UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
DOUTORADO EM BIOMETRIA E ESTATÍSTICA APLICADA

FILIPE MENDONÇA DE LIMA

Construção de estação de monitoramento de temperatura e umidade do ar com sistema de detecção multiponto de umidade do solo e energia reversa

RECIFE
2024

FILIPE MENDONÇA DE LIMA

Construção de estação de monitoramento de temperatura e umidade do ar com sistema de detecção multiponto de umidade do solo e energia reversa

Tese apresentada ao Programa de Pós-Graduação em Biometria e Estatística Aplicada da Universidade Federal Rural de Pernambuco, como requisito para a obtenção do título de Doutor em Biometria e Estatística Aplicada.
Orientador: Dr. Moacyr Cunha Filho
Co-orientador: Dr. Leandro Ricardo Rodrigues Lucena

RECIFE
2024

Filipe Mendonça de Lima

Construção de estação de monitoramento de temperatura e umidade do ar com sistema de detecção multiponto de umidade do solo e energia reversa

Tese apresentada ao curso de Doutorado em Biometria e Estatística Aplicada do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco, para obtenção do grau de Doutor em Biometria e Estatística Aplicada.

Data de Aprovação: 19 de Agosto de 2024

Banca Examinadora:

Dr. Moacyr Cunha Filho
Universidade Federal Rural de Pernambuco - Orientador

Dr. Leandro R. R. de Lucena
Universidade Federal Rural de Pernambuco

Dr. Lucian Bogdan Bejan
Universidade Federal Rural de Pernambuco

Dr. Victor Casimiro Piscoya
Universidade Federal Rural de Pernambuco

Dr. Renisson Neponuceno de Araújo Filho
Universidade Federal do Tocantins

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

L732c Lima, Filipe Mendonça de
Construção de estação de monitoramento de temperatura e umidade do ar com sistema de detecção multiponto de umidade do solo e energia reversa / Filipe Mendonça de Lima. - 2024.
92 f. : il.

Orientador: Moacyr Cunha Filho.
Coorientador: Leandro Ricardo Rodrigues Lucena.
Inclui referências e apêndice(s).

Tese (Doutorado) - Universidade Federal Rural de Pernambuco, , Recife, 2024.

1. DHT11. 2. FC-28. 3. Arduino. 4. Agrometeorologia. I. Filho, Moacyr Cunha, orient. II. Lucena, Leandro Ricardo Rodrigues, coorient. III. Título

CDD

DEDICATÓRIA

Dedico esse texto ao meu pai, Luiz Carlos de Lima, minha tia/madrinha Severina Creuza Mendonça Lindoso e Maria Aurenice dos Santos Silva. Foram cedo demais.

AGRADECIMENTOS

Eu demorei para entender o que significava de fato "estar sobre os ombros de gigantes", como disse Isaac Newton. Queria agradecer aos gigantes aos quais eu fiquei em cima para chegar até aqui. Sou uma pessoa grata a tanta gente porque sei que não cheguei aqui sozinho.

À minha esposa Ana, por ter sido meu farol quando eu estava à deriva. Aos meus filhos Luís Filipe (*Pipo*) e Alice, por terem sido os motores desse barco.

À minha mãe, Maria José, minha irmã Mirna Juliana, meu pai Luiz Carlos (*in memoriam*) e minha avó Creuza, por sempre terem sempre estado à disposição pra me ajudar. Não poderia ter tido uma família melhor.

Ao meu orientador Moacyr Cunha Filho, pela sua paciência e grande liberdade e apoio que me deu durante o trajeto, e ter ensinado muito mais coisas para mim do que se pode imaginar. Espero que mais alunos tenham orientadores como ele.

Aos meus professores que tive no Programa: Antônio Samuel, Frank Sinatra, Guilherme Moreira, José Aleixo, Josimar Mendes e Lucian Bogdan pelo empenho em fazer com que as aulas fossem ótimas sempre. Aprendi muito.

Ao ortopedista do secretário do programa Marco Santos, por cuidar tão bem das costas da pessoas que carrega o programa nas costas.

Aos colegas Leandro R.R. Lucena e Maurício Leite, da Unidade Acadêmica de Serra Talhada, pela pesquisa deles ter sido o motivador principal para eu fazer esse doutorado.

Às minhas professoras Maria Eulália, Márcia Dantas e Maité Kulesza que foram de grande apoio para que eu fosse fazer pós-graduação e acreditaram sempre.

Meus colegas da área de Matemática da UAST: Antônio Viana, Demácio Costa, Felipe Reis, Jarbas Dantas, José Nilton (*in memoriam*), Marcos Melo, Maria

de Fátima, Rafael Barbosa, Renato Augusto e Válter Augusto por terem segurado as pontas enquanto eu estive fora. Também a Alan Cézar, Alexandre Campelo, Anastáscia Brandão, Ana Paula Gomes, Elton França, Eleonora Barbosa, FJ, Kleyton Siqueira e Mariany Brito.

Aos colegas Marcelo Correia, Luany Marciano e Denise Stéphanie pelas contribuições diretas dadas para esse trabalho. E também a seu Lulinha por ter sido o apoio no experimento.

Ao programa de Biometria e Estatística Aplicada por ter me dado uma motivação imensa para voltar a estudar. Aos colegas cuja boa convivência isso não seria possível: Alexandre Gramosa, Augusto César, Caio César, Eduardo Silva, Edvaldo Nunes, Elielma Santana, Felipe Fernando, Felipe Gusmão, Gabriela Isabel, Henrique Santos, Jucarlos Rufino, João Filgueira, João Rocha, Lucas Amaral, Luciano Serafim, Marciele Lima, Marília Miranda, Natália Morais, Rubens Vivaldi e tantos outros que fizeram o ambiente na sala de estudos ser aconchegante, receptivo e que rendeu diversos momentos engraçados.

Aos meus amigos Dr. Leon Denis da Silva e Dr. José Deibsom da Silva, que acompanharam toda a minha trajetória acadêmica e torceram demais por mim. Fica minhas sinceras desculpas por não termos comemorado essa conquista como merecíamos.

As filhas de Seu Dudu (*in memorian*) e dona Alderita *in memoriam*, Aldecira, Aurecir, Aurenice (*in memoriam*), Auriceia, Aurineia e Auristeia, minhas tias.

A minha tia/madrinha/mãe Queu (*in memoriam*), meu tio Paulo, meus primos Renata e Rogério (*in memoriam*) por terem acreditado mais que muitos.

Aos meus amigos de longíssima data: Adeir, Adriana, Arystton, Cláudio, David, Dayse, Diego Martins, Diego *Gago*, Edith, Edson, Eli, Everthon, Heyder, Izaak, Jadsom, Jefferson, Jorge, Laís, Leandro (*in memoriam*), Mário, Priscila, Rafaela, Renata, Ricardo (*in memoriam*), Thiago, Vanessa, Walfrido. São quase 30 anos. E serão mais.

Aos membros do Escrombo da Babilônia: Bokette, Dalton, Eduarda, Elton, Felipe, Jorge, Laerte, Leilane e Tamires. Obrigado por tudo. Mesmo.

Ao Instituto MR, Instituto João Loque Brasil, Instituto Liromont e Top Capital e os demais amigos de bolso do Umbral (R.I.P.).

Tenho sangrado demais,
tenho chorado pra cachorro,
Ano passado eu morri,
mas esse ano eu não morro.
Belchior

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Com a diminuição da área cultivável no mundo há uma demanda crescente para otimizar o consumo de recursos hídricos. A agricultura de precisão pode utilizar estações de baixo custo para satisfazer essa demanda. Este trabalho teve como objetivo a construção de uma estação que monitora a temperatura, a umidade relativa do ar e umidade do solo utilizando seis higrômetros e armazena as informações em um cartão *microSD*. A estação funciona com energia da rede elétrica disponível, mas utiliza baterias e uma placa solar como forma de energia reserva. O experimento foi conduzido na Universidade Federal Rural de Pernambuco, em Recife-PE por um período de oito dias, com os higrômetros inseridos em vasos plantados com capim-Buffel e as informações coletadas foram comparadas com as métricas da estação meteorológica Aeroporto e da estação pluviométrica Dois Irmãos, ambas em Recife-PE. Para comparar os resultados foram utilizados os testes não-paramétricos de Mann-Whitney, Kruskal-Wallis e teste de Dunn. Os valores de temperatura e umidade do ar tiveram comportamento similar às informações da estação Aeroporto, embora tiveram uma variação maior. Comparando as medidas no período observado com as outras duas estações, foi possível detectar períodos chuvosos. As informações de umidade do solo foram coletadas utilizando os seis higrômetros em um único vaso e submetido a uma rega. O teste de Kruskal-Wallis teve um valor p de $5{,}12 \times 10^{-6}$ indicando diferença entre as amostras. Aplicando o teste de Dunn, pode-se identificar semelhanças nas medidas de três pares de higrômetros. Em outro teste, os higrômetros foram colocados em grupos de três em quatro vasos com umidade de solo em 20%, 35%, 50% e 60% e seis medidas foram feitas para cada higrômetro em cada pote. Os valores obtidos foram ajustados a um modelo linear com $R^2 = 0{,}9956$, mostrando que o higrômetro é bastante pode ser utilizado nessa configuração. A estação fun-

cionou por 20 horas sem energia elétrica da rede, o que é uma boa alternativa para pequenos momentos sem energia. A estação pode servir de ponto de partida para uma estação mais robusta e autônoma, mas alguns ajustes visando a eficiência do sistema energético seriam necessários. A utilização de módulos para comunicação das observações pela internet seriam outra excelente adição visando o uso comercial.

Palavras-chave: *DHT11, FC-28, Arduino, Agrometeorologia.*

ABSTRACT

With the decrease in world's cultivable area there is a growing demand to optimize water resources consumption. Precision agriculture can use low-cost stations to satisfy this demand. This work aimed to build a station that monitors temperature, relative air humidity and soil humidity using six hygrometers and stores the information on a microSD card. The station runs on available power grid, but uses batteries and a solar panel as a form of backup energy. The experiment was conducted at the Federal Rural University of Pernambuco, in Recife-PE for a period of eight days, with the hygrometers inserted in pots planted with Buffelgrass and the information collected was compared with the metrics from the Aeroporto meteorological station and the pluviometric station Dois Irmãos, both in Recife-PE. To compare the results, the non-parametric Mann-Whitney, Kruskal-Wallis and Dunn tests were used. The air temperature and humidity values had similar behavior to the information from the Aeroporto station, although they had a greater variation. Comparing the measurements in the observed period with the other two stations, it was possible to detect rainy periods. Soil moisture information was collected using six hygrometers in a single pot and subjected to irrigation. The Kruskal-Wallis test had a p-value of 5.12×10^{-6} indicating a difference between the samples. Applying Dunn's test, similarities can be identified in the measurements of three pairs of hygrometers. In another test, the hygrometers were placed in groups of three into four pots with soil humidity at 20%, 35%, 50% and 60% and six measurements were made for each hygrometer in each pot. The values obtained were adjusted to a linear model with $R^2 = 0.9956$, showing that the hygrometer can be used in this configuration. The station operated for 20 hours without electricity from the grid, which is a good alternative for short moments without power. The station can serve as a starting

point for a more robust and autonomous station, but some adjustments aimed at improving the efficiency of the energy system would be necessary. The use of modules for communicating observations through the internet would be another excellent addition for commercial use.

Keywords: *DHT11, FC-28, Arduino, Agrometeorology.*

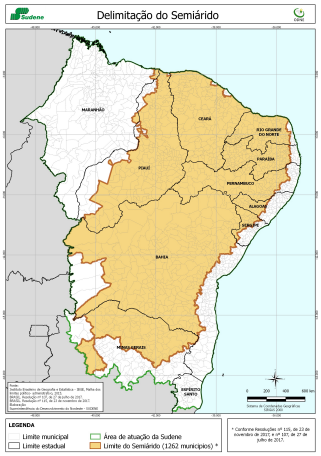
CAPÍTULO 1

INTRODUÇÃO GERAL

1.1. INTRODUÇÃO

Com o risco de aumento de áreas desertificadas no mundo devido às mudanças climáticas, há uma preocupação maior com a escassez de recursos hídricos, principalmente na região Nordeste do Brasil, a região mais árida do país. Segundo o Instituto Brasileiro de Geografia e Estatística - IBGE, a Região Nordeste tem 1.554.291,744 km^2, sendo que 72,24% desse território faz parte do Polígono das Secas como visto em GURGEL (1984), com predominância do clima Semiárido. A Superintendência para o Desenvolvimento do Nordeste-SUDENE, delimita a região Semiárida como no mapa da Figura 1.1.

Figura 1.1: Delimitação do Semiárido



Fonte: SUDENE

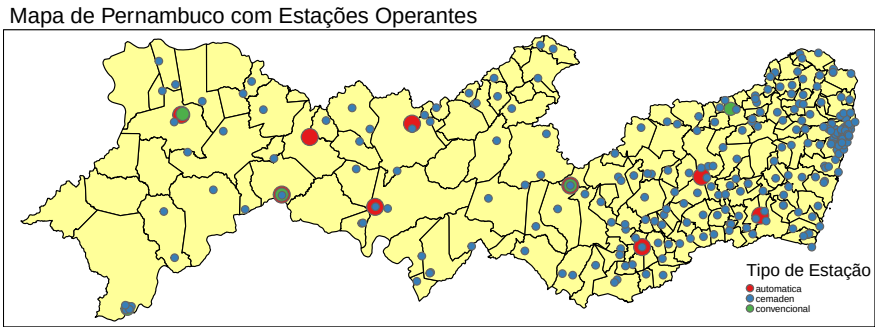
A região é conhecida pela escassez de precipitação e isso influencia na agro-

pecuária como um todo. Saber o comportamento anual de grandezas como temperatura, umidade do ar e precipitação numa região pode ser de grande ajuda para prever os melhores momentos para iniciar uma cultura ou se prevenir de momentos de seca. Monitorar a umidade do solo também é essencial, pois conhecendo a umidade do solo, o agricultor pode determinar a quantidade de água que uma determinada cultura precisa sem desperdiçar a água de irrigação. Ela pode ser medida de diversas maneiras, como o uso de estufa, sensores no solo e até imagens de satélite como visto em Gutierres e Neves (2021). Outra forma de monitorar diversas variáveis meteorológicas é utilizando estações construídas para esse fim.

Os cientistas brasileiros, produtores e a população em geral podem obter acesso às informações de várias estações cuidadas por diferentes agências e institutos brasileiros como: Instituto Nacional de Meteorologia-INMET, Centro Nacional de Monitoramento e Alertas de Desastres Naturais - CEMADEN, Agência Nacional de Águas - ANA, dentre outros.

No *site* do INMET [1] há um mapa interativo com diversas estações no Brasil de responsabilidade do órgão. No *site* do CEMADEN [2] também há um mapa com as estações de responsabilidade do órgão no Brasil. Nota-se que no estado de Pernambuco há uma grande quantidade de pluviômetros dada a necessidade de monitorar a precipitação no estado. Na Figura 1.2 temos uma organização das estações operantes do INMET e do CEMADEN no Estado de Pernambuco.

Figura 1.2: Algumas estações operantes em Pernambuco



Fonte: Autoria própria

O INSTITUTO NACIONAL DE METEOROLOGIA (1999) é um cartilha com

[1] https://mapas.inmet.gov.br/
[2] https://mapainterativo.cemaden.gov.br/

diversos parâmetros e informações sobre as estações meteorológicas, dentre elas há as Estações Automáticas (AT) que têm a característica de registrar as observações feitas automaticamente e enviar esses dados para o órgão responsável. As estações do CEMADEN, mais numerosas, monitoram a precipitação nos diversos locais.

Com essa necessidade de monitoramento de grandezas físicas como umidade do solo, temperatura e umidade do ar em plantações e também pensando nos valores normalmente altos que sistemas de monitoramento no mercado possuem, pode-se utilizar a opção de construção de estações de baixo custo utilizando a plataforma de Arduino^{tm}, que além de ser relativamente barata, têm a característica de ter código-aberto, facilitando a replicação e a versatilidade nas aplicações, já que a quantidade de sensores e módulos a ser utilizada só é limitada pelas características das placas.

Há variados artigos na literatura que mostram a construção de sistemas que podem ser úteis tanto na vida do agricultor como na do pesquisador das Ciências Agrárias. Essas aplicações da plataforma podem contribuir imensamente para a agricultura de precisão e consequentemente para um bom uso da água, aumento e automatização da produção unindo um bom custo-benefício na hora de utilizar os sistemas.

