mostra muito além das necessidades do sistema proposto, fugindo da arquitetura IoT pretendida em que vários *Nodes* possam ser agregados ao sistema final com os respectivos conjuntos de sensores, além disso o custo da placa se mostra bem acima das seguintes alternativas apresentadas.

Na Figura 3.8 está ilustrada a placa Raspberry Pi 3 que pode ser encontrada no mercado nacional em uma faixa de preço médio variando de R$ 190,00 à R$ 210,00, valor bem acima das placas mais simples e que podem atender aos requisitos do sistema.

Figura 3.8: Placa Raspberry PI 3

Fonte: Adaptado de (FOUNDATION, 2017).

3.3.2 Arduino UNO

Segundo D'Ausilio (2012), a placa Arduino UNO é uma placa com microcontrolador baseado no chip ATMEGA328. A placa oferece 14 pinos de *Input/Output* digitais, e seis

pinos de *input* analógico, além de um cristal de *clock* 16 Mhz, uma conexão USB e um conector *Jack* de alimentação.

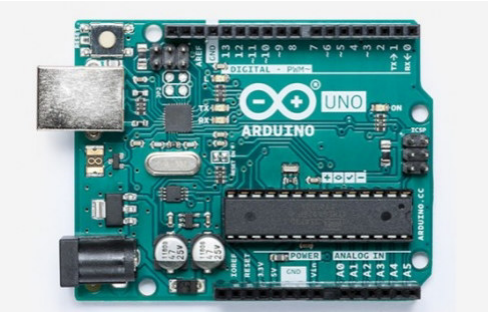
A placa pode ser alimentada pela conexão USB ou com uma fonte externa entre 6 e 20 Volts, e seus 14 pinos digitais podem funcionar como entrada ou saída a partir das funções *pinMode()*, *digitalWrite()* e *digitalRead()*, operando com 5 volts cada e proporcionando uma corrente máxima de 40 mA. Existe ainda um regulador de tensão que torna possível o fornecimento de tensão de 3.3 Volts para alguma necessidade do sistema.

O Arduino UNO possui seis portas analógicas fornecendo 10 bits de precisão (até 1024 valores), tendo como referência o terra aos 5 *Volts*, é possível mudar os limites de referência se utilizando de um pino AREF e a função *analogReference()*.

A plataforma também possui excelente variedade de módulos (chamadas de *Shields*) que podem ser adquiridos e acoplados ao sistema desempenhando as mais variadas funcionalidades, resultando em alternativas excelentes para atividades de automação, e com custo bem acessível.

A placa possui *Design* compacto contando com aproximadamente 68,6mm de comprimento e 53,3mm de altura e um custo médio de mercado de R$ 40,00 a R$ 50,00. Na Figura 3.9 é ilustrada a placa Arduino UNO:

Figura 3.9: Placa Arduino UNO



Fonte: Adaptado de (STORE, 2015).

Apesar das especificações condizentes com o protótipo, um fator decisivo contribuiu para não-adoção da plataforma na construção inicial: A comunicação com o Arduino UNO se faz apenas pela interface USB ou com uso de *Shields* para disponibilidade de outras interfaces de comunicação como RS232 ou WiFi, aumentando o custo do projeto e da complexidade na configuração. Além de que a placa Arduino UNO possui recursos que ficariam subutilizados no projeto da estação.

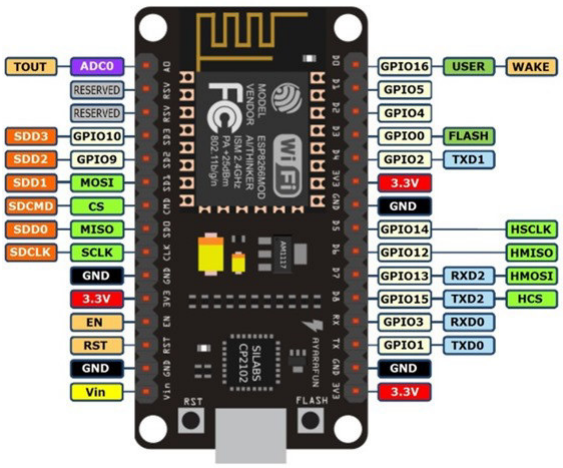
3.3.3 NodeMCU ESP8266

O modulo NodeMCU é uma plataforma IoT de código aberto, a qual inclui o *firmware* que roda no ESP8266 (chip de baixo custo com o protocolo TCP/IP (*Transmission*

Control Protocol/Internet Protocol) integrado) e um hardware baseado no modulo ESP-12 (a mais recente versão do ESP8266), o *firmware* usa a linguagem de programação Lua ou IDE arduino (NUNES et al.,).

A placa NodeMCU microcontrolador, trata-se de um semicondutor em forma de circuito integrado (CI) que possui memórias tanto voláteis como não-voláteis, além de portas de entrada e de saída digitais e analógica. Normalmente, é utilizado em tarefas específicas – principalmente em automação de sistemas (AITA, 2017). O NodeMCU é formado por um microprocessador dual core Tensilica Xtensa de 32 bits com suporte embutido à rede Wi-Fi (802.11) e *bluetooth* versão 4.2, e com a memória flash integrada. O microcontrolador possui 36 GPIOs (entradas ou saída) e 18 entradas A/D a depender do modelo e duas saídas D/A (AITA, 2017). Uma melhor compreensão do dispositivo pode ser observada na Figura 3.10:

Figura 3.10: Placa NodeMCU



Fonte: Adaptado de (ELETRONICOS, 2015).

Na Figura 3.10 está ilustrada a placa, na qual é possível perceber as suas dimensões, possuindo comprimento de 45 mm, largura de 25 mm e espessura de 7 mm. Apresentando capacidade plena para a aplicação, a consolidação no uso de aplicações IoT.

O custo altamente reduzido e possibilidades de agregação de novas tecnologias a placa em questão, permite ser escolhida para o projeto, pelo fato de atender as especificações necessárias não deixando aspectos subutilizados como os anteriores, ter uma comunidade de suporte ativa e aberta, existência de experimentações em projetos científicos e gerar economia ao custo final do projeto com o custo ainda mais reduzido que as alternativas anteriores.

Na tabela 2 é apresentado o custo médio de mercado das placas descritas, um aspecto que deve ser considerado junto aos já elencados anteriormente para a escolha da plataforma.

Tabela 2: Custo das placas analisadas

Modelo da placa	**Custo médio**
Raspberry PI 3	R$ 200,00
Arduino UNO	R$ 40,00
NodeMCU ESP8266	R$ 45,00

Fonte: Levantamento elaborado pelo autor.

Capítulo 4

Requisitos do Sistema

Neste capítulo será abordado a etapa inicial de concepção do sistema, definida pelas entrevistas e levantamento de requisitos com o cliente e elaboração de casos de uso, proporcionando uma melhor visão das funcionalidades a serem desenvolvidas.

4.1 Levantamento de Requisitos

Esta seção descreve os requisitos do sistema, os requisitos podem ser divididos em funcionais e não funcionais.

4.1.1 Requisitos Funcionais

Requisitos funcionais são aqueles que especificam as funcionalidades do sistema.

RF0001: Cadastro de Usuário

Descrição: O sistema deve permitir o cadastro de novos usuários.

Prioridade: Alta

RF0002: Editar Usuário

Descrição: O sistema deve permitir a edição de informações do cadastro do usuário.

Prioridade: Alta

RF0003: Excluir Usuário

Descrição: O sistema deve permitir a exclusão de um usuário.

Prioridade: Alta

RF0004: Associar Usuário ao Experimento

Descrição: O sistema deve permitir a associação de um usuário ao experimento desejado.

Prioridade: Alta

RF0005: Cadastro de Experimento

Descrição: O sistema deve permitir o cadastro de Experimentos.

Prioridade: Alta

RF0006: Excluir Experimento

Descrição: O sistema deve permitir a exclusão de um experimento.

Prioridade: Média

RF0007: Visualizar Dados

Descrição: O sistema deve permitir o usuário visualizar os dados referentes ao experimento ao qual está associado.

Prioridade: Alta

RF0008: Receber variação de temperatura

Descrição: A estação deve receber as leituras dos sensores e tratá-la de acordo com os demais requisitos.