Em Moura Campos et al. (2021) foi criado um sistema de irrigação autônomo com quatro sensores de umidade do solo (10HS; METER Group, Pullman, WA, USA), um de temperatura e umidade relativa do ar (DHT 22; AM2302; Aosong Electronics, Guangzhou, China) em um microcontrolador Arduino (Arduino Mega 2560 Rev3; Arduino, Ivrea, Italy). Os sensores de umidade atuavam em conjunto com quatro válvulas solenoides (HVF-100; Rain Bird, Azuza, CA, USA) para irrigar uma cultura de tomate-cereja (*Solanum lycopersicum var. cerasiforme*). Os dados de umidade do solo, temperatura do ar e umidade relativa do ar eram salvos em um cartão microSD e enviados para um telefone celular utilizando SMS (*Short Message Service*) por meio de um módulo GSM (GSM GPRS Shield SIM900; EFCom, Ramdaspeth, Nagpur, India). Foram escolhidos quatro limiares de umidade do solo (0,23, 0,30, 0,37 e 0,44 $m^3.m^{-3}$) em quatro potes, que foram monitorados a cada 30 minutos pelo sistema. Sempre que alguma leitura era mais baixa que o respectivo limiar, mais água era adicionada por um minuto, até voltar ao limiar. No estudo relatado no artigo, a umidade do solo se manteve estável nos limiares.

Pensando em aplicação voltada para o *mobile*, Almeida et al. (2019) criaram

um sistema de monitoramento do pH de uma solução de modo que o sistema se conecte pela internet e envie as informações para um banco de dados, que pode ser lido utilizando um aplicativo feito para celulares com sistema operacional *Android*. A comunicação acontecia por causa do módulo wifi ESP8266, que enviava as informações para um servidor na nuvem *Google*^{tm} *Firebase*. Os dados sobre o pH eram gerados utilizando-se um módulo sensor Ph4502c junto com um pH eletrodo BNC. O aplicativo foi feito para ler as informações do servidor na *Firebase* e na sua tela pode-se ler o valor do pH da solução, que levava aproximadamente trinta segundos a um minuto para se estabilizar, bem como a informação se a solução era ácida, neutra ou alcalina. Também havia a previsão no projeto de um relé ser acionado para ativar uma bomba em função de um valor do pH, por exemplo.

Em Bhadani e Vashisht (2019), há uma proposta de estação de monitoramento envolvendo um sensor de umidade do solo FC-28 e um sensor de umidade do ar e temperatura DT11. O sistema foi contruído em uma placa Arduino ATMega328P de modo que os dados foram apresentados em um visor LCD I2C 16x2, que mostra 2 linhas de caracteres que cabem 16 caracteres cada. Não há mais informações sobre a frequência de atualização dos dados e os dados mostrados não são salvos de nenhuma maneira.

Utilizando uma placa fotovoltaica, um sensor DT11, um sensor de temperatura do solo e duas sondas que medem a umidade do solo em profundidades diferentes, Ramadan et al. (2018) construíram uma estação autônoma para medições, com armazenamento das informações em cartão microSD. As sondas são feitas de Policloreto de Vinila (PVC), cuja diferença é que uma usa pares de sensores de cobre com um material isolante e a outra usa anéis de aço como sensores, sendo a primeira a que teve um melhor desempenho.

Baseado no fato que sensores de umidade do solo são caros, González-Teruel et al. (2019) criaram um sensor capacitivo de solo que utiliza das capacidades de leitura de dados de um Arduino AtMega328P. O sistema utiliza o protocolo SDI-12 para comunicação entre o sensor e o dispositivo *datalogger* CR1000, que armazenava os dados a cada 10 minutos através de uma *Virtual Private Network* (VPN). Há um teste comparativo entre o sensor e outros sensores comerciais e há a determinação de uma função para calibrar os resultados. Deste modo, é feito uma regressão entre os dados de umidade observados pelo sensor e a umidade observada utilizando o método

gravimétrico, que consiste em secar a amostra de solo e uma estufa e determinar a variação de peso perdido como umidade. A precisão calculada do sensor foi menor que os sensores comerciais, mas com a considerável redução de preço, mostrou um custo-benefício relevante.

Em Aditia e Ilham (2022), foi construída uma incubadora de ovos de galinha automática utilizando um sensor DHT11 para medir a temperatura e umidade do ar, um visor LCD para visualizar os dados e um módulo RTC para contagem de tempo. Utiliza-se duas lâmpadas incandescentes para manter a temperatura interna e uma ventoinha para controlar a umidade, os dois controlados pelo controlador da placa Arduino Uno. Assim conseguiram uma maneira precisa de manter a temperatura entre 38℃ e 39℃ e a umidade interna entre 52% e 55%, que são considerados valores ideais para obter uma taxa de eclosão maior.

Utilizando um sensor DHT11, uma placa Arduino Uno R3, um visor LCD 16x2 e um módulo Ethernet, Hariyanto, Hendrawan e Ritzkal (2020) utilizaram o programa de mensagens Telegram para monitorar a temperatura em um ambiente fechado. Um esquema com dois leds, um vermelho e um verde, serviram para indicar que a temperatura estava maior ou igual a 30℃ ou menor ou igual a 25℃, respectivamente. O sistema foi conectado com a rede LAN utilizando o módulo Ethernet e fazendo uma comunicação com o número de telefone de Telegram do pesquisador.

Na Índia, com a criação da missão Cidades Inteligentes em 2015, houve a necessidade de utilizar avanços tecnológicos para facilitar a crescente urbanização do subcontinente. Sharmila et al. (2019) testaram um sistema para monitorar a temperatura interna de um cômodo e que atue abrindo ou fechando uma janela. Utiliza uma placa Arduino Uno, um sensor DHT11 e um motor de Passo, que recebe sinais de corrente contínua e rotaciona um eixo com ação eletromagnética. Foi utilizado um motor de Passo porque ele permite maior precisão na rotação, já que possui um número de passos fixo para uma volta do eixo. Assim o sistema capta a temperatura ambiente e decide se vai abrir ou fechar a janela utilizando a atuação do motor de Passo, assim, automatizando esse processo que pode ser utilizado em ambientes como hospitais, orfanatos e outros prédios inteligentes.

Um dos componentes importantes para o monitoramento de variáveis ambientais é como o usuário ou pesquisador vai observar os dados. Novelan e Amin (2020) criaram um sistema de monitoramento de temperatura e umidade do ar com o au-

xílio de um microcontrolador NodeMCU e um sensor DHT11. A placa NodeMCU é bastante utilizada em aplicações de internet das coisas (*Internet of Things* - IoT) pela sua conectividade com a internet por rede Wi-Fi. Assim, a placa se conecta a um servidor na *Google*^{tm} *Firebase* e envia as leituras para lá. Com uma aplicação criada com a linguagem VisualBasic.net, as leituras podem ser acessadas de qualquer lugar do mundo.

Em Debele e Qian (2020), foi criado um protótipo que monitora e regula a temperatura de um ambiente utilizando uma placa Arduino Uno, um ventilador e uma lâmpada como forma de aquecimento. A temperatura de referência é definida utilizando um teclado ligado à placa. Quando a temperatura fica baixa, a lâmpada é ligada. Quando a temperatura fica alta, o ventilador é ligado. A velocidade de rotação do ventilador vai depender da diferença da temperatura do ambiente e da temepratura de referência, com uma temperatura maior influenciando num ciclo de velocidade maior.

Numa aplicação mais complexa do Arduino, Bitella et al. (2014) organizaram um sistema de monitoramento com múltiplos sensores: O sensor de temperatura do ar e umidade relativa do ar DHT22, sondas dielétricas Vegetronix VH400 para funcionar como sensor de umidade do solo, como sensor de temperatura do solo foi utilizado um termômetro digital Dallas DS18B20, um termômetro infravermelho MLX90614ESF-BAA-00TU-ND para ser utilizado para monitorar a temperatura do dossel da vegetação. Junto a isso, foi utilizado um módulo para gravar dados em um cartão SD, um módulo GSM para enviar os dados pela internet utilizando a rede de celular e um display LCD. Houve um teste para calibração das sondas dielétricas utilizando três tipos de solos: Franco-Arenoso, Franco-Argiloso e Arenoso. Conseguiu uma equação de calibração com comportamento sigmoidal para calcular a umidade em função da tensão medida pelas sondas.

Embora fugindo um pouco da aplicação direta com Arduino, Roux et al. (2018) mostraram a criação de um sensor de umidade do solo e salinidade do solo utilizando um tubo de PVC e cobre em formato de hélices ao redor do tubo. Esse sensor é fisicamente um capacitor e portanto há um processo de calibração da umidade do solo e a capacitância medida utilizando um instrumento. A partir daí, faz testes para detecção de água em um pomar comparando com um sensor comercial. A sua conclusão é que o seu capacitor tem um custo menor e uma funcionalidade à mais

que os sensores comerciais.

Pensando em analisar o higrômetro resistivo FC-28 para medir umidade do solo, Mendes et al. (2021) fizeram um experimento para medir o sinal de saída do sensor inserido em substrato comercial Bioplant que sofreu estresse hídrico. O substrato após secagem em estufa foi dividido em 5 amostras colocadas em um recipiente de PVC com uma tela para drenagem, foram encharcadas com água destilada e após isso tiveram a variação de peso medida em balança para obter a umidade gravimétrica (massa da água/massa do solo seco) utilizando o sensor a cada medida. O sensor FC-28 tem como saída valores entre 0-1023 com o 0 sendo um valor com maior umidade e 1023 um valor com menor umidade. Por fim, obteve a umidade volumétrica (densidade da água/densidade do solo) para utilizar como variável independente de um modelo polinomial de grau 3 com um ótimo ajuste.

Utilizando um microcontrolador diferente, Gutiérrez et al. (2013) criaram um sistema de irrigação automatizado conectado com a internet que utiliza sensores de temperatura e umidade do solo e atuadores para determinar quando deve-se iniciar a irrigação. Foi criada uma rede de unidades de sensores sem-fio (do inglês, *Wireless Sensor Units*-WSU) que se conectam utilizando o protocolo de rádio-frequência Zigbee a uma unidade de informação sem-fio (do inglês, *Wireless Information Unit*-WIU). Em cada WSU há uma placa fotovoltaica MPT4.8-75, 3 baterias recarregáveis, um microcontrolador PIC24FJ64GB004, um *transceiver* XBee Pro S2, um sensor de temperatura VH4000 e um sensor de temperatura do solo DS1822. A WIU possui o mesmo microcontrolador e *transceiver* das WSUs, com adição de um módulo GPRS, um painel solar KC130TM, bateria L-24M/DC-140 e duas bombas que servem como atuadoras para liberar a água da irrigação conectada a um tanque de 5000 litros. Foram idealizados 4 tipos de irrigação no algoritmo da WIU: Manual com tempo de duração fixo; Data e tempos agendados utilizando uma aplicação *web*; Automatizada com duração fixa acionada se a umidade medida pela WSU for menor que a de limiar; Automatizada com duração fixa acionada se a temperatura medida pela WSU ficar maior que a temperatura de limiar. O sistema foi utilizado em uma produção de sálvia *(Salvia officinalis)* em San Jose del Cabo, Baja California, México. Comparado com o método comum de gotejamento livre utilizado pelos produtores, a irrigação automatizada economizou 90% de água para uma redução de 20% da produção. Utilizado em outros locais, a economia de água das irrigações

automatizadas economizaram 60% de água em relação às irrigações programadas. Como o sistema foi projetado com momentos de desligamento automático, as placas fotovoltaicas foram suficientes para manter o sistema funcionando.

Com o pensamento voltado para um sistema mais complexo e autônomo energeticamente, López-Vargas et al. (2019) estabeleceram um protótipo bem eficiente e de baixo custo na Espanha. Esse protótipo é alimentado por uma placa fotovoltaica e uma bateria de 12V, é controlado por uma placa Arduino Uno e monitora diversos medidas ambientais: irradiância, temperatura, umidade do ar, velocidade horizontal do vento e precipitação. Com foco em produzir industrialmente o sistema, houve necessidade de seguir parâmetros da Comissão Eletrotécnica Internacional e assim alguns componentes foram utilizados para medir algumas grandezas do próprio sistema, como tensão, corrente, temperatura, etc. Os principais sensores utilizados foram um sensor DHT22 para temperatura e umidade do ar, sensores de temperatura DS18B20 para medir a temperatura da placa fotovoltaica, células fotovoltaicas calibradas para medir a irradiância, um anemômetro para velocidade do vento e um sensor de chuva Rain-O-Matic para medir a precipitação. Todas as informações foram salvas em um cartão SD para análise posterior e comparadas com as medidas realizadas por um sistema comercial Agilent e foi considerado bastante eficiente utilizando-se da estatística de Raíz do Erro Quadrático Médio (REQM). Além disso, o sistema foi configurado para ciclos de baixa energia para fins de eficiência energética.

1.2. OBJETIVOS

Dadas as diversas necessidades de monitoramento de variáveis ambientais no campo, essa tese teve os seguintes objetivos:

1.2.1. Geral

Analisar a viabilidade de uma estação experimental de baixo custo, que registre informações de temperatura, umidade relativa do ar e umidade do solo com suporte de energia.

1.2.2. Específicos

- Construir estação experimental utilizando Arduino.
- Executar testes de funcionamento dos sensores.
- Realizar análises estatísticas de validação das leituras dos sensores.
- Testar autonomia energética da estação.

CAPÍTULO 2

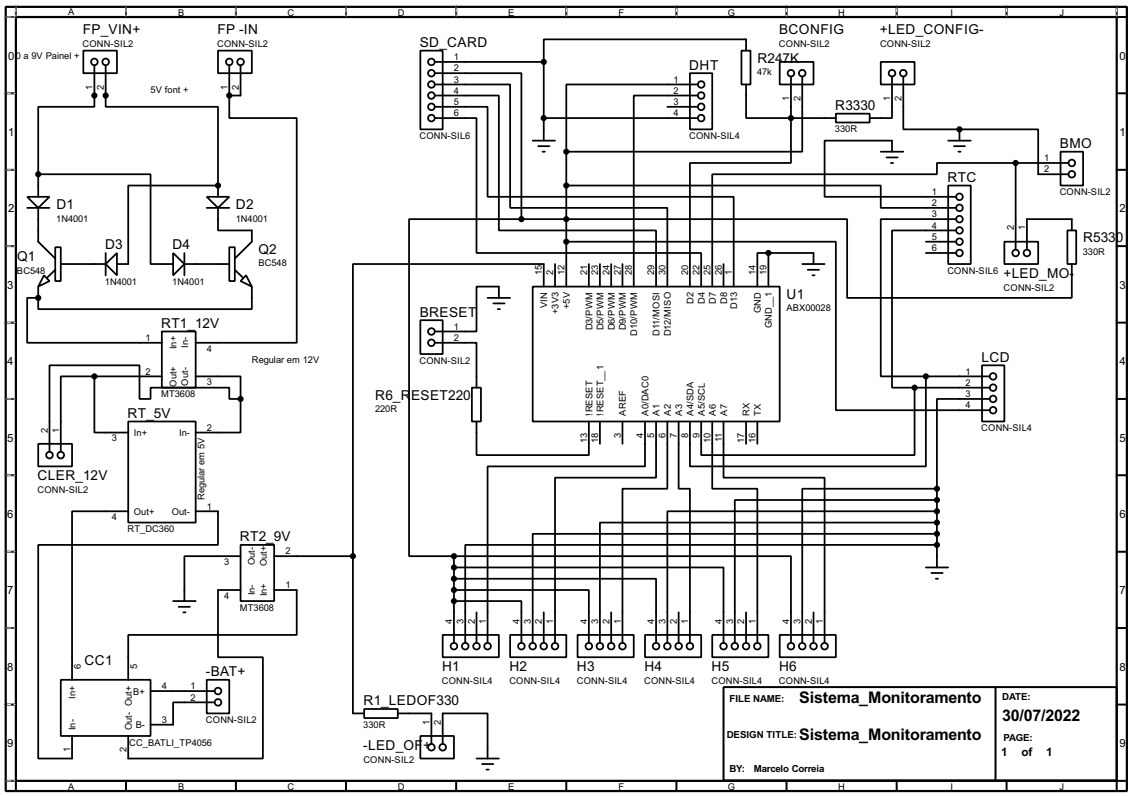
MATERIAL E MÉTODOS

Nesse capítulo será relatado o processo de construção da estação, detalhando os softwares utilizados, o esquema elétrico da placa, os componentes elétricos, os testes realizados e os testes estatísticos executados.

2.1. SOFTWARES

Foi utilizado o Proteus Design Suite 9.14 (LABCENTER ELECTRONICS, 1988), para projetar a placa com a disposição dos componentes e gerar o arquivo que será utilizado para imprimir a placa de circuito impresso. O esquema eletrônico do sistema na Figura 2.1 especifica a disposição dos componentes eletrônicos na placa de circuito impresso. A Figura 2.2a mostra a frente da placa e a Figura 2.2b mostra o verso da placa com a trilha, que depois de impressa vai ser transferida como decalque para uma placa de fenolite e após isso corroída em uma solução de percloreto de ferro até ficar como na Figura 2.2c.

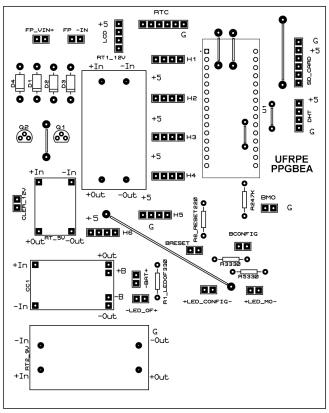
Figura 2.1: Esquema eletrônico do sistema

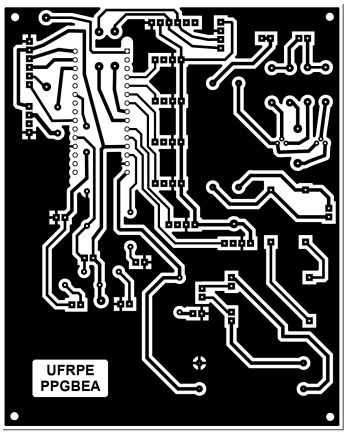


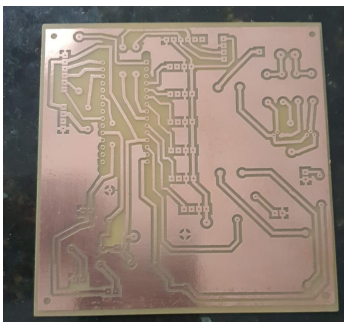
Fonte: Autoria própria

Figura 2.2: Desenhos da placa de circuito impresso

(a) Frente da placa (b) Verso da placa (c) Trilha da placa







Fonte: Autoria própria

O Arduino IDE 1.8.19 (THE ARDUINO TEAM, 2005) é um software utilizado para programar os algoritmos listados no Apêndice A que vão informar ao micro-

controlador como lidar com os sensores e as informaçõss geradas por ele. É utilizada uma linguagem de alto nível baseada nas linguagens C/C++.

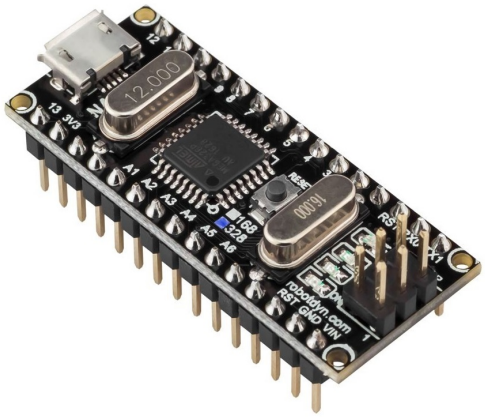
O software R 4.05 (R CORE TEAM, 2022) é uma linguagem de programação com foco na estatística utilizada no mundo todo. Foi utilizado para tabulação das informações geradas pela estação, análises e comparação com as medidas das outras estações e geração dos gráficos dos dados gerados utilizando os códigos do Apêndice C.

2.2. COMPONENTES ELETRÔNICOS

Os principais componentes utilizados foram:

- Arduino Nano V3 Atmega 328 (ver Figura 2.3) que tem função de microcontrolador e recebe o código do Apêndice A.1 para executar sua função.

Figura 2.3: Arduivo V3 Atmega

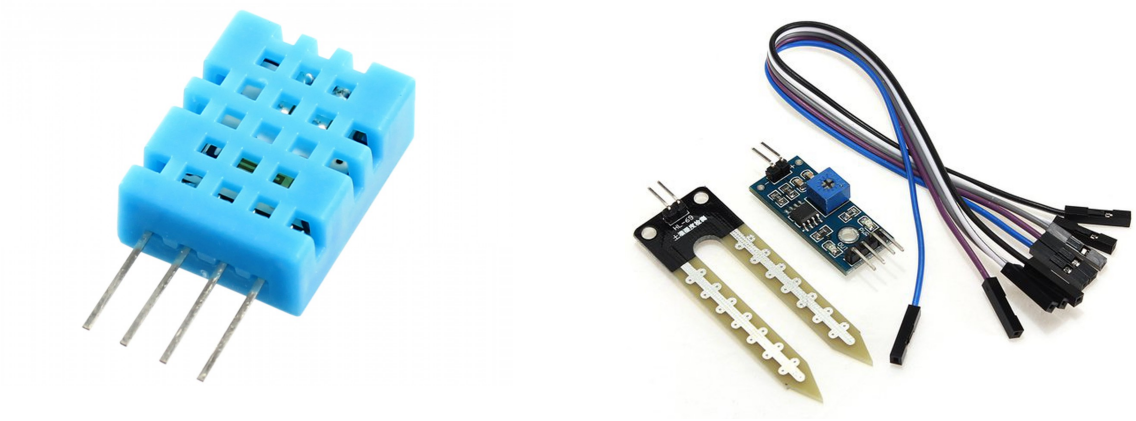


Fonte: (ROBÓTICA, 2024)

- Sensor de temperatura e umidade relativa do ar DHT11 (ver Figura 2.4a). A temperatura é retornada em grau Celsius e a umidade relativa em percentual.
- Sensores de umidade do solo (higrômetros) do tipo FC-28 (ver Figura 2.4b). A umidade é retornada em valores de 0 a 1023, sendo 1023 o maior valor possível de umidade.
- Painel solar CDR02 9V (ver Figura 2.5a), para ajudar na alimentação da estação e dar um pouco de autonomia.

Figura 2.4: Sensores usados na estação

(a) Sensor DHT11 (b) Módulo Higrômetro FC-28



Fonte: (ROBÓTICA, 2024)

- Baterias de lítio 18650 3.7v (ver Figura 2.5b), para armazenar energia para que a estação funcione à noite.

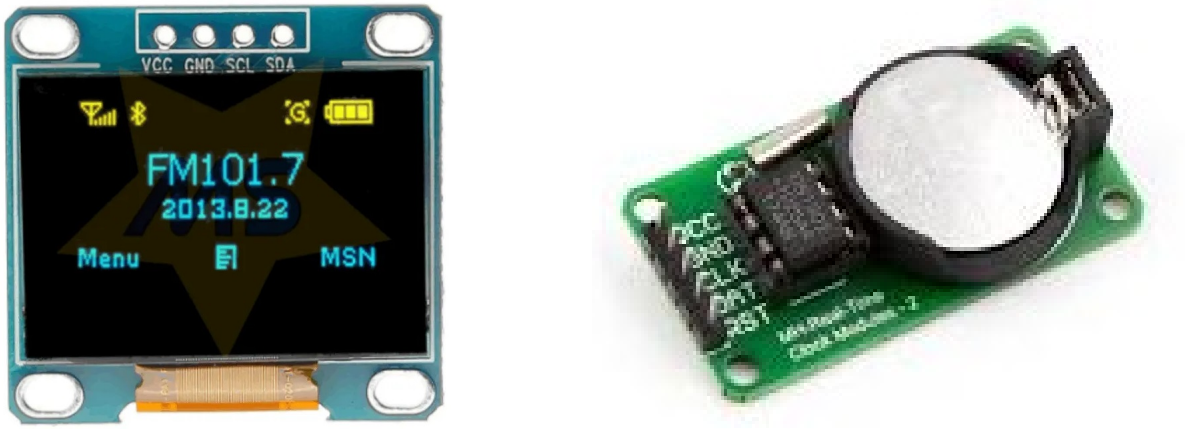
Figura 2.5: Componentes de fornecimento de energia

(a) Placa Solar CDR02 (b) Baterias de Lítio 18650

Fonte: (ROBÓTICA, 2024)

- Módulo RTC DS1302 (ver Figura 2.6b), para utilizar data e hora nos dados.
- Display OLED 128x64 0.96 I2C (ver Figura 2.6a), para mostrar informações da leitura do sistema e dos sensores.
- Diodo 1N5819 Schottky (ver Figura B.1a), para ligação segura da placa solar.
- Regulador de tensão Xl6009 conversor DC-DC *Step Up and Down* (ver Figura

Figura 2.6: Display OLED e Módulo RTC
(a) Display OLED I2C (b) Módulo Real Time Clock - RTC



Fonte: (ROBÓTICA, 2024)

B.2a), que serve para converter uma tensão mais baixa na sua entrada para uma tensão mais alta na saída.

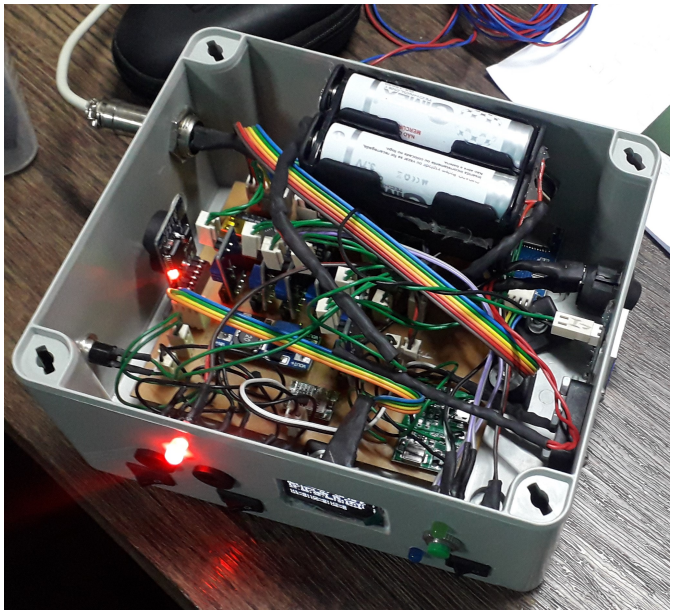
- Módulo regulador de tensão *Step Up* DC-DC e carregador de bateria 18650 (ver Figura B.2b), auxilia como regulador de tensão na saída para alimentar o Arduino, garantindo a estabilidade da tensão combinada da fonte e bateria.
- Botão *push* do tipo pulsador de 12mm (ver Figura B.3a), utilizado para trocar as opções de registro de tempo.
- Mini *Cooler* de 30x30x10mm (ver Figura B.3a) para resfriar os componentes internamente e evitar falhas no funcionamento por causa de sobreaquecimento.
- LEDs diversos para indicar funcionamento.
- Módulo leitor de cartão *microSD* (ver Figura B.1b), para poder conectar o cartão e salvar os dados em um arquivo.
- Resistores e cabos diversos.

2.3. CONSTRUÇÃO DA ESTAÇÃO

A estação foi construída em uma caixa de plástico, com um botão de ligar e desligar, uma entrada para cartão microSD, um botão para acionar a gravação no cartão microSD, 6 saídas para cada higrômetro, com uma distância de aproximadamente de 8 metros para cada higrômetro e uma saída para o sensor DHT11. Há

também entradas para fontes de 5 volts ou 9 volts. A Figura 2.7 mostra a caixa aberta.

Figura 2.7: Caixa com o circuito do sistema



Fonte: Autoria própria

A caixa foi montada e ligada na tomada utilizando uma fonte de 9 volts dentro de uma sala perto do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco, campus Sede, no bairro de Dois Irmãos, latitude -8,015972 e longitude -34,95055. Foi escolhido um local para não deixar exposta para pessoas que transitassem no local (ver Figura 2.8). Desse modo, a placa fotovoltaica foi colocada na telha desse cômodo (ver Figura 2.9) e o sensor DHT11 colocado sob a sombra da telha (ver Figuras 2.10a e 2.10b), com os cabos entrando por um buraco no teto. O mesmo buraco permite a passagem dos cabos dos sensores de umidade do solo que foram colocados nos vasos (ver Figuras 2.11a e 2.11b). Foi feito um único teste para atestar o tempo que a estação funcionaria sem energia da tomada, e a estação funcionou por 20 horas, inciando-se às 12:33 do dia 16-12-2021 e parando 8:22 do dia 17-12-2021.

Figura 2.8: Caixa Instalada

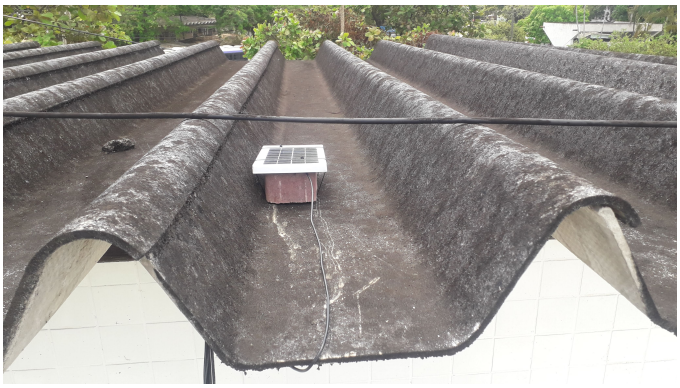
(a) Visão frontal

(b) Visão lateral

(c) Caixa na prateleira

Fonte: Autoria própria

Figura 2.9: Placa fotovoltaica



Fonte: Autoria própria

Figura 2.10: Posição do sensor DHT11

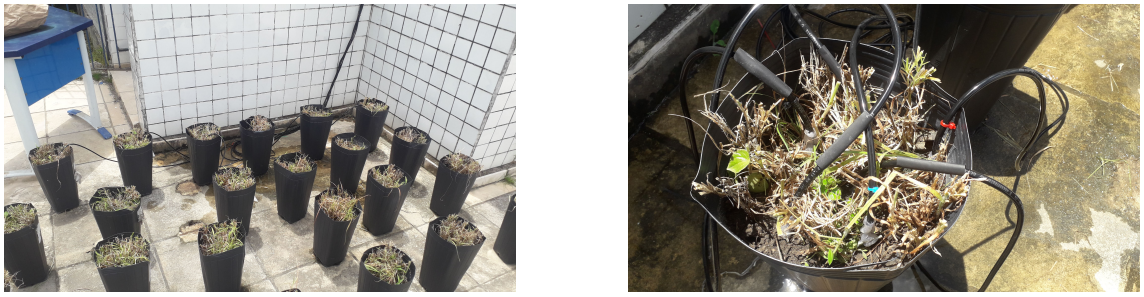
(a) Sensor DHT11 embaixo da telha

(b) Amplificação e detalhamento da imagem

Fonte: Autoria própria

Figura 2.11: Sensores de umidade

(a) Sensores nos vasos (b) Sensores em um único vaso



Fonte: Autoria própria

2.4. TESTES REALIZADOS

Havia no local um experimento conduzido por outra estudante e os vasos foram utilizados para testar a estação. Trinta vasos com capim-buffel (*Cenchrus ciliaris*) que eram irrigados diariamente com água pura. Os vasos tinham volume total de 7 dm^3 e aproximadamente 6 dm^3 de solo classificado como Cambissolo Háplico, de acordo com a classificação da Embrapa Solos (2006).

O solo era proveniente da área experimental da Universidade Federal Rural de Pernambuco (UFRPE), Unidade Acadêmica de Serra Talhada (UAST). A unidade é localizada sob as coordenadas geográficas latitude -7,954745 e longitude -38,295225, com altitude aproximada de 500 m, situada no município de Serra Talhada, Pernambuco. Coletado na profundidade de 0-20 cm do perfil, destorroado, submetido ao revolvimento para secagem ao ar, homogeneizado e acondicionado em vasos plásticos. A Figura 2.12 mostra a disposição dos vasos.

Foram feitos os seguintes testes de funcionamento da estação:

1. Um teste de aproximadamente 12 minutos com os 6 higrômetros em um mesmo pote com o uso de uma rega de 400 ml no pote. A estação mediu a umidade do solo a cada 5 segundos.
2. Um teste onde a estação funcionou entre 25/11/2021 a 17/12/2021 para medir temperatura e umidade do ar a cada minuto.
3. Um teste onde foram utilizados 4 copos de 500 cm^3 do mesmo solo do experimento que tiveram suas umidades do solo acertadas para 20%, 35%, 50% e 60% utilizando diferentes quantidades de águas e aferidas por um higrômetro

Figura 2.12: Ambiente do teste

Fonte: Autoria própria

analógico do tipo Minipa MV-331 (MINIPA, 2024), mostrado na Figura 2.13. A partir disso, os 6 higrômetros foram divididos em 2 grupos com 3 higrômetros cada, e cada grupo passou por cada pote fazendo 6 medições, cada uma em um minuto.

Figura 2.13: Medidor de Umidade do Solo MV-331

(a) Medidor (b) Leitor analógico

Fonte: Autoria própria

As leituras foram utilizadas para comparar os dados com outras estações da região e fazer testes estatísticos adequados para comparação dos dados.

2.5. TABULAÇÃO DOS DADOS

2.5.1. Dados de fontes externas

Para fins comparativos, foram utilizados os dados da estação Aeroporto (82899), código METAR=SBRF, localizada no Aeroporto Internacional do Recife Guararapes - Gilberto Freyre, em Recife, latitude -8,127281, longitude -34,921764, com uma distância aproximada de 12 km do local dos testes, disponíveis em RP5 (2022). A estação é de responsabilidade do Departamento de Controle de Espaço Aéreo-DECEA da Força Aérea Brasileira. Foram escolhidos os dados retirados desse site como referência porque os dados disponíveis no site do INMET para a referida estação mostravam medidas a cada 6 horas, ao invés de medidas a cada hora.