Prioridade: Alta

RF009: Enviar Dados

Descrição: A estação deve enviar as leituras para a aplicação web a cada 10 minutos.

Prioridade: Alta

RF0010: Login

Descrição: O sistema deve coletar informações do usuário que realiza o acesso através do login.

Prioridade: Alta

RF0011: Relatórios

Descrição: O sistema deve gerar relatório por data (diário ou semanal) contendo gráfico informativo/ilustrativo com todas as medições realizadas durante o período selecionado.

Prioridade: Alta

RF0012: Relatórios

Descrição: O sistema deve gerar arquivo de saída para *download* no formato .csv, este arquivo só poderá ser baixado por usuários autenticados.

Prioridade: Média

4.1.2 Requisitos Não Funcionais

Os requisitos não funcionais são responsáveis por especificar as restrições sobre as funções providas pelo sistema.

RNF0001: Restrição Cadastro de Usuário

Descrição: Um usuário Administrador só pode ser cadastrado por outro Administrador.

RNF0002: Exclusão de usuários

Descrição: Um usuário só pode ter seu cadastro excluído por um Administrador.

RNF0003: Restrição de Cadastro de Experimento

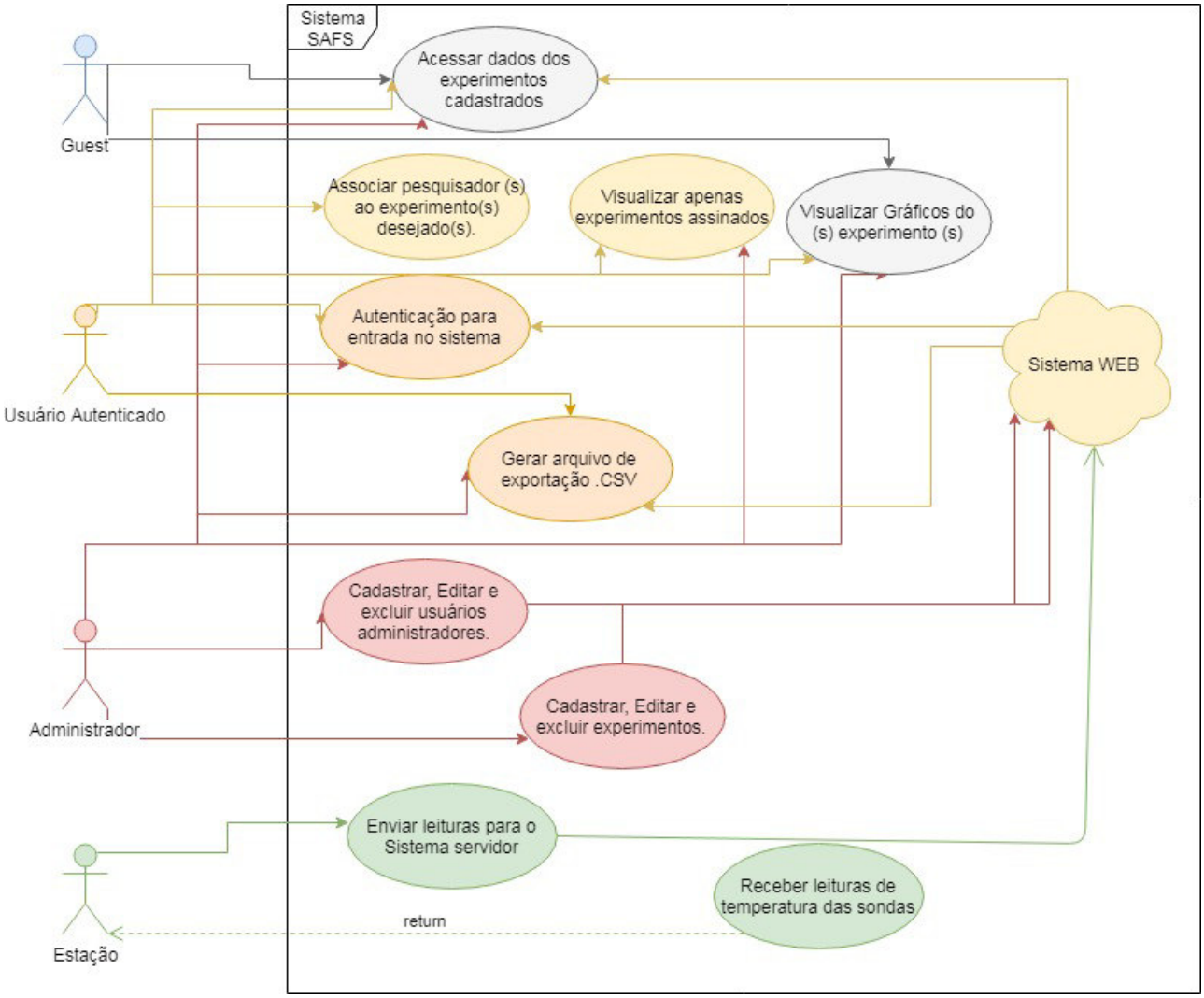
Descrição: Um experimento só pode ser cadastrado por um Administrador.

4.2 Casos de Uso

Os casos de uso são responsáveis por ilustrar possíveis situações em que o sistema pode ser colocado de acordo com as decisões do usuário. Na Figura 4.1 são ilustrados os casos de uso observados a partir dos requisitos e da concepção do sistema.

Cada ator do sistema (usuário convidado, autenticado, administrador e Estação) pode colocar o sistema em distintos estados a depender do tipo de interação, no diagrama

Figura 4.1: Diagrama de casos de uso SAFS



Fonte: Produzido pelo autor.

de casos de usos são mostrados estes estados, alguns compartilhados com outros atores e outros não.

Capítulo 5

Materiais e Métodos

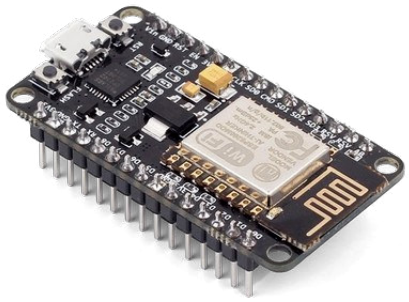
Neste capítulo será descrito todo o *hardware* utilizado na confecção do protótipo da estação assim como as etapas de desenvolvimento da aplicação embarcada e web.

5.1 Materiais utilizados

5.1.1 NodeMCU ESP8266

O módulo escolhido como plataforma para montagem do protótipo da estação foi o NodeMCU Esp8266 devido a comparação citada no capítulo anterior. Figura 5.1.

Figura 5.1: Placa NodeMCU Esp8266



Fonte: Adaptado de (ARTOFCIRCUITS.COM, 2015).

5.1.2 Display LCD 16x2

O *display* utilizado é um RT162-7, que possui *backlight* da cor azul e 16 colunas com 2 linhas, totalizando 32 espaços para caracteres. O componente pode ser visto na Figura 5.2.

Figura 5.2: *Display* 16x2 RT162-7

Fonte: Adaptado de (ORIENT DISPLAY CORP., 2014).

O *display* possui 16 pinos para conexão com outro hardware se utilizando de comunicação paralela seguindo a indicação de pinos conforme *datasheet* do componente como ilustrado na Figura 5.3.

Figura 5.3: Pinos do *display* RT162-7

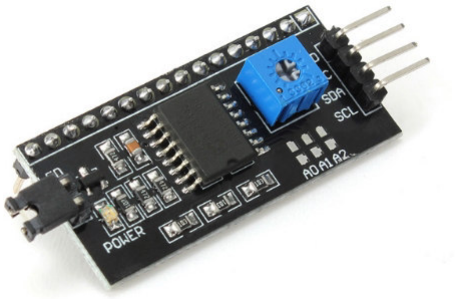
PIN CONNECTIONS			
PIN	**Symbol**	**Level**	**Function**
1	VSS		GND(0V)
2	VDD		Supply Voltage for Logic(+5V)
3	V0		Power supply for LCD
4	RS	H/L	H: Data# L: Instruction Code
5	R/W	H/L	H: Read# L: Write
6	E	H/L	Enable Signal
7	DB0	H/L	Data Bus Line
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	BL1		Backlight Power(+5V)
16	BL2		Backlight Power(0V)

Fonte: Adaptado de (ADAFRUIT INDUSTRIES, 2015).