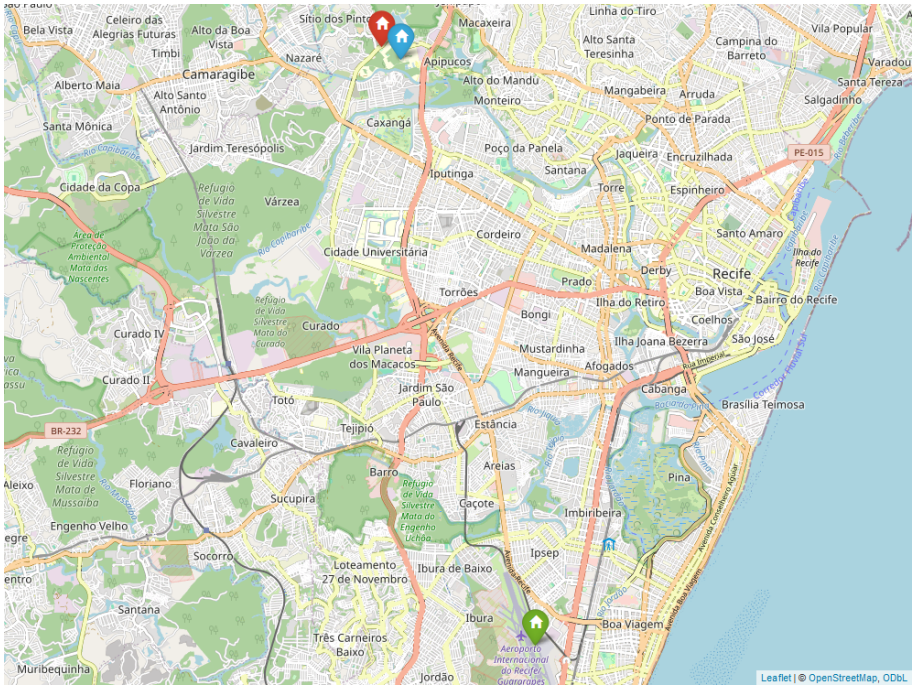
Os dados obtidos vêm com as informações de data e hora local, temperatura em graus Celsius a 2 metros de altitude, pressão atmosférica ao nível da estação em milímetros de mercúrio, pressão atmosférica ao nível do mar, umidade relativa do ar em porcentagem na altitude de 2 metros, direção do vento em uma altitude de 10-12 metros da superfície terrestre, velocidade do vento em metros por segundo, valor máximo de rajada do vento na altitude de 10-12 metros, fenômenos especiais observados no aeroporto ou próximo dele, fenômenos de último tempo, nebulosidade geral, distância de visibilidade horizontal e temperatura de condensação a uma altura de 2 metros. Serão utilizadas a temperatura, a umidade do ar e os fenômenos especiais observados perto do aeroporto.

Também foram utilizados os dados da estação pluviométrica Dois Irmãos do CEMADEN, código 261160603A, localizada na Universidade Federal Rural de Pernambuco, em Recife, latitude: -8,01838, longitude -34,94706, com uma distância aproximada de 500 metros do local de testes, disponíveis em CEMADEN (2024). Essa estação foi escolhida por ser a mais próxima da estação instalada.

O conjunto de dados do CEMADEN é disponibilizado por mês e vêm com informações de mais de uma estação do município, como o código, o estado, o nome da estação, a latitude e longitude da estação. A estação registra dados de precipitação com a data e a hora. A informação é dada em milímetros nos últimos 10 minutos. A posição das estações pode ser visualizada na Figura 2.14, com o alfinete vermelho representando a estação UFRPE, o azul a estação Dois Irmãos e

a verde a estação AEROPORTO.

Figura 2.14: Localização das estações em Recife-PE



Fonte: Autoria própria

2.5.2. Geração de dados pela estação

O código principal carregado na memória da placa Arduino está localizado no Apêndice A.1. Uma maneira de resumir o código é a seguinte:

1. Início
   - O Arduino inicializa e executa a função `setup()` uma vez.
2. Setup
   - Inicialização:
     - Inicializa a comunicação serial.
     - Inicializa o RTC e o LCD.
   - Configuração de Tempo:
     - Lê a configuração de tempo de leitura dos sensores armazenada na Electrically-Erasable Programmable Read-Only Memory-EEPROM.

- Configuração de Pinos:
  - Configura os pinos dos botões e sensores como entrada ou saída.
- Verificação do Cartão SD:
  - Verifica se o cartão SD está presente e funcionando.
  - Exibe mensagens de falha ou sucesso no LCD.

3. Loop
   - Atualização de Tempo:
     - Atualiza a hora atual do RTC.
   - Configuração de Leitura:
     - Verifica o estado do botão de configuração e ajusta o tempo de leitura dos sensores.
   - Leitura de Sensores de Solo:
     - Calcula a média das leituras dos sensores de solo.
   - Leitura do Sensor DHT:
     - Lê os dados de temperatura e umidade do sensor DHT.
   - Gravação de Dados:
     - Grava dados no cartão SD ou exibe informações na serial, dependendo do modo de operação.
   - Atualização de Modo de Operação:
     - Verifica se o modo de operação mudou e atualiza as informações no LCD ou serial.

A estação gera um arquivo *MyData.csv* com as informações ambientais organizadas por horário de medição. Uma característica importante da estação é que pode-se configurar a leitura entre 5 espaços de tempo pré-definidos: a cada 5 segundos, a cada minuto, a cada 30 minutos, a cada hora e a cada seis horas. Essas informações podem ser visualizadas no *display* LCD em tempo real.

O arquivo é composto por 17 colunas, organizadas na seguinte ordem: Data, Hora, Temperatura do Ar, Umidade relativa do ar, Umidade do higrômetro 1, Umidade do higrômetro 2, Umidade do higrômetro 3, Umidade do higrômetro 4, Umidade do higrômetro 5, Umidade do higrômetro 6, Umidade relativa do higrômetro

Tabela 2.1: Primeiras linhas do arquivo MyData

data	hora	temp	ar	1	2	3	4	5	6	1rel	2rel	3rel	4rel	5rel	6rel
24/11/2021	11:43:41	33	52	420	418	304	277	328	363	58.94	59.14	70.28	72.92	67.94	64.52
24/11/2021	11:44:41	33	52	419	420	305	276	328	400	59.04	58.94	70.19	73.02	67.94	60.9
24/11/2021	11:45:41	34	57	417	419	305	288	330	403	59.24	59.04	70.19	71.85	67.74	60.61
24/11/2021	11:46:41	32	56	412	417	303	278	159	404	59.73	59.24	70.38	72.83	84.46	60.51
24/11/2021	11:47:41	32	54	416	418	301	276	176	393	59.34	59.14	70.58	73.02	82.8	61.58
24/11/2021	11:48:41	32	52	412	418	303	280	149	396	59.73	59.14	70.38	72.63	85.43	61.29

Fonte: Autoria própria

1, Umidade relativa do higrômetro 2, Umidade relativa do higrômetro 3, Umidade relativa do higrômetro 4, Umidade relativa do higrômetro 5, Umidade relativa do higrômetro 6. Na Tabela 2.1 mostramos as primeiras linhas do arquivo MyData gerado depois de carregado no R e ter as colunas renomeadas.

Posteriormente foi organizado o *dataframe* utilizando a função *Melt* do pacote Reshape2 (WICKHAM, 2007) de modo que ele fique com as colunas *data*, *hora*, *medida* e *valor*. A coluna *medida* vai ter como entrada os nomes dos sensores (temperatura do ar, umidade relativa, Umidade do Higrômetro 1 a 6, Umidade relativa do higrômetro 1 a 6), e *valor* terá o valor de cada medida correspondente e cada período de tempo. É organizada a coluna data para que fique no formato ISO 8601 (ou seja, ANO-MÊS-DIA) utilizando a função *parse_ date_ time* do pacote Lubridate (GROLEMUND; WICKHAM, 2011) e depois separar um *dataframe* para cada medida, de modo que fique mais organizado e simples de se analisar. E por fim, os *dataframes* foram organizados de modo que todas valores dentro de uma hora sejam simplificados para a média desses valores. Por exemplo, a temperatura medida à hora 13 do dia 29/11/21 represente a média das temperaturas medidas entre os minutos 13:00 e 13:59 do mesmo dia. O código em R apresentado no Apêndice C.1 mostra como esses dados foram organizados.

2.6. TESTES ESTATÍSTICOS

2.6.1. Teste de Shapiro-Wilk

O teste de Shapiro-Wilk, visto em Shapiro e Wilk (1965) é um teste estatístico utilizado em análises de dados para verificar se uma amostra de dados é proveniente de uma distribuição normal. É uma das opções mais populares para nesse sen-

tido devido à sua boa capacidade de detectar desvios da normalidade em diferentes tamanhos de amostra.

As hipóteses do teste são:

$$\begin{cases} H_0: & \text{A amostra provém de uma população normal} \\ H_\alpha: & \text{A amostra não provém de uma população normal} \end{cases}$$

A estatística de teste W é calculada utilizando a seguinte fórmula:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Onde a_i são os coeficientes de ponderação, que são determinados com base na média e na variância da distribuição normal para o tamanho da amostra. A distribuição normal padrão é utilizada para calcular esses coeficientes. $x_{(i)}$ representa o i-ésimo valor ordenado da amostra, e $\bar{x}$ é a média da amostra.

Em Fontelles (2012) vemos que caso o W calculado seja menor que o valor crítico $W_{(\alpha,n}$, onde α é a significância e n o tamanho da amostra, temos que rejeitamos a hipótese H_0 de que os dados não seguem uma distribuição normal.

2.6.2. Teste de Mann-Whitney

O teste de Mann-Whitney, visto em Mann e Whitney (1947), é um teste não paramétrico utilizado para comparar duas amostras independentes. Ele é utilizado quando as suposições para a aplicação do teste t de Student não são satisfeitas, como quando as amostras não são normalmente distribuídas ou quando a variância é diferente entre as amostras. O teste é usado para avaliar se as médias de duas amostras independentes são estatisticamente diferentes. As hipóteses do teste são:

$$\begin{cases} H_0: & \mu_0 = \mu_1 \\ H_\alpha: & \mu_0 \neq \mu_1 \end{cases}$$

O teste é realizado da seguinte forma:

1. As amostras são combinadas e ordenadas em ordem crescente.

2. Os valores são então classificados em ordem, atribuindo um ranking de acordo com sua posição na ordem classificada. Em caso de empate, calcula-se a média dos postos ocupados pelos dois ou mais números.
3. A soma dos postos da primeira amostra é calculada, denotada por R_1.
4. A soma dos postos da segunda amostra é calculada, denotada por R_2.

O teste é então realizado comparando a soma dos postos entre as duas amostras, com base na hipótese nula de que as médias das duas amostras são iguais. A estatística de teste U é calculada pela seguinte fórmula:

$$U_1 = \frac{n_1(n_1+1)}{2} - R_1$$

ou,

$$U_2 = \frac{n_2(n_2+1)}{2} - R_2$$

Onde n_1 é o tamanho da primeira amostra e n_2 o tamanho da segunda. A distribuição exata de U é complicada, mas pode ser aproximada por uma distribuição normal para amostras grandes.

2.6.3. Teste de Kruskal-Wallis

O teste de Kruskal-Walli, visto em Kruskal e Wallis (1952) é um teste não paramétrico utilizado para comparar três ou mais grupos independentes de dados. Ele é utilizado quando as suposições para a aplicação da ANOVA não são satisfeitas.

A hipótese nula H_0 do teste é que não há diferença estatisticamente significativa entre as medianas dos grupos, enquanto a hipótese alternativa H_α é que ao menos um dos grupos difere significativamente dos demais. Uma opção para encontrar qual dos grupos difere é aplicar o teste *post hoc* de Dunn, que será visto na seção 2.6.4 na página 40.

O teste é realizado da seguinte forma:

1. As amostras são combinadas e ordenadas em ordem crescente.
2. Os valores são então classificados em ordem, atribuindo um posto de acordo com sua posição na ordem classificada.
3. A soma dos postos de cada grupo é calculada.

4. A estatística de teste H é então calculada com base na hipótese nula de que as medianas dos grupos são estatisticamente iguais.
5. Comparar com a estatística da distribuição $H_{\alpha;n_1;n_2}$, com α sendo a significância e n_i o tamanho das amostras.

A estatística de teste H é dada por:

$$H_{\text{calc}} = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1)$$

Onde N é o tamanho total das amostras, k é o número de grupos, n_i é o tamanho da i-ésima amostra e R_i é a soma dos postos da i-ésima amostra. A distribuição exata de H não é explícita e é computacionalmente difícil de se obter, mas para amostras com $n > 6$ ela se aproxima da qui-quadrado com $k - 1$ graus de liberdade. Em Choi et al. (2003) há uma sugestão de algoritmo para calcular a distribuição exata de H e comparando-a com a qui-quadrado. Em Fontelles (2012) vê-se que caso haja empate nos postos, há a necessidade de ajustar H_{calc} por um fator de correção FC de modo que

$$FC = 1 - \frac{\sum(t^3 - t)}{N^3 - N}$$

sendo t o número de observações empatadas em cada posto. E portanto,

$$H_{\text{ajust}} = \frac{H_{\text{calc}}}{FC}$$

2.6.4. Teste de Dunn

O teste de Dunn, visto em Dunn (1964), é um teste não-paramétrico *post hoc*, ou seja, é utilizado após a execução de outro teste, mais especificamente após descobrir que ao menos uma das médias de uma amostra é diferente das médias de amostras de acordo com o teste de Kruskal-Wallis.

Supondo-se que há j amostras, calcula-se os postos médios $\bar{R}_j$ de cada amostra seguindo os mesmos princípios do teste de Kruskal-Wallis na seção 2.6.3, página 39. Os postos médios serão comparados dois a dois utilizando a seguinte fórmula para Q_{calc}, vista em Fontelles (2012),

$$Q_{\text{calc}} = \frac{|\overline{R}_A - \overline{R}_B|}{EP} \quad \text{com } EP = \sqrt{\frac{N(N+1)}{12} \cdot \left(\frac{1}{n_A} + \frac{1}{n_B}\right)}$$

onde $\overline{R}_A$ é o posto médio da referida amostra A (analogamente com a amostra B, EP é o erro padrão de cada diferença entre os pontos médios comparados, N é o tamanho total das duas amostras e n_A o tamanho da amostra A (analogamente com a amostra B). Do mesmo jeito que há um fato de correção no teste de Kruskal-Wallis caso haja empates entre os postos, teremos que corrigir o EP caso ocorram no teste de Dunn:

$$EP = \sqrt{\left(\frac{N(N+1)}{12} - \frac{\sum(t^3 - t)}{12(N-1)}\right)\left(\frac{1}{n_A} + \frac{1}{n_B}\right)}$$

, sendo t o número de observações empatadas em cada posto.

Para decidir se as médias das amostras calculadas em Q_{calc} diferem ou não a um nível de significância α, basta fazer $p = \frac{k(k-1)}{2}$ e comparar com $z_{1-\frac{\alpha}{2p}}$ onde z representa os valores da curva normal padrão.

2.7. ANÁLISE DE REGRESSÃO

Depois da coleta de dados de umidade houve a necessidade de analisar se os sensores de solo funcionavam como deveriam. Ou seja, verificamos qual a relação entre a medida coletada pelos sensores e a medida real, que foi medida por um higrômetro analógico. Para isso, vamos rever o conceito de regressão simples vista em Draper e Smith (1998).

Suponha que os dados da amostra sejam representados por um conjunto de ordenados $(x_1,y_1), (x_2,y_2),...,(x_n,y_n)$ onde x é a **variável independente** (ou variável preditora) e o y é a **variável dependente** (ou variável-resposta). Deseja-se encontrar uma reta que melhor se ajuste aos pontos tabelados. Para isso vamos ajustar esses dados ao modelo dado na Equação 3.1:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{2.1}$$

onde β_0 é o coeficiente linear (ou intercepto), o β_1 é o coeficiente angular e o $\epsilon \sim N(0,\sigma^2)$, ou seja, o ϵ é o componente que representa o erro do modelo e que segue uma distribuição Normal de média zero e variância constante.

O Método dos Mínimos Quadrados, descrito em Boldrini et al. (1980, p. 241), cria uma função que representa a soma dos quadrados dos desvios dos pontos observados em relação aos pontos da reta procurada e encontra os coeficientes dessa reta que minimizam essa soma. A técnica é descrita em termos de produto interno e pode ser aplicada a outros tipos de funções. Ela também pode ser vista como uma função de duas variáveis que é minimizada usando derivadas parciais, como visto em Anton, Bivens e Davis (2014) e também em Draper e Smith (1998).

> "Os estatísticos ao utilizarem o método dos mínimos quadrados chamam-no de *análise de regressão*. Assim, fazer regressão linear, quadrática ou exponencial significa aproximar os dados de uma tabela por uma função do tipo $a+bx$, $a+bx+cx^2$ ou $a.e^{bx}$. respectivamente, utilizando o método dos mínimos quadrados"(BOLDRINI et al., 1980, p. 247).

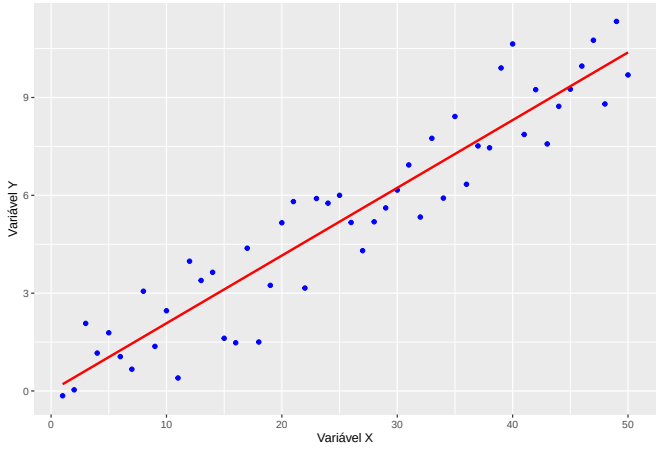
Assim, Draper e Smith (1998) mostram uma fórmula explícita para calcular os coeficientes da reta de regressão procurada. Também afirma que o padrão é usar as letras latinas b_0 e b_1 para as estimativas dos coeficientes no lugar das letras gregas β_0 e β_1.

$$b_1 = \frac{\sum X_i Y_i - \frac{(\sum X_i)(\sum Y_i)}{n}}{\sum X_i^2 - \frac{(\sum X_i)^2}{n}} = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{\sum (X_i - \overline{X})^2}$$
$$b_0 = \overline{Y} - b_1 \overline{X}$$

Onde $\overline{X}$ e $\overline{X}$ são as médias dos valores de X e Y, respectivamente. Considere nessa seção que $\sum = \sum_{i=1}^{n}$, onde n é o tamanho da amostra. Os índices foram retirados da fórmula para melhor visualização. Na Figura 2.15 há um exemplo onde se ajusta uma reta para um conjunto de pontos.

Para testar a significância estatística do coeficiente de regressão b_1, pode-se utilizar a **Análise de Variância (ANOVA)**. A técnica consiste em analisar as fontes de variação das respostas dos dados utilizando o cálculo das somas dos quadrados e

Figura 2.15: Exemplo de uma reta de regressão



Fonte: Autoria própria

a partir disso, calcular a estatística F da regressão para comparar com a distribuição F de Fisher-Snedecor. Em Jayaraman (2000) vimos que é comum determinar $H_0 : b_1 = 0$ como hipótese nula e $H_1 : b_1 \neq 0$ como hipótese alternativa. Uma tabela para a ANOVA de uma regressão linear pode ser organizada como mostra a Tabela 2.2.

Tabela 2.2: Tabela de Análise de Variância - ANOVA

Fonte de Variação	Graus de Liberdade (df)	Soma dos Quadrados (SQ)	Quadrados Médios (QM)	Estatística F
Regressão	1	SQ_{Reg}	$QM_{\text{Reg}} = \frac{SQ_{\text{Reg}}}{df}$	$\frac{QM_{\text{Reg}}}{QM_{\text{Res}}}$
Resíduos	n-2	SQ_{Res}	$QM_{\text{Res}} = \frac{SQ_{\text{Res}}}{df}$	
Total	n-1	SQ_{Total}		

Fonte: (JAYARAMAN, 2000, p. 55)

A Soma dos Quadrados da Regressão (SQ_{Reg}) e a Soma dos Quadrados Total (SQ_{Total}) são calculados da seguinte maneira:

$$SQ_{\text{Reg}} = \frac{\left[\sum xy - \frac{\sum x \sum y}{n}\right]^2}{\sum x^2 - \frac{(\sum x)^2}{n}}$$

$$SQ_{\text{Total}} = \sum y^2 - \frac{(\sum y)^2}{n}$$

A Soma dos Quadrados dos Resíduos (SQ_{Res}) é dada por $SQ_{\text{Res}} = SQ_{\text{Total}} - SQ_{\text{Reg}}$. Dados d_1, d_2 os graus de liberdade *df* da Regressão e dos Resíduos, respectivamente e B a função Beta $B(x,y) = \int_0^1 t^{x-1}(1-t)^{y-1}\,dt$, a distribuição de Fisher-Snedecor tem como função de densidade de probabilidade

$$f(x; d_1, d_2) = \frac{1}{\mathrm{B}\left(\frac{d_1}{2}, \frac{d_2}{2}\right)} \left(\frac{d_1}{d_2}\right)^{\frac{d_1}{2}} x^{\frac{d_1}{2}-1} \left(1 + \frac{d_1}{d_2}\, x\right)^{-\frac{d_1+d_2}{2}}$$

que também pode ser descrita em termos da função Γ, como visto em DeGroot e Schervish (2014, p. 598).

Uma técnica bastante utilizada para avaliar a eficiência de uma regressão linear é o cálculo do **Coeficiente de Determinação** (R^2), que como visto em Jayaraman (2000) é calculado pela razão entre a Soma dos Quadrados da Regressão e a Soma dos Quadrados Total.

$$R^2 = \frac{SQ_{\text{Reg}}}{SQ_{\text{Total}}}$$

Assim, o R^2 representa o percentual da variação da variável dependente y que pode ser explicado pela variação da variável independente x. Em outras palavras, o quanto dos dados observados do fenômenos podem ser explicados pela reta de regressão encontrada.

Em Fontelles (2012, p. 50), vimos que ele também pode ser determinado como o quadrado do coeficiente de correlação produto-momento r, também conhecido como coeficiente de correlação de Pearson, com símbolo ρ.

$$r = \frac{\text{cov}_{xy}}{s_x . s_y}, \text{ onde: } \text{Cov}_{xy} = \frac{\sum (x - \overline{x})(y - \overline{y})}{n - 1}$$

onde s é o desvio-padrão de cada variável e Cov é a covariância. O r varia entre 1 e -1. Se o r for positivo, dizemos que as variáveis têm uma associação linear direta, e se for negativo, uma associação linear indireta, como visto em Kutner et al. (2005). Quanto mais próximo de 1 (ou -1), mais forte essa associação.

Também em Fontelles (2012) há uma lista de pressupostos que devem ser observadas caso seja necessário utilizar a reta de regressão para fazer previsões:

- **Normalidade** - Ao menos a variável-resposta de ser normalmente distribuída.
- **Homocedasticidade** - A variância das variáveis deve ser a mesma.
- **Linearidade** - Os dados devem ter uma tendência linear, positiva ou negativa. Isso pode ser verificado pelo diagrama de dispersão dos dados.
- **Aleatoridade** - Os valores dos dados devem ser obtidos aleatoriamente.

CAPÍTULO 3

RESULTADOS E DISCUSSÃO

Nesse capítulo organizaremos os resultados e as discussões do estudo por blocos de informação para uma melhor compreensão do leitor. Trataremos dos resultados relativos a temperatura e umidade do ar e os resultados relativos a umidade do solo em seções separadas.

3.1. DADOS DE TEMPERATURA E UMIDADE DO AR

Foram analisados os dados de temperatura e umidade do ar entre 00:00 do dia 25-11-2021 e 23:59 do dia 16-12-2021. Na Tabela 3.1 tem-se as temperaturas mínimas, médias e máximas para cada dia, com um conjunto de colunas para os dados da estação construída UFRPE e outro para os dados da estação Aeroporto. Da mesma maneira, na Tabela 3.2, vemos a umidade relativa do ar mínima, média e máxima diária para o período, com uma coluna com os dados referentes à estação UFRPE e a outra coluna referente aos dados da estação Aeroporto.

Na Figura 3.1, vemos como se concentram as quantidades de medidas de temperatura mínima, média e máxima para o período total. As velas laranjas são referentes à estação Aeroporto e as de cor ciano são referentes à estação UFRPE. Nota-se que as temperaturas mínimas da estação UFRPE foram mais baixas que as da estação Aeroporto. As médias estão próximas e as máximas para a estação UFRPE foram bem mais altas que as da estação Aeroporto, a ponto de nenhuma máxima ter algum valor próximo a alguma máxima da estação UFRPE, com exceção dos *outliers*. Da mesma maneira, na Figura 3.2, temos as medidas para a umidade

Tabela 3.1: Mínimas, médias e máximas diárias de temperatura (°C)

Data (M/D/A)	Categorias					
	Estação UFRPE			Estação Aeroporto		
	Mínima (°C)	Média (°C)	Máxima (°C)	Mínima (°C)	Média (°C)	Máxima (°C)
11/25/2021	25	29,17	34	27	28,71	31
11/26/2021	26	29,42	34	27	28,83	31
11/27/2021	26	28,58	34	27	28,88	31
11/28/2021	24	27,83	34	26	28,29	30
11/29/2021	25	28,46	33	27	28,83	31
11/30/2021	25	28,62	34	28	28,92	31
12/1/2021	25	28,79	34	27	28,67	31
12/2/2021	25	28,92	34	27	28,79	31
12/3/2021	24	29,21	35	27	28,75	31
12/4/2021	25	28,88	34	24	28,17	30
12/5/2021	25	28,58	34	25	28,17	31
12/6/2021	26	28,83	34	26	28,33	30
12/7/2021	25	29,04	35	27	28,38	30
12/8/2021	26	28,83	34	27	28,71	31
12/9/2021	25	28,92	34	27	28,58	30
12/10/2021	26	28,29	33	26	28,46	30
12/11/2021	26	28,79	33	27	28,62	31
12/12/2021	26	29,54	34	27	28,62	31
12/13/2021	24	25,88	29	24	25,67	28
12/14/2021	24	27,54	33	26	28,29	30
12/15/2021	24	28,62	34	25	28,08	30
12/16/2021	24	28,12	32	24	28,08	30

Fonte: Autoria própria

Tabela 3.2: Mínimas, médias e máximas diárias de umidade relativa do ar (%)

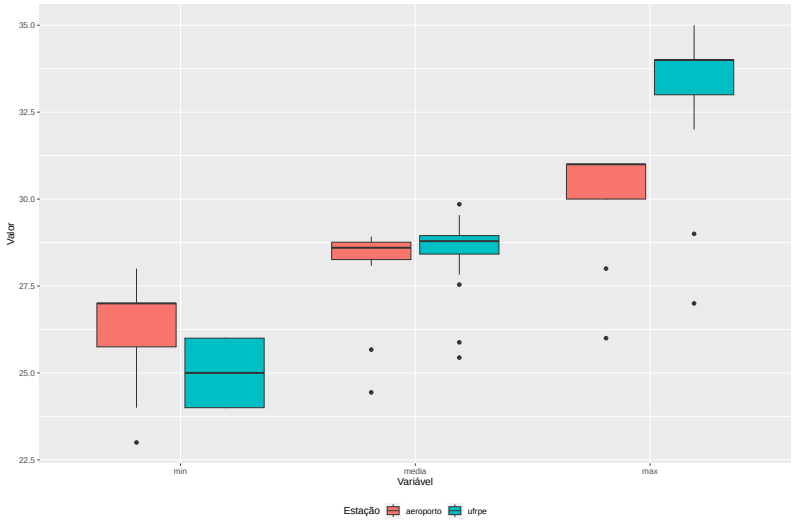
Data (MM/DD/AAAA)	Categorias					
	Estação contruída			Estação Aeroporto		
	Mínima (%)	Média (%)	Máxima (%)	Mínima (%)	Média (%)	Máxima (%)
11/25/2021	46	72,5	89	59	70,38	79
11/26/2021	51	73,12	89	62	70,96	79
11/27/2021	51	76,79	91	59	69,75	79
11/28/2021	58	82,29	92	66	73,92	84
11/29/2021	55	78,08	91	62	70,54	79
11/30/2021	55	77,33	91	59	69,92	79
12/1/2021	46	74	91	59	70,04	79
12/2/2021	49	73,83	89	59	68,46	74
12/3/2021	55	76,42	91	62	71,21	79
12/4/2021	57	80,92	92	66	78	100
12/5/2021	52	79,62	92	59	75,17	94
12/6/2021	50	75,38	89	59	70,17	89
12/7/2021	47	71,96	90	59	67,71	74
12/8/2021	56	78,38	90	59	70,04	74
12/9/2021	47	76,96	91	62	71,58	79
12/10/2021	59	80,42	90	66	71,67	89
12/11/2021	55	75,04	90	59	68,33	79
12/12/2021	44	70,88	86	59	67,54	74
12/13/2021	85	90	92	66	85,54	94
12/14/2021	65	83,58	94	62	72,33	89
12/15/2021	55	78,46	94	62	73,92	89
12/16/2021	63	81,33	92	62	73,46	94

Fonte: Autoria própria

relativa do ar. Nesse caso, o comportamento se manteve, com exceção das umidades médias, que no caso da estação UFRPE foram um pouco maiores que as da estação UFRPE. As mínimas da estação UFRPE e as máximas da estação Aeroporto tiveram uma variação maior.

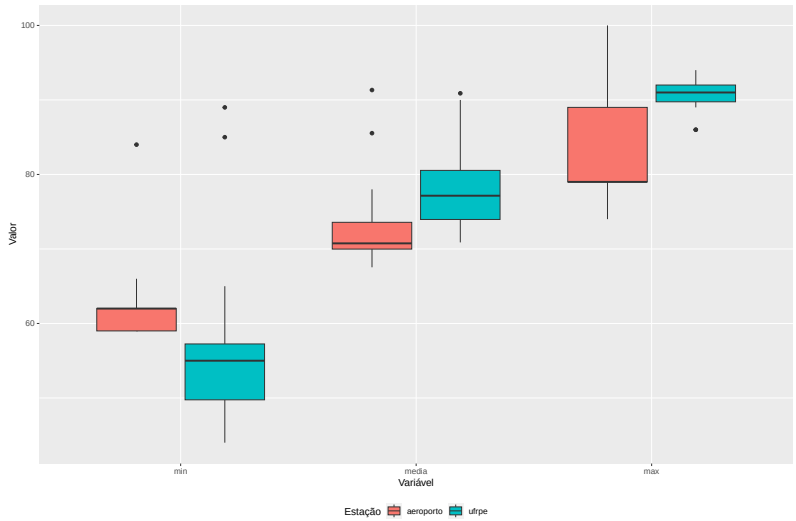
Observando as médias das temperaturas mínima, média e máxima, temos que: para a estação Aeroporto, a média das mínimas foi 26,21°C enquanto para UFRPE foi 25,04°C. A média das médias foi 28,26°C para Aeroporto e 28,51°C para UFRPE e a média das máximas foi 30,29°C para Aeroporto e 33,33°C para UFRPE. Em relação à umidade do ar, temos que a umidade média das mínimas parar a estação Aeroporto foi de 62,21% enquanto que para a estação UFRPE foi de 55,92%. A umidade média das médias foi de 72,58% para Aeroporto e 77,91% para UFRPE, e a umidade média das máximas foi de 83,46% para Aeroporto e 90,58% para UFRPE.

Figura 3.1: Boxplot com os valores para temperatura mínima, média e máxima do período



Fonte: Autoria Própria

Figura 3.2: Boxplot com os valores para umidade mínima, média e máxima do período



Fonte: Autoria Própria

Foram comparados os valores de temperatura totais gerado pelas duas estações bem como as mínimas, médias e máximas computadas. Na Tabela 3.4 são apresentados os resultados dos p-valor dos testes para as medidas. Da mesma maneira, na Tabela 3.3, os resultados dos testes estatísticos para os casos envolvendo a umidade do ar. A umidade do ar média diária medida pela estação aeroporto apresentou normalidade pelo teste Shapiro-Wilk com p-valor 0,0555, mas ao comparar utilizando o teste de Mann-Whitney com as médias diárias da estação UFRPE, foi identificado uma diferença significativa das medianas com p-valor $5{,}079 \times 10^{-5}$.

Nenhuma classe de temperatura apresentou normalidade seguindo Shapiro-Wilk, porém, as medianas das temperaturas médias diárias apresentaram similaridade estatística utilizando Mann-Whitney com p-valor 0,11, o que pode ser observado na Figura 3.1.

Foi feito um cálculo do coeficiente de correlação de Pearson entre as variáveis temperatura e umidade de cada uma das estações, com os resultados mostrados na Figura 3.3. As variáveis *temp_sta* e *umi_sta* representam a temperatura e a umidade do ar para a estação UFRPE, respectivamente e as variáveis *temp_apt* e *umi_apt* representam a temperatura e a umidade do ar medidas na estação Aeroporto.