5.1.3 Módulo adaptador de Interface Serial i2c

O módulo converte a comunicação paralela do *display* para o protocolo i2c resolvendo a questão das múltiplas conexões requeridas pelo *display*, visando a máxima economia de portas usadas na placa NodeMCU afim de que futuras implementações não encontrem problemas de disponibilidade, o módulo funciona utilizando apenas duas portas além das destinadas para GND e VCC. Figura 5.4.

Figura 5.4: Módulo Auxiliar I2c



Fonte: Adaptado de (ELETRONICOS, 2015).

5.1.4 Sensor de temperatura à prova d'água Ds18b20

Para testar o funcionamento do protótipo físico e do sistema foram utilizados dois sensores DS18B20 tipo sonda, o sensor se trata de um termômetro digital com resolução de 9 à 12 bits, sua faixa de medição varia de $-55\,°C$ à $125\,°C$ e garante precisão de $\pm -0.5\,°C$ na faixa de $-10\,°C$ à $85\,°C$ (TORRES et al., 2015).

Para acessar os valores do sensor se faz necessário um resistor *Pull-up* de 4.7 kΩ, resistores que servem para controlar a flutuação do sinal na porta de entrada, estes não podem ter valores muito altos pois o sinal não irá passar ou muito baixos pois uma corrente muito alta poderá ser direcionada a porta, queimando o microcontrolador. Na falta do componente foram utilizados resistores de 10 kΩ em série inicialmente, sendo substituídos pelos corretos no protótipo final.É necessário citar que, o sensor em questão utiliza um protocolo chamado *1-wire*, em que cada sensor possui um código serial único de 64 bits e através do protocolo vários destes podem ser conectados em um canal de comunicação único com o controlador. No caso do protótipo estes sensores serão usados unicamente para aquisição de dados no lugar das sondas específicas. O sensor é ilustrado na Figura 5.5.

Figura 5.5: Sensor tipo sonda DS18B20

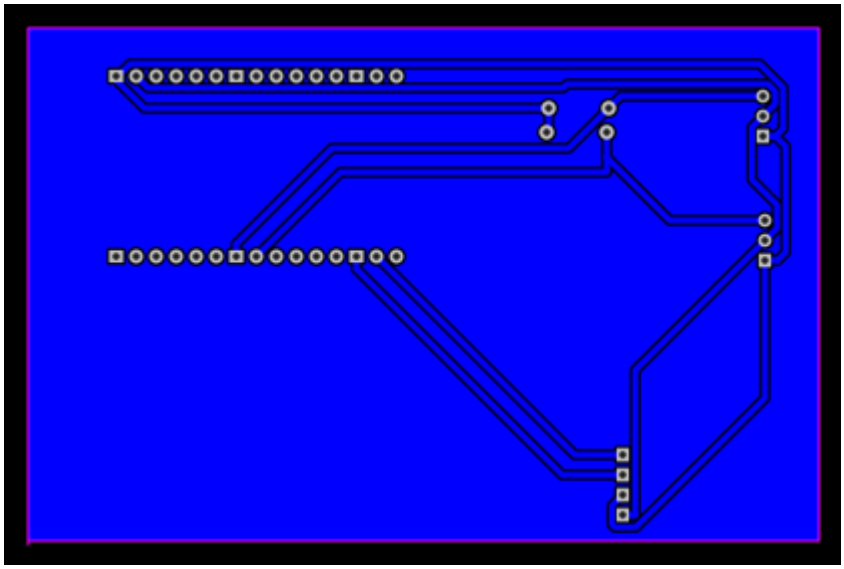
Fonte: Adaptado de (ELETRONICOS, 2015).

5.2 Métodos

5.2.1 Estação SAFS

No aspecto referente ao hardware utilizado no desenvolvimento do sistema,com os componentes de hardware descritos na seção 5.1, eles foram integrados em uma placa de circuito impresso projetada com o software livre *online EasyEDA*. O projeto da placa de circuito impresso é ilustrado na Figura 5.6 e foi prototipado com a LPKF ProtoMat® S62, uma *plotter* avançada de placas de circuitos de propriedade do Instituto de energia Elétrica da Universidade Federal do Maranhão (IEE/UFMA). O processo é mostrado na Figura **??**.

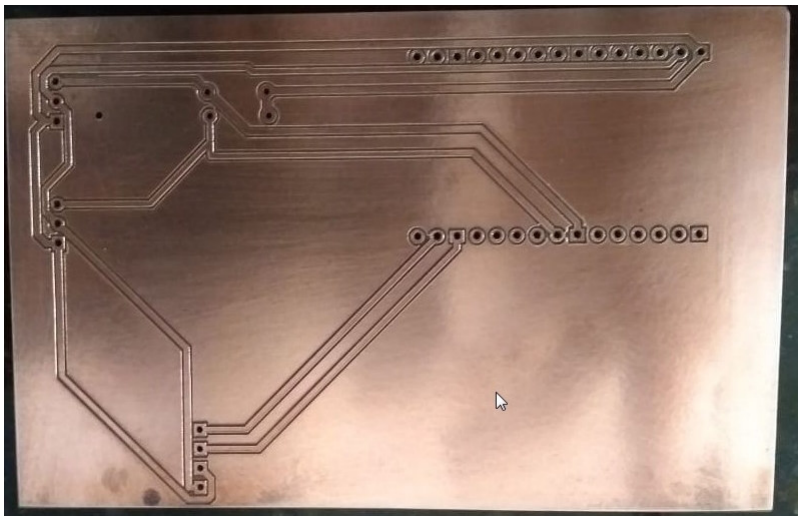
Figura 5.6: Projeto da placa de circuito impresso da estação



Fonte: Produzida pelo autor.

Na Figura 5.7 são ilustradas as trilhas e furos realizados pela *plotter* na qual serão encaixados e soldados os pinos GPIO do NodeMCU, módulo I2C e os pinos para conexão dos sensores utilizados no protótipo.

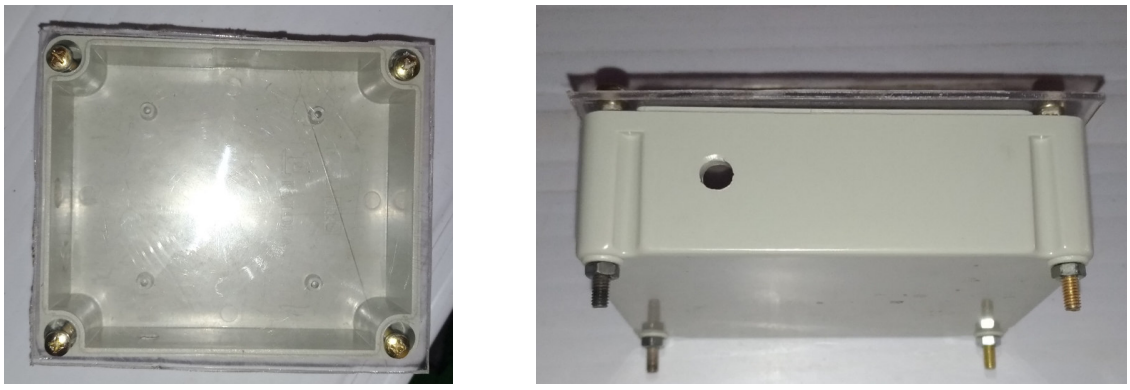
Figura 5.7: Placa de circuito impresso da estação



Fonte: Produzida pelo autor.

Após concluída a soldagem dos componentes, a placa é presa por parafusos dentro de um invólucro de PVC com orifícios para passagem e integração dos fios extensores dos sensores utilizados no protótipo, a caixa é selada com uma tampa de acrílico parafusada à caixa de PVC (Figura 5.8), ilustrando o invólucro terminado após corte do acrílico, furos e desgaste nas laterais da caixa.

Figura 5.8: Invólucro protetor da estação SAFS



(a) Visão de cima (b) Visão de baixo

Fonte: Elaborada pelo Autor.

O protótipo final da estação SAFS é ilustrado na Figura 5.9.

Figura 5.9: Protótipo final da estação SAFS

Fonte: Fabricada pelo autor.