Todos os p-valor das correlações são menores que $2{,}2 \times 10^{-16}$, indicando alta

Tabela 3.3: Testes estatísticos envolvendo dados de umidade do ar

Dados de Umidade	Shapiro-Wilk (p-valor)	Mann-Whitney (p-valor)
Totais (Estação)	$6{,}353 \times 10^{-21}$***	$4{,}695 \times 10^{-21}$***
Totais (Aeroporto)	$2{,}828 \times 10^{-17}$***	
Mínimas (Estação)	$4{,}933 \times 10^{-7}$***	$4{,}379 \times 10^{-5}$***
Mínimas (Aeroporto)	$1{,}05 \times 10^{-4}$***	
Médias (Estação)	$1{,}628 \times 10^{-5}$***	$5{,}079 \times 10^{-5}$***
Médias (Aeroporto)	0,0555	
Máximas (Estação)	0,0043*	0,0013*
Máximas (Aeroporto)	0,0452*	

Fonte: Autoria própria

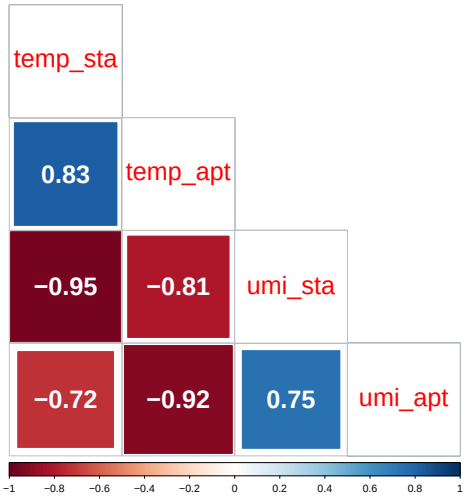
Tabela 3.4: Testes estatísticos envolvendo dados de temperatura

Dados de Temperatura	Shapiro-Wilk (p-valor)	Mann-Whitney (p-valor))
Totais (UFRPE)	$3{,}7537 \times 10^{-17}$***	0,0224*
Totais (Aeroporto)	$4{,}825 \times 10^{-17}$***	
Mínimas (UFRPE)	0,001120**	0,00065***
Mínimas (Aeroporto)	0,00037***	
Médias (UFRPE)	$2{,}681 \times 10^{-7}$***	0,11
Médias (Aeroporto)	0,0002***	
Máximas (UFRPE)	$9{,}184 \times 10^{-7}$***	$2{,}606 \times 10^{-7}$***
Máximas (Aeroporto)	$6{,}34 \times 10^{-7}$***	

Fonte: Autoria própria

significância. A entre as temperaturas tivemos $r = 0{,}83$ e entre as umidades $r = 0{,}75$, o que mostra que uma associação direta forte entre as estação. Ao analisar as correlações entre as variáveis na mesma estação, nota-se um comportamento parecido: Na estação UFRPE tivemos $r = -0{,}95$ e na estação Aeroporto $r = -0{,}92$. Isso mostra que em ambas as estações a relação entre temperatura e umidade relativa do ar é indireta e muito forte, no sentido que quanto maior a temperatura, menor a umidade relativa do ar, que é uma característica do clima de Recife, dado por Am segundo a classificação de Köppen em Saboya et al. (2021).

Figura 3.3: Correlações entre as grandezas medidas



Fonte: Autoria Própria

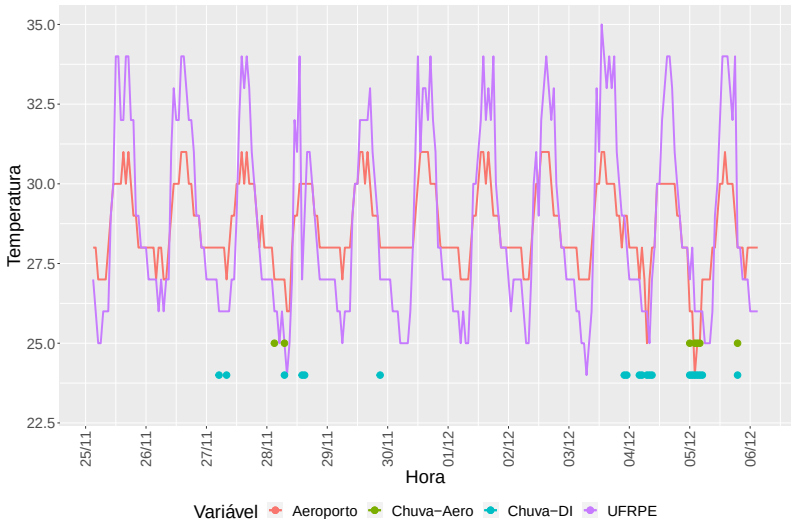
Os dados obtidos da estação Aeroporto possuem uma variável que caracteriza fenômenos especiais. Cada entrada é formada por uma junção de variáveis categóricas indicando um fenômeno natural. O *dataframe* foi filtrado para mostrar todas as entradas não-vazias para essa variável e posteriormente foi fixada duas novas variáveis constantes: *chuva*, para representar a umidade no momento em que houve chuva, com valor 70 e a variável *chuva_ temp* para representar a temperatura quando houve chuva, com valor de 25.

Por outro lado, os dados da estação Dois Irmãos possuem uma medida de precipitação horária. Ainda assim, foi decidido utilizar essas informações da mesma maneira da estação anterior, filtrando o *dataframe* e criando as variáveis *cema* e

cema_temp para representar a umidade e a temperatura do ar, com valores de 65 e 24 respectivamente.

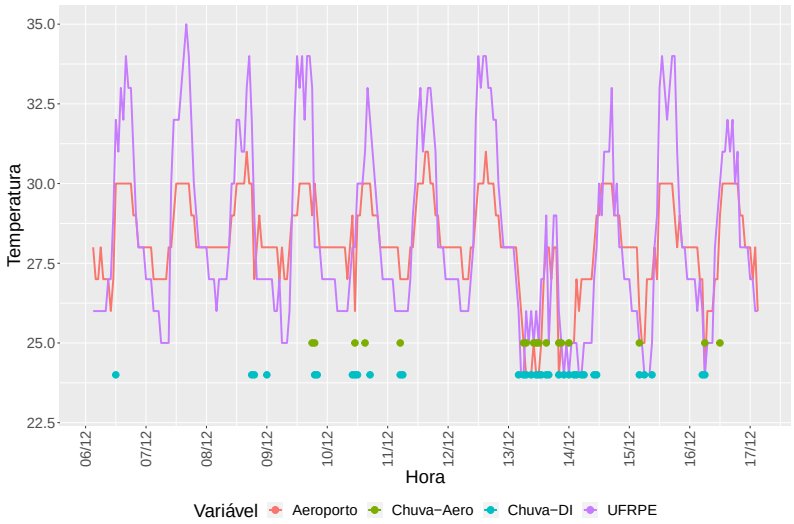
Essa abordagem foi utilizada para representar os momentos onde ocorreram chuva nas estações Dois Irmãos e Aeroporto e relacionar com os gráficos de temperatura e umidade do ar mostrado a seguir. Nas Figuras 3.4 e 3.5 temos um gráfico da temperatura para cada hora com as duas estações ao mesmo tempo. Na legenda, além dos valores de temperatura para ambas as estações, são destacados os momentos onde ocorreram chuva para as estações Aeroporto (Chuva-Aero) e Dois Irmãos (Chuva-DI).

Figura 3.4: Temperatura das estações: 25/11 a 06/12



Fonte: Autoria Própria

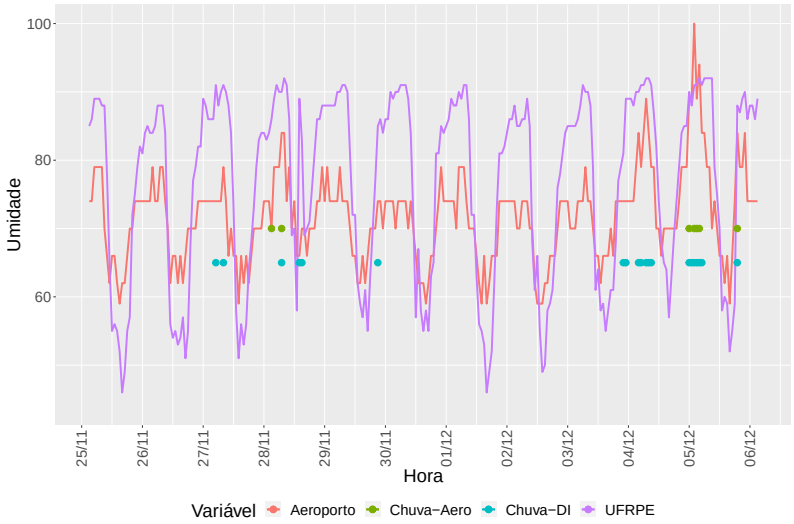
Figura 3.5: Temperatura das estações: 06/12 a 17/12



Fonte: Autoria Própria

Nas Figuras 3.6 e 3.7 temos um gráfico da umidade do ar com as indicações dos momentos de chuva da mesma maneira que nos gráficos de temperatura.

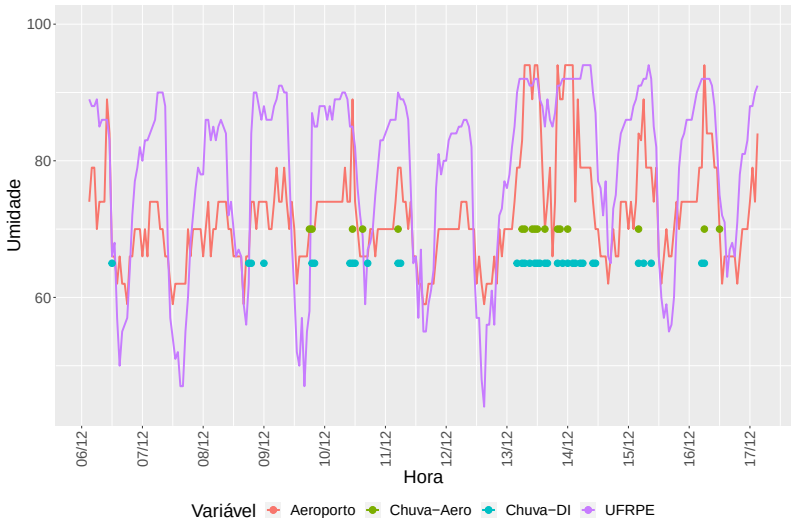
Figura 3.6: Umidade do ar das estações: 25/11 a 06/12



Fonte: Autoria Própria

Os gráficos nas Figuras 3.4, 3.5, 3.6 e 3.7, indicam alguns períodos de chuva que alteraram o padrão das medidas de ambas estações: Entre 28/11 e 29/11, entre 04/12 e 06/12 e entre 13/12 e 15/12. A alteração no padrão se deu tanto na temperatura quanto na umidade do ar de ambas estações, mostrando que o aumento de chuva

Figura 3.7: Umidade do ar das estações: 06/12 a 17/12

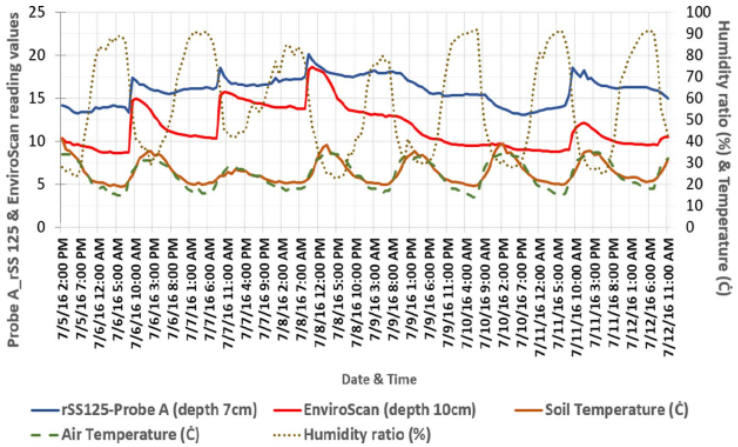


Fonte: Autoria Própria

aumenta a umidade e baixa a temperatura, corroborando os cálculos das correlações feitos previamente.

A estação construída em Ramadan et al. (2018) também utilizou um sensor DHT11 no seu teste, realizado no sudeste da Espanha em um clima semiárido. Na Figura 3.8, é possível observar que o comportamento da temperatura e umidade do ar são linearmente indiretos, provavelmente indicando uma correlação negativa. Não foi possível calcular esses valores pois não os dados do experimento não estão disponíveis.

Figura 3.8: Performance da estação de Ramadan et al. (2018)



Fonte: (RAMADAN et al., 2018)

Foi feita uma análise descritiva para a temperatura e a umidade do ar baseada nas diferenças entre as medidas da estação UFRPE e estação AEROPORTO. A essas diferenças chamamos de resíduos nesse texto. As Tabelas 3.5 e 3.6 indicam essas estatísticas.

Novelan e Amin (2020) fizeram testes para analisar as leituras do seu sistema de monitoramento. Foi utilizado um termômetro digital como método de referência e o sistema foi medido sob estresse de calor. Foram feitas 15 medidas de temperatura e 15 de umidade do ar e feitas comparações dos resíduos de cada uma delas. A média dos resíduos de temperatura foi 0,97 e a média dos resíduos de umidade foi 5,9. A estação UFRPE teve média de resíduos de temperatura de 0,2145 e de umidade de 5,562, valores muito próximos aos de Novelan e Amin (2020), porém com uma quantidade maior de leituras, 550 no caso da Estação UFRPE.

Tabela 3.5: Estatísticas dos Resíduos da Temperatura

Mínimo	25%	Mediana	Média	75%	Máximo	Desvio-Padrão
-3	-1	0	0,2145	2	5	1,842

Tabela 3.6: Estatísticas dos resíduos da Umidade do Ar

Mínimo	25%	Mediana	Média	75%	Máximo	Desvio-Padrão
-19	-2	8	5,562	12	23	9,037

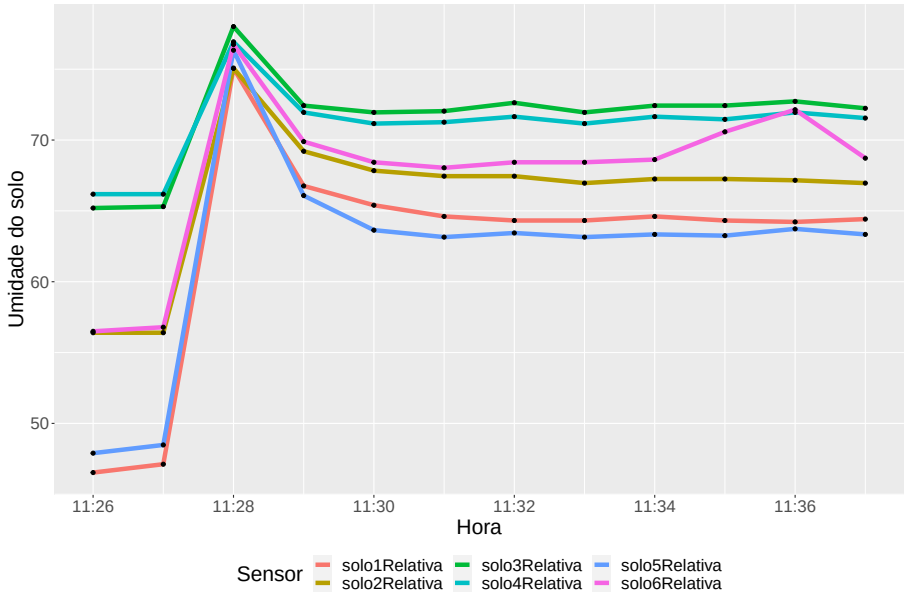
É importante lembrar que a distância das estação UFRPE e Aeroporto é aproximadamente 12 Km. Além disso, as Figuras 2.10a e 2.10b mostram a posição do sensor DHT11 e a telha que protege o sensor pode ter influenciado na variação de temperatura e umidade medida.

3.2. DADOS DE UMIDADE DO SOLO

Foi realizado um teste onde os 6 higrômetros de solo foram colocados em um pote (ver Figura 2.11b), por um período de 12 minutos com uma rega de aproximadamente 300 ml de água, com a estação fazendo medições a cada 5 segundos. Os dados foram tratados de modo que a cada minuto tivéssemos a média de umidade do solo medidas no referido minuto pelos sensores. Consideraremos nesse caso só

as medidas de umidade relativa, em percentual. Esses dados geraram o gráfico da Figura 3.9.

Figura 3.9: Umidade do solo (em %) de um único pote



Fonte: Autoria Própria

Para cada uma das medidas foi realizado o teste de Shapiro-Wilk, cujos resultados podem ser vistos na Tabela 3.7a. Como todas as medidas não têm amostras vindas de uma população normal, realizamos o teste de Kruskal-Wallis, com os preceitos visto na Seção 2.6.3, já que temos mais de três categorias para comparar. O p-valor para o teste de Kruskal-Wallis foi $5{,}152 \times 10^{-09}$, significativo para 5%. Com isso, temos que ao menos um par de medidas difere significativamente e portanto executamos o teste de Dunn para identificar essas diferenças (ver Seção 2.6.4).

O resultados do teste de Dunn podem ser vistos na Tabela 3.7b. Desses resultados, podemos concluir que: O sensor *solo1* tem medidas próximas ao do *solo2* e *solo5*, o sensor *solo2* ao *solo6*, o *solo3* ao *solo4* e *solo4* ao *solo6*. Nota-se essas diferenças nas leituras quando cada sensor é posto em um vaso distinto (ver Figura 2.11a). O gráfico na Figura 3.10 mostra essa diferença aparente, com picos de aumento de umidade do solo, representando as irrigações diárias. Os sensores de umidade do solo tiveram uma boa resposta com relação à adição de água nos vasos, embora as medidas tenham sido diferentes no mesmo vaso, o que pode indicar variação nas medidas por causa da distância entre os pontos de medição. O sensor

Tabela 3.7: Testes estatísticos para a umidade do solo

Medida	Shapiro-Wilk (p-valor)
Solo1	0.0013964***
Solo2	0.00301***
Solo3	0.002693***
Solo4	0.004896***
Solo5	0.003189***
Solo6	0.009731***

(a) Teste de Shapiro-Wilk.

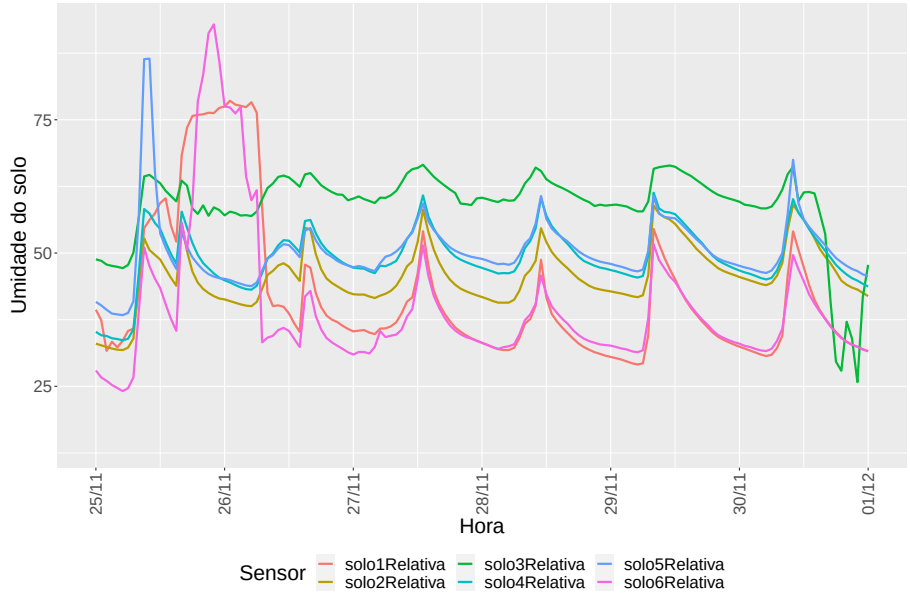
	solo1	solo2	solo3	solo4	solo5
solo2	0.0972				
solo3	0.0000*	0.0037*			
solo4	0.0006*	0.0264*	0.2292		
solo5	0.2678	0.0276*	0.0000*	0.0001*	
solo6	0.0157*	0.1966	0.0340*	0.1364	0.0028*

(b) Matriz de p-valores do teste de Dunn

Teste de Kruskal-Wallis com p-valor igual a 5,12e-06.
Fonte: Autoria própria

FC-28 é bastante sensível à ação da água no solo e como é um sensor resistivo, a corrosão natural por causa da água de irrigação poderá dificultar as leituras feitas pelo produto.

Figura 3.10: Umidade do solo em diferentes potes

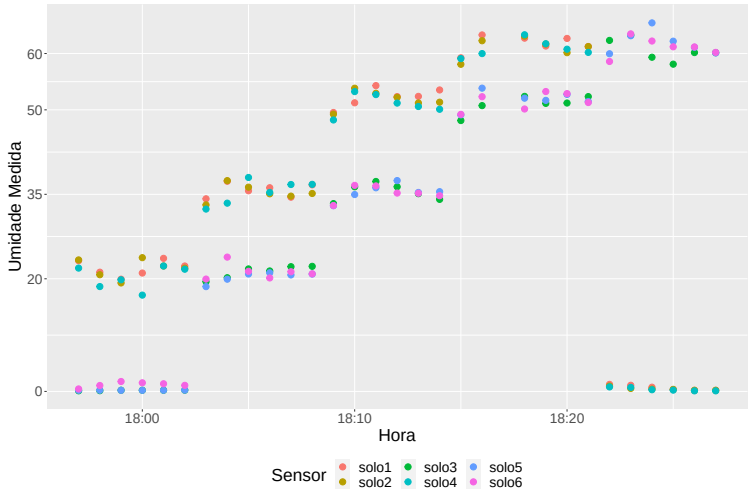


Fonte: Autoria própria

Por fim, seguindo os caminhos apontados em González-Teruel et al. (2019), Bitella et al. (2014) e Mendes et al. (2021), foi feito um teste para verificar se as medidas dos higrômetros correspondiam a uma medida de umidade real. Com pequenos potes com solo, foram feitos pequenos furos para escoar a água que foi colocada em quantidades diferentes para obter diferentes níveis de umidade do solo.

Utilizando o higrômetro analógico da Figura 2.13, foram observados os seguintes níves: 20%, 35%, 50% e 60%, com um outro nível de controle 0% que consistia em uma medida do higrômetro sem estar no solo. Assim, os higrômetros foram divididos em dois blocos: um com os sensores *solo1*, *solo2* e *solo4* e outro com os sensores *solo3*, *solo5* e *solo6*. Cada bloco foi colocado por aproximadamente 6 minutos em cada pote. Na Figura 3.11 mostra as medidas de cada sensor durante o período do teste. Na Figura 3.12 temos as medidas feitas por cada sensor em função da umidade real observada no higrômetro analógico.

Figura 3.11: Umidade medida pelos sensores por minuto



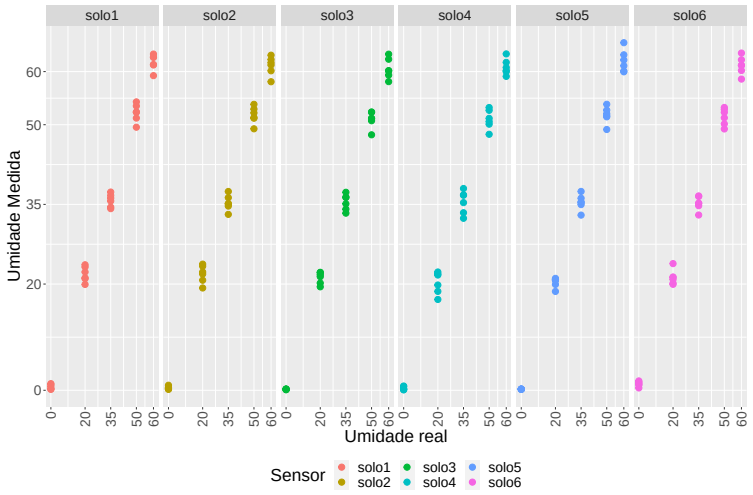
Fonte: Autoria Própria

Com os dados da umidade medida em função da umidade real, foi feita uma regressão linear para analisar esses resultados. Primeiro, foi feito um teste de normalidade de Shapiro-Wilk, com os resultados na Tabela 3.8, que indicam que os dados não seguem uma normalidade. Na Tabela 3.9 temos os valores dos coeficientes da regressão e na Tabela 3.10 as estatísticas do modelo de regressão. Na Figura 3.13 temos os dados e a reta de regressão encontrada.

Tabela 3.8: Resultados do Teste de Normalidade Shapiro-Wilk

Variável	W	p-valor
valor	0,8991	$1{,}009 \times 10^{-9}$
umidade_real	0,87587	$4{,}885 \times 10^{-11}$

Figura 3.12: Umidade medida em função da umidade real por sensores



Fonte: Autoria Própria

Tabela 3.9: Resumo dos Coeficientes da Regressão Linear

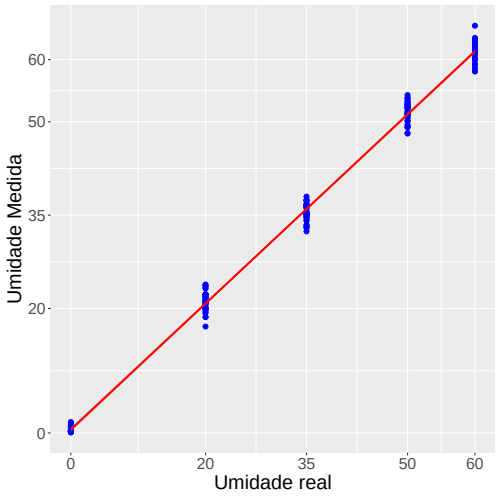
Parâmetro	Estimativa	Erro Padrão	Valor t	p-Valor
(Intercepto)	0,54413	0,19850	2,741	0,00675 **
umidade_real	1,01284	0,00505	200,556	$< 2 \times 10^{-16}$ ***

Assim, a reta de regressão pode ser descrita pela na Equação 3.1. O coeficiente linear é bem próximo do 0 e o angular bem próximo do 1, com p-valores significativos de acordo com a Tabela 3.9. Os erros padrão são também pequenos. Assim, ao medir a umidade real, há o resultado medido pela estação terá uma margem de erro de ±3,5 pontos percentuais de umidade, com 95% de confiança, na faixa de medição entre 0% e 60%.

$$y = 0{,}54413 + 1{,}01284.x \tag{3.1}$$

Em Mendes et al. (2021) foi feita uma calibração do sensor FC-28 em substrato comercial. O método de referência foi o método gravimétrico e foi mapeada uma função com a umidade volumétrica como variável independente e os bits de saída do sensor, variando em 0-1023 como variável dependente. Na Figura 3.14 podemos ver o gráfico de dispersão dessa função, que o autor não conseguiu uma função polinomial que ajustasse a esses dados.

Figura 3.13: Reta de regressão linear para o modelo



Fonte: Autoria Própria

Tabela 3.10: Estatísticas do Modelo de Regressão

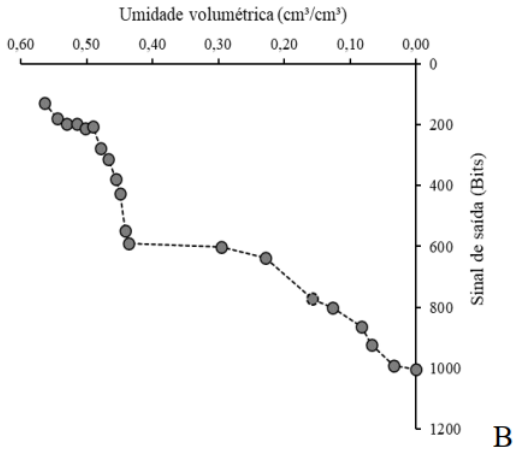
Erro Padrão Residual	Graus de Liberdade(Resíduos)	R^2	Estatística F	p-valor do Modelo
1,447	178	0,9956	$4{,}022 \times 10^{04}$	$<2{,}2 \times 10^{-16}$

Portanto, por causa da variação de sinal que ocorreu entre as umidades 0,3 e 0,4 cm^3cm^{-3}, o autor decidiu dividir a função em duas faixas e ajustar funções diferentes para cada uma, como mostrado na Figura 3.15. O R^2 para ambas as funções foi bem alto, com um ajuste excelente.

Foram feitos em Lu e Chen (2007), alguns testes de calibração comparando diferentes tipos de sensores de solo e diferentes funções para a calibração em solos saturados de sal. Foram comparados sensores resistivos e capacitivos com funções lineares simples e polinomiais. Nesse texto, houve a conclusão que sensores resistivos, como o FC-28, têm uma precisão menor que os capacitivos, e o modelo de regressão linear simples é o que mostrou menor precisão. Esses resultados corroboram os resultados de Mendes et al. (2021) e vão de encontro aos resultados reportados na Tabela 3.10.

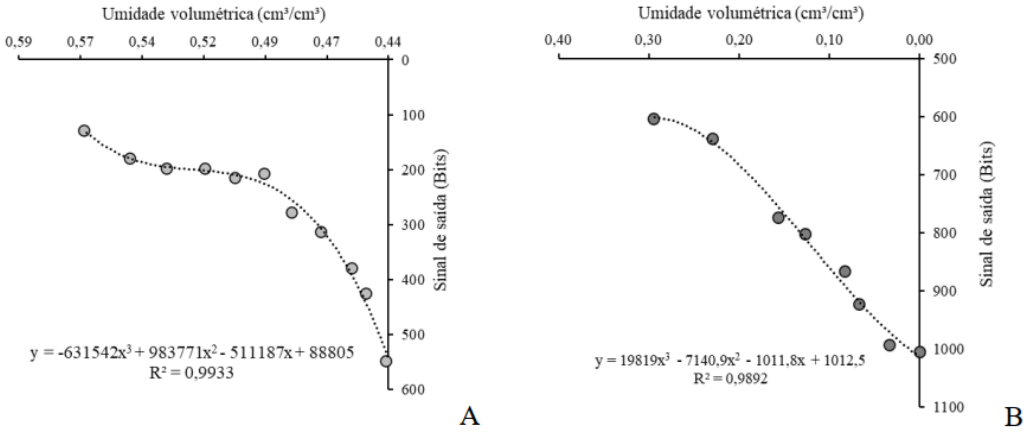
Porém, para ser utilizado do ponto de vista prático, o modelo de Mendes et al. (2021) precisa ser embutido na programação da placa Arduino, forçando que ele só possa ser utilizado com o substrato comercial que foi utilizado na calibração. O modelo mostrado nessa tese é muito mais simples de ser interpretado, embora

Figura 3.14: Saída em função da umidade volumétrica em Mendes et al. (2021)



Fonte: (MENDES et al., 2021)

Figura 3.15: Calibração do FC-28 em Mendes et al. (2021)



Fonte: (MENDES et al., 2021)

tenha menos precisão que modelos polinomiais. Caso haja a necessidade de usar em outros tipos de solo, é mais fácil adaptar-se a uma nova calibração fazendo uma medição como feita nessa tese do que refazer a programação na placa Arduino, o que seria extremamente trabalhoso e técnico para o usuário comum. Ainda assim, em Gujarati e Porter (2011) é frisada a necessidade de normalidade e aleatoriedade para fazer previsões com um modelo linear, mas o comportamento do sensor e a praticidade permite a utilização com pequenas ressalvas acerca da precisão.

A estação funcionou por 20 horas sem estar diretamente ligada à energia em um teste, que porém só iniciou-se de meio-dia, com pouco tempo de luz solar se fosse ligado de manhã, então há sugestão de novos testes para avaliar a autonomia

da estação. Também seria importante atestar quanto tempo leva para carregar as baterias para se ter uma janela adequada de funcionamento caso a estação seja utilizada em ambientes que passem muito tempo sem energia.

Uma estação totalmente autônoma como a reportada por Ramadan et al. (2018), López-Vargas et al. (2019) e Gutiérrez et al. (2013) necessita de um cuidado maior com as placas fotovoltaicas além de uma quantidade maior delas, o que pode aumentar demais o custo. A indicação mais importante seria utilizar estratégias para reduzir o consumo de energia do sistema. López-Vargas et al. (2019) fizeram isso de três maneira: Diminuindo a frequência do microcontrolador; utilizando bibliotecas que criam um modo *stand-by* com modos de baixa energia na estação; desabilitando o leitor LCD e fazendo ligação manual dele. Gutiérrez et al. (2013) também utilizam um modo *stand-by* na sua na estação assim como componentes com baixa energia para que a placa solar possa carregar as baterias utilizadas.

Por fim, na Tabela 3.11, temos uma lista com o custo dos componentes da estação no mercado nacional, atualizado em Agosto de 2024. Esses preços não levam em conta o frente nem a mão-de-obra para montar a placa. preço não entra o valor da mão-de-obra de quem for fazer nem o frete dos produtos.