5.2.2 Ambiente de Desenvolvimento

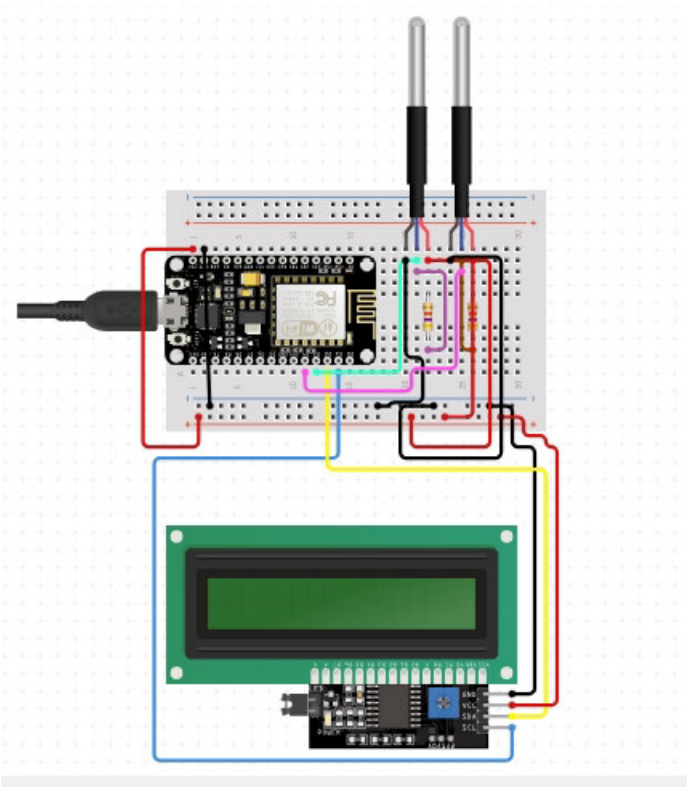
O software utilizado na programação do NodeMCU, foi a IDE do Arduino 1.8.2 para programação e operação dos sensores, VS Code para programação da aplicação web, além do pacote WAMP SERVER 3.1.3 contendo o servidor web *Apache* 2.4.33 com utilização de PHP 7.2.4 e MySql 5.7.21, ambos utilizados nesse projeto para a interface web e o banco de dados. Também foi utilizado o *Framework* Laravel 5.6.26 para direcionamento e agilização do projeto.

5.2.3 Código embarcado

O desenvolvimento do protótipo consistiu em duas etapas desenvolvidas paralelo: montagem do hardware e desenvolvimento do software: na primeira temos o uso dos materiais já descritos no capítulo deste trabalho e sua integração em um único circuito. Figura 5.10.

Na parte de desenvolvimento do software embarcado, tem-se na placa NodeMCU o controle de recebimento e envio das leituras de temperatura ao servidor, a exibição de informações no *display* local do protótipo. O código do servidor é responsável por receber as leituras e inserir no banco de dados, e disponibilizar as informações e recursos ao usuário através de interface web.

Figura 5.10: Montagem para experimentação em *protoboard*



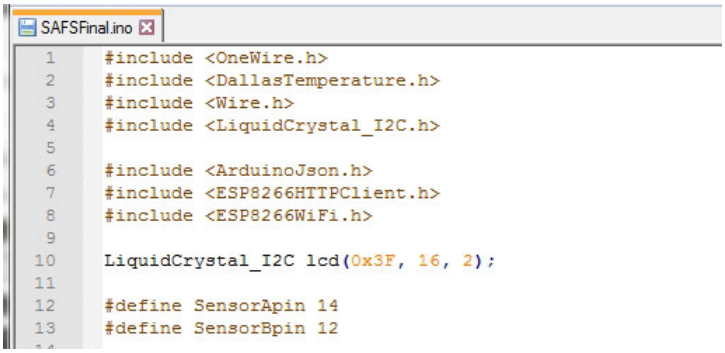
Fonte: Adaptado pelo autor.

A aplicação embarcada conta com rotinas básicas como importação de bibliotecas necessárias ao funcionamento, conexão e autenticação em uma rede *WiFi* disponível, definição de funções, inicialização de *Display* e sensores.

Funções específicas foram definidas, como as leituras, cálculo da variação ΔT, Atualizações do *Display*, criação de estrutura e configuração de mensagens JSON para envio ao servidor. As rotinas de comunicação existem para estabelecimento de parâmetros para conexão com o servidor, inserção de dados nas mensagens JSON e posterior envio destas. Existe também uma rotina verificadora de conexão, desconectando no caso de um determinado período de *timeout*.

Conforme ilustrado na Figura 5.11, foram utilizadas as bibliotecas *OneWire* e *DallasTemperature*, para utilização dos sensores, *LiquidCrystal_I2* e *Wire* para utilização do conversor de comunicação do *Display*, *ArduinoJson* para configuração, estabelecimento e envio das mensagens REST ao servidor. Além dessas, as bibliotecas *ESP8266HTTPClient* e *ESP8266WiFi* para configuração e comunicação do módulo via HTTP pela rede sem fio.

Figura 5.11: Bibliotecas importadas no projeto embarcado



Fonte: Produzido pelo autor.

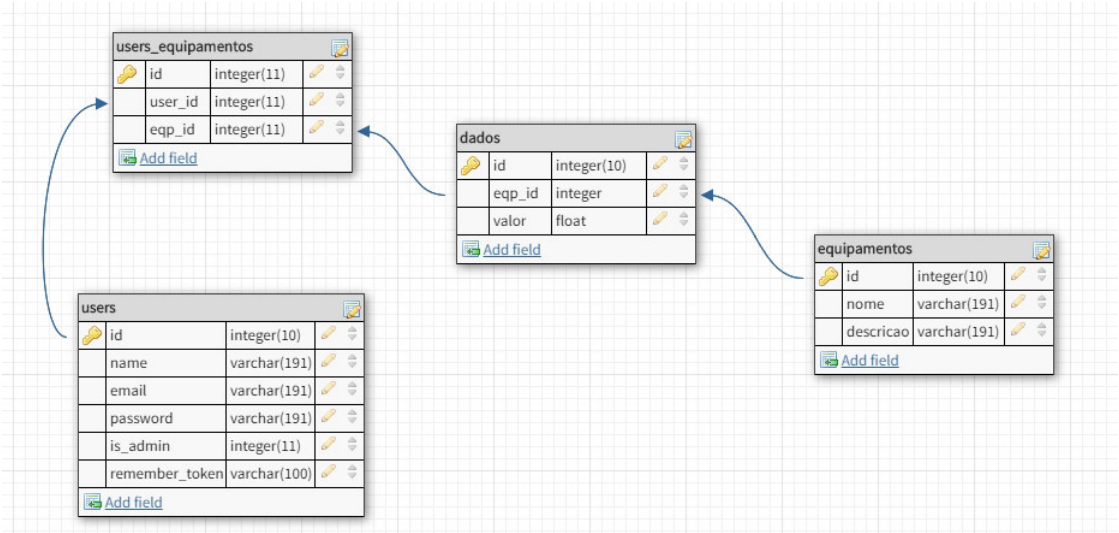
A função *LOOP* do código consiste em atender os requisitos de aquisição de dados, realizando medições de temperatura com *delay* de dez minutos e guardando a diferença entre os sensores em uma nova variável: o ΔT, que será enviado pelo método POST ao servidor em uma mensagem REST no formato JSON em uma URL definida como repositório de dados.

5.2.4 Base de Dados

O gerenciador de banco de dados utilizado foi o MySQL, hospedado em um servidor local. A principal finalidade do banco de dados do SAFS é armazenar dados do sistema, as configurações de login e senha, dados do experimento e as leituras da sonda.

Inicialmente, foi procedido a definição de escopo do sistema: temos dados (as leituras), temos equipamentos que geram dados (as estações NodeMCU) e temos usuários que acessam estes dados de forma distintas. O banco de dados foi projetado utilizando o modelo *Top-Down*. Esse modelo de projeto é executado partindo de um nível de abstração mais alto até atingir o mais baixo, com o máximo de detalhamento possível. O modelo de abstração utilizado foi o entidade relacionamento, no qual o mini mundo em questão é tratado como um conjunto de objetos conhecidos como entidades (os usuários, equipamentos e dados citados no inicio do parágrafo) e os relacionamentos entre estes objetos. As Tabelas do banco seguem o modelo conceitual (Figura 5.12).

Figura 5.12: Tabelas do Banco de dados do SAFS



Fonte: Produzido pelo autor.

Ainda na Figura 5.12, o banco de dados é formado por quatro entidades e seus relacionamentos, no caso, a entidade *user_equipamentos* foi formada pelo relacionamento de N para N entre usuários e equipamentos.

5.2.5 Sistemas e Componentes Externos Utilizados

Como aplicação externa há o uso do *Laravel*, responsável por direcionar a implementação do sistema web através do padrão MVC e disponibilizar classes para reuso mediante adaptação pelos desenvolvedores. O próprio *Framework* é o responsável pelo sistema de login e segurança.

Capítulo 6

Funcionalidades

Neste capítulo serão abordadas as principais funcionalidades do Sistema web do SAFS, sendo ilustradas e explicadas de forma a serem destacados os pontos discutidos no decorrer deste trabalho e principalmente atendendo os requisitos do cliente e seu problema. Importante frisar, como já explicado na fundamentação deste trabalho que o funcionamento do sistema utilizando as metodologias escolhidas e o *framework* LARAVEL, basicamente se resume à chamadas por serviços utilizando URLs (rotas) que são atendidos por um *controller* que chama uma função utilizando dados de entrada a partir de uma *view* ou no caso do SAFS, de mensagens JSON advindas da estação NodeMCU.

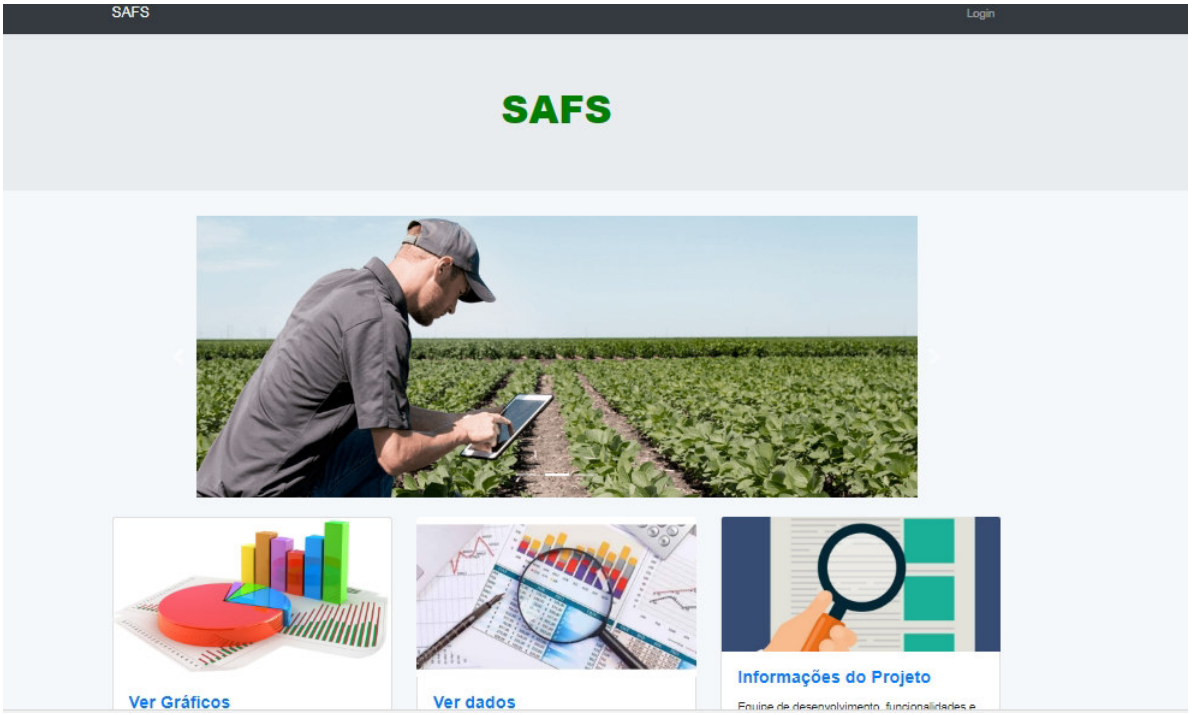
6.1 Interações de convidado

O usuário convidado, é qualquer pessoa com acesso a rede local do SAFS de posse da URL inicial. A ideia é não restringir o acesso à informação no âmbito da pesquisa. Por algum motivo no caso de atraso de cadastro , o usuário já pode ter alguma interação mínima com o sistema.

O usuário convidado pode interagir com o sistema conforme os casos de uso, podendo acessar as seguintes funções (Figura 6.1):

- Visualizar informações de equipe desenvolvedora, descrição e manual do sistema;
- Visualizar gráfico das leituras de estação NodeMCU;
- Visualizar leituras de equipamentos cadastrados no servidor;

Figura 6.1: Tela inicial para convidado



Fonte: Produzido pelo autor.

6.2 Interações de usuário autenticado

O usuário autenticado é uma pessoa já credenciada por outro usuário ou um administrador,ou seja a função é a mesma do usuário convidado sobre a restrição de acesso a informação sobre pesquisa, porém, o usuário autenticado possui algumas funcionalidades adicionais.

O usuário autenticado pode interagir com o sistema conforme os casos de uso, além das funcionalidades listadas para o usuário convidado pode (Figura 6.2):

- Realizar *download* dos dados de experimento e período selecionados em um arquivo .CSV;
- Cadastrar um novo usuário não-administrador;
- Editar os próprios dados;
- Assinar publicações de determinado equipamento para sua página personalizada de experimentos;

Figura 6.2: Tela inicial para usuário autenticado

Fonte: Produzido pelo autor.

6.3 Interações de administrador

O usuário administrador, nomeado por outro administrador seja na implantação inicial do sistema ou posteriormente, podendo utilizar de todas as funcionalidades citadas e algumas adicionais.

O usuário administrador pode interagir com o sistema conforme os casos de uso, além das funcionalidades listadas para o usuário convidado e autenticado ele também pode:

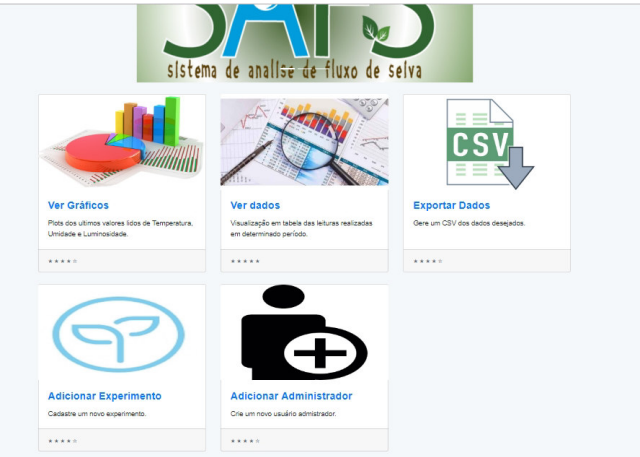
- Cadastrar, editar e excluir novo usuário;
- Cadastrar, editar e excluir novo usuário administrador;
- Cadastrar, editar e excluir novo equipamento (estação NodeMCU).

As funcionalidades implementadas no protótipo do SAFS conforme análise de requisitos formam recursos que podem ser acessados a depender da credencial como explicado anteriormente, algumas *Views* apresentam códigos diferentes a depender do tipo de acesso, mas isto não afeta a funcionalidade em si, apenas o acesso ou não ao recurso.

As seguir, as Figuras 6.4, 6.5, **??**, 6.6, e 6.6 mostram as telas que implementam as funcionalidades do SAFS, começando pela principal funcionalidade: a seleção de uma

Figura 6.3: Tela inicial de administrador

(a) Novos itens no menu suspenso



(b) Novos botões de ação

Fonte: Adaptado pelo Autor.

determinada estação e a visualização das medições (Figuras 6.4 e 6.5). Na seleção de equipamentos, o usuário autenticado e administrador podem assinar os equipamentos que quiserem para sua página personalizada e agregando o *link* do gráfico correspondente a cada estação, já o convidado não possui esse recurso, o que pode ser um incômodo a depender da quantidade de equipamentos cadastrados. Na tela de visualização das leituras há três colunas: o ID do equipamento utilizado, a leitura de ΔT e os dados de data e horário de cada uma.

Figura 6.4: Tela de seleção de equipamentos para visualização de dados

Fonte: Produzido pelo autor.

Figura 6.5: Tela de visualização de dados do equipamento selecionado

Fonte: Produzido pelo autor.

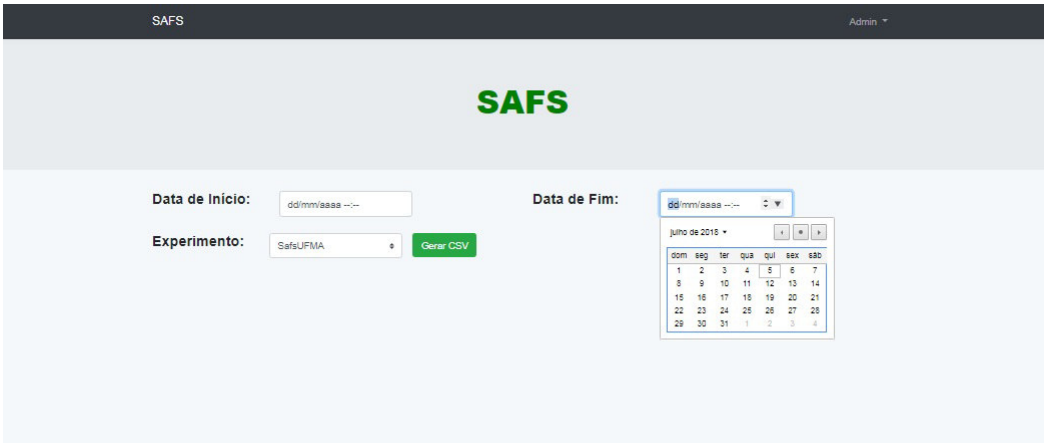
Figura 6.6: Tela de visualização de gráfico de estação selecionada

Fonte: Produzido pelo autor.

Outra funcionalidade (6.7), foi implementada seguindo um dos principais requisitos do cliente: a necessidade de um arquivo .csv para posterior manipulação de dados. Como informado o usuário autenticado e o administrador possuem acesso a este recurso. Na

tela descrita temos campos para seleção do usuário da data e hora iniciais até a data e hora que deseja limitar as informações no arquivo, assim como de qual estação cadastrada deseja baixar os dados.

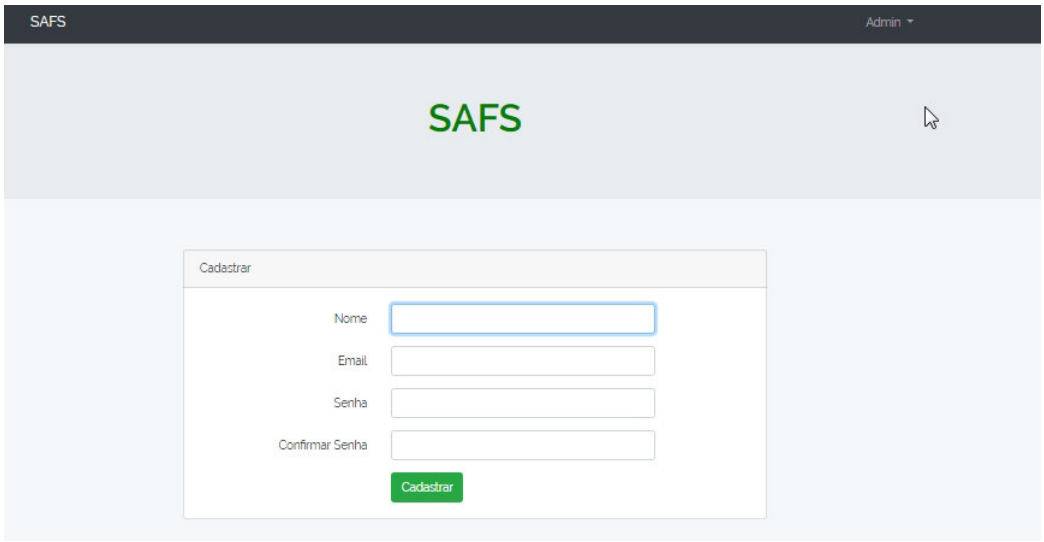
Figura 6.7: Tela exportação de arquivo .csv



Fonte: Produzido pelo autor.

Uma funcionalidade disponível apenas para usuário administrador é a adição de outro administrador no sistema, que poderá cadastrar novos equipamentos pode ser vista na Figura 6.8.

Figura 6.8: Tela adição de usuário administrador



Fonte: Produzido pelo autor.

Capítulo 7

Resultados e Discussões

Neste capítulo serão apresentados os resultados dos testes realizados em *hardware* e *software*, observações acerca do período experimental e protótipo e propostas para melhorias futuras.

7.1 Testes e Avaliação

Os testes do sistema se deram de maneira a homologar três aspectos: funcionamento da estação remota, conexão entre estação e servidor e funcionalidades do servidor web.

Uma vez montado o protótipo final, foram apagados os registros do banco de dados anteriores dos testes em *protoboard*, criado um ponto de ancoragem local pelo celular, configuradas credenciais no código embarcado e iniciadas novas leituras em meios de temperaturas distintas (copos com água, sendo um com gelo) que foram monitoradas pela estação no período de duas horas, ou seja, 12 leituras de ΔT (Figura 7.1).

Uma vez que as mensagens mostradas no visor LCD do protótipo indicam o êxito na leitura e envio. Dois novos usuários foram criados (um administrador e outro não), e os testes prosseguiram com cada credencial de acesso (convidado, usuário autenticado e administrador).

As funcionalidades do sistema foram testadas e alguns erros detectados no tocante a erro de rotas e links, corrigidos em função da organização provida pelo *framework*, desta forma o sistema conseguiu receber todas as medições da estação no tempo proposto e disponibilizar a informação ao usuário conforme esperado na concepção das funcionalidades seguindo os requisitos impostos (7.2).

Uma vez concluídos, os testes aferem os aspectos propostos para homologação: estação remota, comunicação com o servidor, sistema web.

Figura 7.1: Experimento para teste do protótipo final

Fonte: Produzido pelo autor.

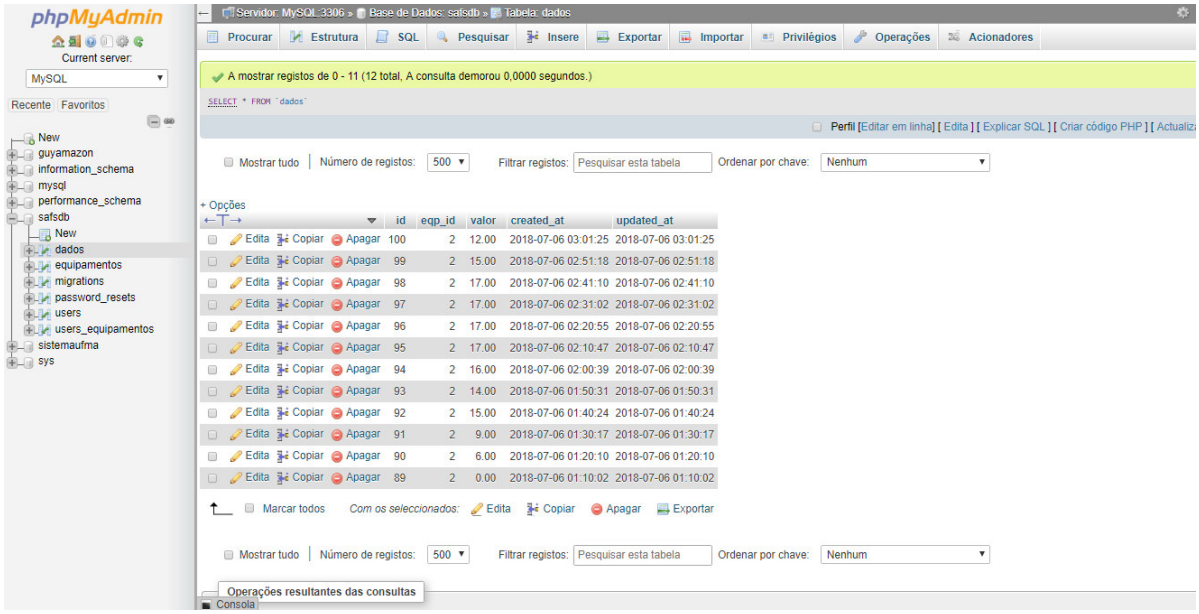
7.2 Desempenho em ambiente controlado

O Sistema foi operado e testado por uma pesquisadora da área lotada na Pós-graduação da Universidade Estadual do Maranhão, no qual recebeu uma breve introdução de como operar o sistema, explorou as funcionalidades em cada nível de credencial (convidado, autenticado e administrador) e encontrou alguns problemas já corrigidos, a partir do *feedback* e sugestões novas funcionalidades já podem ser estudadas para implementação em uma nova versão. No geral a pesquisadora se mostrou satisfeita com o uso do SAFS, uma observação importante é que já foi usuária do dispositivo da *Edaphic Scientific* e relatou principalmente a dificuldade de acesso aos dados.

7.3 Observações sobre o experimento

Foi observado um ganho significante de tempo no desenvolvimento do sistema web e na detecção e correção de erros, tudo devido à arquitetura orientada a serviços utilizando o REST e seus verbos, além das bibliotecas de reuso e das classes e organização de códigos em diretórios do LARAVEL. Ainda sobre o *Framework* foi observado que a interação com *tablets* e celulares necessita do uso a implementação de um filtro verificador para redirecionamento das rotas para dispositivos móveis. A adição de novos recursos, assim

Figura 7.2: Leituras registradas na duração do experimento



(a) Leituras registradas no Banco de dados

ID	Temperatura	Horário
100	12	2018-07-06 03:01:25
99	15	2018-07-06 02:51:18
98	17	2018-07-06 02:41:10
97	17	2018-07-06 02:31:02
96	17	2018-07-06 02:20:55
95	17	2018-07-06 02:10:47
94	16	2018-07-06 02:00:39
93	14	2018-07-06 01:50:31
92	15	2018-07-06 01:40:24
91	9	2018-07-06 01:30:17
90	6	2018-07-06 01:20:10
89	0	2018-07-06 01:10:02

Voltar

(b) leituras mostradas na aplicação

Fonte: Adaptado pelo Autor.

como a edição dos já existentes se mostrou amigável e sem custos de retrabalho para dependências.

Foi observado que o fuso-horário selecionado durante a instalação do projeto através do *framework* deve ser um item a ser verificado, posteriormente os horários de leitura recebidos e disponibilizados sofreram alteração significativa de três horas, a correta configuração do Laravel no arquivo app . php acerca da *timezone* se mostra importante para este tipo de projeto.

7.4 Custo do dispositivo

O custo final do protótipo apresentado neste trabalho pode pode ser estimado na Tabela 3, os valores podem variar no mercado mas neste projeto foram listados os valores médios de mercados de vendas *online*.

Tabela 3: Custo do Sistema SAFS

Componente	**Custo médio**
Placa NodeMCU Esp8266	R$ 45,00
Placa de circuito impresso (cobre)	R$ 10,00
Display Backlight Blue 16x2	R$ 12,00
Módulo I2C	R$ 11,00
Sensores DS18B20 * (par)	R$ 30,00
Invólucro de proteção com tampa transparente	R$ 25,00
PowerBank 10.000mAh **	R$ 60,00
Outros custos ***	R$ 50,00
TOTAL	**R$ 165,00**

Fonte: Levantamento feito pelo autor.

*: Sensores adicionados para efeito ilustrativo do custo do protótipo, o produto final agrega os custos dos sensores específicos.

**: Caso a operação do sistema ocorra em ambiente controlado com disponibilidade de alimentação esse custo não é necessário.

***: Custos diversos estimados com transporte, pequenos componentes, ferramentas e ajustes terceirizados.

De acordo com a Tabela 3, é possível observar que, o valor do protótipo do SAFS ficou bem abaixo dos dispositivos com funções semelhantes, ponto principal levantado pelo cliente, mesmo o custo dos sensores específicos de seiva sendo agregado a este valor (cerca R$ 340,00 cada no fabricante mais barato) o custo do sistema ainda compensaria, especialmente mediante os outros diferenciais, como o sistema web de supervisão, o suporte local, o hardware de fácil substituição além de possibilidades de implementações futuras à pedido do cliente. O sistema SAFS se mostra uma plataforma IoT altamente escalonável e versátil. É necessário salientar que esses custos contemplam o protótipo, sendo adicionados valores referentes à mão-de-obra de desenvolvimento e custos de *design* final, porém, o custo final ainda torna a solução viável perante o custo dos outros produtos citados neste trabalho.

Capítulo 8

Conclusão

O protótipo se mostrou funcional dentro de seu escopo, aplicando conceitos de eletrônica e dispositivos embarcados junto a uma arquitetura orientada à serviços. A estação remota conseguiu se comunicar com o servidor utilizando o protocolo de transporte HTTP e enviando os dados esperados seguindo a arquitetura proposta: a REST, mensagens em formato JSON entregam os dados se utilizando de URLs, ou rotas que acabam funcionando como serviços distintos a serem oferecidos para os consumidores.

O uso do *framework* no desenvolvimento web em PHP se mostrou eficiente em todos os aspectos que foram explorados da ferramenta neste projeto. Através dos benefícios do uso do padrão MVC e da simplicidade de instalação e configuração inicial, além da divisão em diretórios, o reuso de código e as funcionalidades nativas contribuíram para que o sistema fosse concluído de forma funcional em seu produto mínimo viável.

O protótipo apresentado neste trabalho não abordou o desenvolvimento de sondas específicas por conta do tempo e recursos disponíveis, porém, trabalhos futuros poderão abordar o desenvolvimento ou a integração com sondas já desenvolvidas.

O SAFS se mostrou um produto compacto, escalonável, de fácil usabilidade e principalmente barato, quando comparado aos equipamentos do segmento. O desenvolvimento de sondas específicas que se integrem ao sistema no lugar dos sensores DS18B20, ou em adição a estes, é um ponto a ser discutido e estudado em trabalhos futuros, além do aperfeiçoamento das tecnologias aqui apresentadas e implementação de novos recursos, como por exemplo, o uso de sensores próprios para medição de umidade do solo integrados ao sistema, a versão *mobile* do sistema ou mesmo sistemas distintos porém com comunicação possível através das arquiteturas aqui descritas se utilizando da informação: um sistema automatizado de irrigação baseado no coeficiente de desgate hídrico das plantas, coeficiente este que leva em consideração o fluxo de seiva vegetal.

No tocante aos objetivos gerais deste trabalho, todos foram alcançados assim como

os objetivos específicos propostos. A partir da observação de um problema e também necessidade, as etapas para concepção e fabricação de uma solução envolvendo variados aspectos de eletrônica e tecnologia de informação foram numerosas e degastantes, se fazendo necessário a compreensão do problema, suas especificidades, requisitos do cliente, levantamento teórico e pesquisa de produtos similares no mercado, de tecnologias que poderiam ser utilizadas e como as utilizar. Neste trabalho podemos constatar a aplicação de conhecimentos adquiridos ao longo de toda formação do engenheiro da computação frente a um desafio da realidade do mercado de trabalho, exigindo organização, observação e experimentação como métodos acadêmicos e aplicação de conceitos e técnicas assimilados na formação e na experiência com tecnologias diversas durante este período.

Para trabalhos futuros temos então a pesquisa e desenvolvimento da sonda específica *SAP-FLOW* ou integração com sondas já desenvolvidas em outros trabalhos, o aperfeiçoamento da interface gráfica e de funcionalidades como o relatório em formato .csv, a implementação do filtro *mobile*, oestudo sobre a integração de outro tipos de sensores ao sistema e a implantação de recursos na estação como suporte ao cartão SD e um *Real Time Clock* (RTC).

Este trabalho contribui para homologação e difusão do uso de sistemas embarcados unidos à tecnologia web através de protocolos de comunicação e arquitetura orientada à serviços. É mostrado que o conjunto de tecnologias pode ser utilizado para resolução das mais variadas problemáticas e aplicações acadêmicas e comerciais.

Referências Bibliográficas

ADAFRUIT INDUSTRIES. *Esquema de pinos do Display 18x2*. 2015. Disponível em: <http://cdn-shop.adafruit.com/datasheets/RT162-7.pdf>. Acesso em: 22 jun. 2018. Citado na página 48.

AITA, R. H. Sistema de irrigação localizada e automatizada. 2017. Citado na página 41.

ARTOFCIRCUITS.COM. *Esquema de pinos do NodeMCU ESP-8266*. 2015. Disponível em: <http://artofcircuits.com/product/nodemcu-amica-lua-r2-esp8266-wifi-module>. Acesso em: 22 jun. 2018. Citado na página 47.

ATTORRE, B. F. M. *Web Services Soap vs REST*. 2015. Disponível em: <URL https://www.devmedia.com.br/web-services-rest-versus-soap/32451>. Acesso em: 22 jun. 2018. Citado na página 22.

BURGESS, U. S. *User Manual*. 2017. Disponível em: <http://www.ictinternational.com-/content/uploads/2014/03/SFM1-Manual-v5.pdf>. Acesso em: 20 mai. 2018. Citado 2 vezes nas páginas 34 e 35.

CODENUCLEAR. *SOAP Web Services*. 2018. Disponível em: <http://www-.codenuclear.com/soap-web-services/>. Acesso em: 01 jul. 2018. Citado na página 24.

COHEN, Y.; FUCHS, M.; GREEN, G. Improvement of the heat pulse method for determining sap flow in trees. *Plant, Cell & Environment*, Wiley Online Library, v. 4, n. 5, p. 391–397, 1981. Citado 2 vezes nas páginas 32 e 33.

DYNAMAX, U. *Flow32-1k Manual*. 2017. Disponível em: <http://dynamax.com/images-/uploads/papers/Flow32-1K_Manual.pdf>. Acesso em: 20 mai. 2018. Citado na página 36.

D'AUSILIO, A. Arduino: A low-cost multipurpose lab equipment. *Behavior research methods*, Springer, v. 44, n. 2, p. 305–313, 2012. Citado na página 39.

FERREIRA, P. B. V. *Arquitetura REST em smartphones Android*. Tese (Doutorado), 2015. Acesso em: 22 jun. 2018. Citado na página 25.

FERREIRA[1], C. d. F.; MOTA[1], R. D. Comparando aplicação web service rest e soap. 2014. Citado 2 vezes nas páginas 24 e 26.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. Citado na página 25.

FOUNDATION, R. P. *Placa Raspberry PI 3*. 2017. Disponível em: <https:/-/www.raspberrypi.org/products/>. Acesso em: 14 jun. 2018. Citado na página 39.

FOUNDATION, R. P. *Raspberry Pi Hardware*. 2018. Disponível em: <http://www-.raspberrypi.org/documentation/hardware/>. Acesso em: 14 jun. 2018. Citado na página 39.

FRANCO, E.; PIEDADE, M.; RÊGO, R. Protótipo de um framework mvc para aplicações php de pequeno porte. *Anais do Encontro Regional de Computação e Sistemas de Informação*, 2014. Citado 2 vezes nas páginas 28 e 29.

GRANIER, A. Une nouvelle méthode pour la mesure du flux de sève brute dans le tronc des arbres. In: EDP SCIENCES. *Annales des Sciences forestières*. [S.l.], 1985. v. 42, n. 2, p. 193–200. Citado na página 22.

GRANIER, A. Evaluation of transpiration in a douglas-fir stand by means of sap flow measurements. *Tree physiology*, Heron Publishing, v. 3, n. 4, p. 309–320, 1987. Citado na página 32.

GUEDES, A. *BI Operacional: O BI de próxima geração*. 2013. Disponível em: <https://imasters.com.br/devsecops/bi-operacional-o-bi-de-proxima-geracao-parte-02>. Acesso em: 01 jul. 2018. Citado 2 vezes nas páginas 22 e 23.

INTERNATIONAL, I. *SFM1 Sap Flow System*. 2018. Disponível em: <www-.ictinternational.com>. Acesso em: 20 mai. 2018. Citado na página 34.

JÚNIOR, G. d. S. S. et al. Respostas biométricas, ecofisiológicas e nutricionais em genótipos diplóides de bananeira (musa spp) submetidos à salinidade. Universidade Federal Rural de Pernambuco, 2007. Citado na página 21.

LUTTGE, U. Interaction of stress factores and the middaydepression in plants with c3: photosynthesis. *Physiological Ecology of Tropical Plants*, v. 35, p. 122–124, 1997. Citado na página 21.

MARSHALL, D. Measurement of sap flow in conifers by heat transport. *Plant physiology*, American Society of Plant Biologists, v. 33, n. 6, p. 385, 1958. Citado na página 21.

MENDES, M. A. d. S. estilos arquiteturais web baseados em padrões abertos w3c. *Governança da internet e a atuação brasileira*, p. 85, 2014. Citado na página 27.

MORAES, J. A. P. V. de; PRADO, C. H. B. de A. Photosynthesis and water relations in cerrado vegetation. *Oecologia Brasiliensis*, Universidade Federal do Estado do Rio de Janeiro (UNIRIO), v. 4, n. 1, p. 3, 1998. Citado na página 21.

NICOLAS, E. et al. Evaluation of transpiration in adult apricot trees from sap flow measurements. *Agricultural water management*, Elsevier, v. 72, n. 2, p. 131–145, 2005. Citado na página 17.

NUNES, J. H. S. et al. Aplicaç ao da arquitetura soa em sistemas embarcados para iot. Citado 3 vezes nas páginas 17, 27 e 41.

ORIENT DISPLAY CORP. *Display 18x2 Blue Backlight*. 2014. Disponível em: <http://www.orientlcd.com/AMC1602BR_B_B6WTDW_I2C_16x2_Char_in_I2C_p-/amc1602ar-b-b6wtdw-i2c.htm>. Acesso em: 22 jun. 2018. Citado na página 48.

PAUTASSO, C. *REST vs SOAP, Making the right architetural decision*. 2008. Disponível em: <http://www.slideshare.net/cesare.pautasso/rest-vs-soap-making-the-right-architectural-decision-1st-international-soa-symposium-amsterdam-october-2008-presentation>. Acesso em: 01 jul. 2018. Citado na página 25.

REICHARDT, K.; TIMM, L. C. *Solo, planta e atmosfera: conceitos, processos e aplicações*. [S.l.]: Manole Barueri, 2004. Citado na página 21.

REIS, F. D. O.; CAMPOSTRINI, E.; SOUSA, E. F. D. Fluxo de seiva xilemática em mamoeiro'golden'cultivado por microaspersão sobre copa: relações com as variáveis ambientais. *Bragantia*, Instituto Agronômico de Campinas, v. 68, n. 2, 2009. Citado na página 17.

RICHARDSON, L.; RUBY, S. *RESTful web services*. [S.l.]: "O'Reilly Media, Inc.", 2008. Citado na página 27.

SAKURATANI, T. A heat balance method for measuring water flux in the stem of intact plants. *Journal of Agricultural Meteorology*, The Society of Agricultural Meteorology of Japan, v. 37, n. 1, p. 9–17, 1981. Citado na página 21.

SANTOS, C. et al. Ambientes inteligentes aplicadosa agricultura de precisao. 2016. Citado na página 16.

SCIENTIFIC, E. *Edaphici Scientific Environmental Monitoring System*. 2018. Disponível em: <www.edaphic.com.au/>. Citado 2 vezes nas páginas 33 e 34.

SILVA, M. *Desenvolvimento de sensor de fluxo de seiva e de coeficiente indicador de estresse hídrico para plantas de cafeeiro arábica. 2008. 114f.* Tese (Doutorado) — Tese (Doutorado em Produção Vegetal)-Universidade Estadual do Norte Fluminense, RJ.[Links], 2008. Citado 3 vezes nas páginas 17, 21 e 22.

STORE, A. *Placa Arduino UNO*. 2015. Disponível em: <https://store.arduino.cc/arduino-uno-rev3>. Acesso em: 14 jun. 2018. Citado na página 40.

TORRES, J. D. et al. Aquisição de dados meteorológicos através da plataforma arduino: construção de baixo custo e análise de dados. *Scientia Plena*, v. 11, n. 2, 2015. Citado na página 49.

VAQQAS, M. Restful web services: A tutorial. *Dr. Dobb's*, 2014. Citado 2 vezes nas páginas 26 e 27.

ZABADAL, B. M.; CASTRO, B. F. L. M. de. Iot e seus principais desafios. *Revista Interdisciplinar de Tecnologias e Educação*, v. 3, n. 1, 2017. Citado na página 16.