Tabela 3.11: Custo estimado dos componentes principais em Agosto/24

Componente	Quantidade	Preço (R$)	Total (R$)
Arduino AtMega328	1	45,00	45,00
Sensor DHT11	1	16,00	16,00
Sensor FC-28	6	8,00	48,00
Placa Solar CDR02	1	90,00	90,00
Bateria 18650	4	15,00	60,00
Módulo RTC DS1302	1	10,00	10,00
Diodo 1N5819 Schottky	4	1,50	6,00
Regulador Xl6019	2	21,00	42,00
Display OLED I2C	1	26,00	26,00
Módulo Regulador Carregador de bateria	1	4,50	4,50
Módulo cartão microSD	1	14,00	14,00
Cabo PP 2v 0,50mm 2x0,5 50m	1	123,00	123,00
Outros	1	100,00	100,00

Fonte: Autoria Própria

Dadas as possibilidades de uso da estação e o custo estimado em R$ 584,50, nota-se que a estação é bem barata. Desse valor, R$ 182,00 (31% do total) são voltados para a autonomia energética da estação e R$ 171,00 (29%) voltados para os sensores de umidade do solo e cabos para ligar os sensores à estação. Muitas estações à venda no mercado são mais caras com menos funcionalidades, ou com funcionalidades que não agregam tanto valor ao custo geral da estação.

CAPÍTULO 4
CONCLUSÕES

Com base no que foi apresentado, foi possível atestar a viabilidade da construção da estação experimental bem como executar e validar os testes. As leituras de temperatura e umidade do ar foram consideradas satisfatórias tendo como base os dados da estação AEROPORTO e da estação Dois Irmãos, além de ter sido possível detectar momentos de chuva durante o período de teste. As leitura de umidade do solo tiveram valores esperados satisfatórios dadas as condições na calibração, porque o modelo apresentado é mais fácil de interpretar embora tenha menos precisão que os recomendados pela literatura. Além disso, o preço calculado foi bem reduzido em relação a outras estações disponíveis no mercado.

A pesquisa foi muito afetada pela pandemia de COVID-19 (2020-2023), o que não permitiu o uso de instalações mais propícias para a realização dos testes na estação. Para estudos futuros, seria recomendado fazer testes mais relevantes com o sensor de umidade do solo em relação ao tempo de uso, já que o tipo de sensor utilizado pode ter uma vida útil menor que o esperado. Da mesma maneira, seria interessante testar mais os procedimentos de recarga das baterias, para saber qual o intervalo mínimo de energia da rede elétrica é necessária para que a estação funcione sem parar.

Esse protótipo pode servir como base para estações com mais funções que auxiliem na Agricultura de precisão. Se for ligada a atuadores específicos pode ser utilizada para controlar a irrigação de uma plantação. Caso aumente-se a sua autonomia, pode ser utilizada para monitorar as grandezas ambientais em locais sem acesso a energia. Aliada a um módulo GSM os dados podem ser transmitidos pela

internet em locais com cobertura de antenas de telefonia. Também pode ser uma excelente aliada na pesquisa acadêmica, visto que o período entre os anos de 2015 a 2022 houve uma diminuição nas estações meteorológicas no Brasil e esses dados vão ficando cada vez mais escassos, tornando estações de baixo custo uma alternativa viável e necessária.

REFERÊNCIAS

ADITIA, I.; ILHAM, R. Penetas Telur Otomatis Berbasis Arduino Dengan Menggunakan Sensor DHT11. **Jurnal Ilmiah Mahasiswa Kendali dan Listrik**, v. 3, n. 1, p. 113–119, 2022.

ALMEIDA, B. M. de et al. Integração de sistema mobile com sistemas que utilizam a plataforma arduino aplicados à agricultura. **Tekhne e Logos**, v. 10, n. 2, p. 62–75, 2019.

ANTON, H.; BIVENS, I.; DAVIS, S. **Cálculo-Volume II**. [S.l.]: Bookman Editora, 2014.

BHADANI, P.; VASHISHT, V. Soil moisture, temperature and humidity measurement using arduino. In: IEEE. 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence). [S.l.: s.n.], 2019. p. 567–571.

BITELLA, G. et al. A novel low-cost open-hardware platform for monitoring soil water content and multiple soil-air-vegetation parameters. **Sensors**, MDPI, v. 14, n. 10, p. 19639–19659, 2014.

BOLDRINI, J. L. et al. **Álgebra linear**. [S.l.]: Harbra, 1980.

CEMADEN. **Estação Meteorológica Recife - Dois Irmãos**. 2024. Disponível em: <https://resources.cemaden.gov.br/graficos/interativo/grafico_CEMADEN.php>. Acesso em: 1 jun. 2024.

CHOI, W. et al. An Algorithm for Computing the Exact Distribution of the Kruskal–Wallis Test. **Communications in Statistics - Simulation and Computation**, Taylor & Francis, v. 32, n. 4, p. 1029–1040, 2003. Disponível em: <https://doi.org/10.1081/SAC-120023876>.

DEBELE, G. M.; QIAN, X. Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor. In: IEEE. 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). [S.l.: s.n.], 2020. p. 428–432.

DEGROOT, M. H.; SCHERVISH, M. J. **Probability and statistics**. [S.l.]: Pearson Education London, UK: 2014. v. 563.

DRAPER, N. R.; SMITH, H. **Applied regression analysis**. [S.l.]: McGraw-Hill. Inc, 1998.

DUNN, O. J. Multiple comparisons using rank sums. **Technometrics**, Taylor & Francis, v. 6, n. 3, p. 241–252, 1964.

EMBRAPA SOLOS. **Sistema brasileiro de classificação de solos**. Rio de Janeiro: Embrapa, 2006.

FONTELLES, M. J. **Bioestatística aplicada à pesquisa experimental**. [S.l.]: São Paulo: Livraria da Física, 2012. v. 2.

GONZÁLEZ-TERUEL, J. D. et al. Design and Calibration of a Low-Cost SDI-12 Soil Moisture Sensor. **Sensors**, v. 19, n. 3, 2019. ISSN 1424-8220. DOI: 10.3390/s19030491. Disponível em: <https://www.mdpi.com/1424-8220/19/3/491>.

GROLEMUND, G.; WICKHAM, H. Dates and Times Made Easy with lubridate. **Journal of Statistical Software**, v. 40, n. 3, p. 1–25, 2011. Disponível em: <https://www.jstatsoft.org/v40/i03/>.

GUJARATI, D. N.; PORTER, D. C. Econometria básica. ed. **Porto Alegre: AMGH**, 2011.

GURGEL, J. Sobre a producao de pescado dos acudes publicos do semi-arido nordeste brasileiro. **VILA, I. e FAGETTI, E. Trabajos presentados al Taller Internacional sobre ecologia y manejo de peces en lagos y embalses. Santiago: COPESCAL/FAO. Doc. Téc**, p. 54–64, 1984.

GUTIERRES, M. I.; NEVES, E. A importância do monitoramento da umidade do solo através de sensors para otimizar a irrigação nas culturas. **ENCICLOPÉDIA BIOSFERA**, v. 18, n. 35, 2021.

GUTIÉRREZ, J. et al. Automated irrigation system using a wireless sensor network and GPRS module. **IEEE transactions on instrumentation and measurement**, Ieee, v. 63, n. 1, p. 166–176, 2013.

HARIYANTO, M. W.; HENDRAWAN, A. H.; RITZKAL, R. Monitoring the environmental temperature of the Arduino assistance engineering faculty using telegram. **Journal of Robotics and Control (JRC)**, v. 1, n. 3, p. 96–101, 2020.

INSTITUTO NACIONAL DE METEOROLOGIA. **Manual de Observações Meteorológicas**. Brasília, 1999.

JAYARAMAN, K. **A statistical manual for forestry research**. Bangcoc, Tailândia: FORSPA, 2000.

KRUSKAL, W. H.; WALLIS, W. A. Use of ranks in one-criterion variance analysis. **Journal of the American statistical Association**, Taylor & Francis, v. 47, n. 260, p. 583–621, 1952.

KUTNER, M. H. et al. **Applied linear statistical models**. [S.l.]: McGraw-hill, 2005.

LABCENTER ELECTRONICS. **Proteus Design Suite**. Yorkshire, England, 1988. Disponível em: <www.labcenter.com>.

LÓPEZ-VARGAS, A. et al. Low-Cost datalogger intended for remote monitoring of solar photovoltaic standalone systems based on Arduino™. **IEEE Sensors journal**, IEEE, v. 19, n. 11, p. 4308–4320, 2019.

LU, T.; CHEN, C. Uncertainty evaluation of humidity sensors calibrated by saturated salt solutions. **Measurement**, Elsevier, v. 40, n. 6, p. 591–599, 2007.

MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. **The annals of mathematical statistics**, JSTOR, p. 50–60, 1947.

MENDES, J. P. P. et al. Calibração de sonda de baixo custo para monitorar umidade em substrato comercial. **Meio Ambiente (Brasil)**, v. 3, n. 1, 2021.

MINIPA. **Proposta Técnica do Medidor de Umidade do solo MV-331**. São Paulo, Brasil, 2024. Disponível em: <https://www.minipa.com.br/images/proposta_tecnica/MV-331-1300-BR.PDF>.

MOURA CAMPOS, H. de et al. Low-cost open-source platform for irrigation automation. **Computers and Electronics in Agriculture**, Elsevier, v. 190, p. 106481, 2021.

NOVELAN, M. S.; AMIN, M. Monitoring system for temperature and humidity measurements with DHT11 sensor using nodeMCU. **International Journal of Innovative Science and Research Technology**, v. 5, n. 10, p. 123–128, 2020.

R CORE TEAM. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2022. Disponível em: <https://www.R-project.org/>.

RAMADAN, K. M. et al. Design and implementation of a low cost photovoltaic soil moisture monitoring station for irrigation scheduling with different frequency domain analysis probe structures. **Computers and Electronics in Agriculture**, Elsevier, v. 148, p. 148–159, 2018.

ROBÓTICA, C. da. **A Casa da Robótica**. 2024. Disponível em: <http://www.casadarobotica.com>. Acesso em: 30 abr. 2024.

ROUX, J. et al. A connected soil humidity and salinity sensor for precision agriculture. In: IEEE. 2018 International Conference on Computational Science and Computational Intelligence (CSCI). [S.l.: s.n.], 2018. p. 1028–1031.

RP5. **Estação Meteorológica Recife (Aeroporto) METAR=SBRF**. 2022. Disponível em: <http://rp5.am/Arquivo_de_tempo_em_Recife_(aeroporto),_METAR>. Acesso em: 31 jul. 2022.

SABOYA, L. M. F. et al. MÉTODOS DAS CLASSIFICAÇÕES CLIMÁTICAS DE THORNTWAITE E KÖPPEN PARA RECIFE–PE, BRASIL. **RECIMA21-Revista Científica Multidisciplinar-ISSN 2675-6218**, v. 2, n. 8, e28575–e28575, 2021.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, v. 52, n. 3/4, p. 591–611, 1965.

SHARMILA, F. M. et al. IoT based smart window using sensor Dht11. In: IEEE. 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). [S.l.: s.n.], 2019. p. 782–784.

THE ARDUINO TEAM. **Arduino Software IDE**. Ivrea, Italy, 2005. Disponível em: <https://www.arduino.cc/>.

WICKHAM, H. Reshaping Data with the reshape Package. **Journal of Statistical Software**, v. 21, n. 12, p. 1–20, 2007. Disponível em: <http://www.jstatsoft.org/v21/i12/>.

APÊNDICE A

CÓDIGOS ARDUINO

A.1. CÓDIGO PRINCIPAL

```
1  //Bibliotecas
2  #include <dht.h>
3  #include <SPI.h>
4  #include <SD.h>
5  #include <Wire.h>
6  #include <RtcDS3231.h> // Biblioteca relógio DS3231
7  #include <MicroLCD.h>
8  #include <EEPROM.h>
9  //DEFINIÇÕES
10 #define PinMOP 7     // Pino modo de operação (Uso do cartão ou não)
11 #define espacoEEPROM 1000 // Espaço da EEPROM dedicado a armazenar o tempo de leitura
12 //dos sensores configurado
13 #define PinBotTime1 2 // Pino do botão para configurar tempo de leitura dos sensores
14 #define DHTPIN 10  //Pino do DHT
15 #define DHTTYPE DHT11 //Modelo do DHT
16
17 // VARIÁVEIS/CONSTANTES GLOBAIS
18 const int PinSD_CS = 4;  // Pino CS do módulo Cartão SD
19 unsigned long Dt_Medida;
20 unsigned long Dt_Anterior = 0;
21 int PinSolo[] = {A0, A1, A2, A3, A6, A7}; // Vetor com Pinos dos seis sensores do solo
22 int MO; //int MO_Inicial;
23 int ReadSolo[6]; //Leitura do sensor do solo 0 a 1023 (analógico)
24 float USolo[6];  //Umidade do solo
25 int Config1; // Para uso no botão configuração do tempo de leitura
26 bool BotTime1Atu = false;
27 bool BotTime1Ant = false;
28 byte hiByte; //Para uso o armazenamento de tempo de leitura na memória eeprom
29 byte loByte; //Para uso o armazenamento de tempo de leitura na memória eeprom
30 String Descri; //Variável para guardas descrição da escolha do tempo de leitura
31 unsigned long delay1 = 0;
32 unsigned long delay2 = 0;
33 int MO_Ant; //Configura o modo de operação (Com ou sem SDCard)
34 File MyFile;
35 bool cartaoOk = true;
36
37 // OBJETOS
38
39 dht DHT;
```

```
40  RtcDS3231<TwoWire> Rtc(Wire); // Cria o objeto Rtc - relógio
41  RtcDateTime t; //Para instante atual do RTC
42  RtcDateTime tempoatual;
43
44  // CONFIGURANDO O TIPO DE LCD UTILIZADO
45
46  LCD_SSD1306 lcd; /* para módulo SSD1306 OLED */
47
48  void setup() {
49    Serial.begin(9600);
50    //Serial.println("Início setup");
51    //dht.begin(); //Inicia o dht
52    //SPI.begin();  // Inicia a comunicação SPI - Para o Sd card
53    Rtc.Begin(); //Inicia o objeto criado rtc;
54    lcd.begin(); //Inicia o lcd
55
56  /* PARA AJUSTAR A HORA DO RTC
57      Se for necessário ajustar o relógio do RTC
58      carrega o sketch com este trecho. Após o ajuste,
59      deve-se comentar e carregar o sketch com este trecho comentado*/
60   //   tempoatual = RtcDateTime(__DATE__,__TIME__); // Ajusta o horário.
61   //   Rtc.SetDateTime(tempoatual);
62
63  //Para armazenar informações do tempo de leitura na eeprom
64    hiByte = EEPROM.read(0);
65    loByte = EEPROM.read(1);
66    Config1 = word(hiByte, loByte);
67
68  // Configuração dos pinos em uso
69    pinMode(PinBotTime1, INPUT); // configura o pino para entrada da configuração do tempo
70    // pinMode(PinMOP, INPUT);
71    pinMode(PinSD_CS, OUTPUT);
72  //  for(int n=0; n<6; n++){  // Pinos para entrada leitura sensores solo
73  //    pinMode(PinSolo[n], INPUT);
74  //  }
75
76    // Variável que armazena o instante atual - RTC
77    // Será usado em muitas funções chamdas no setup
78   t  = Rtc.GetDateTime();
79
80   //Verifica estado do botão para modo de operação (se usa SD ou não)
81    MO = Botao();
82    MO_Ant = MO;
83    //Serial.println(MO);
84
85  // Verifica se o botão está pressionado para uso de SD e inicia módulo SD
86    if (MO == 0) { //Caso em que o SD entra em uso para gravação e é feita vericação no Setput
87      while(!SD.begin(PinSD_CS) && MO == 0) { //verifica se sd está inserido e OK
88        Falha_G_SD(); // Função que apresenta no LCD informação de falha do SD Card
89        delay(2000);
90      } // Se SD Ok prosegue para as linhas posteriores
91      logo(); //Função que apresenta logo no LCD
92      delay(2000);
93      Sd_NotT(); //Função que apresenta mensagem de SD_OK no LCD
94      delay(3000);
95    }
96
97    // Verifica se o botão não está pressionado para não uso de SD
98    if (MO == 1) { //Caso em que o SD não está sendo usado
99      logo();
100     delay(2000);
101     Falha_G_SD();
102     delay(2000);
103   }
```

```
104 }
105 void loop() {
106   t = Rtc.GetDateTime();
107   MO = Botao();
108
109 BotTime1Atu = digitalRead(PinBotTime1);
110
111   if(BotTime1Atu && !BotTime1Ant){
112     Config1 = Config1 + 1;
113     EEPROM.write(0, highByte(Config1));
114     EEPROM.write(1, lowByte(Config1));
115     if(Config1 > 5){ // Aqui aumenta o número de configurações e as incluem no switch abaixo.
116       Config1 = 1;
117       EEPROM.write(0,highByte(Config1));
118       EEPROM.write(1,lowByte(Config1));
119     }
120   }
121
122   BotTime1Ant = BotTime1Atu; //Muda o estado do Botão para Config
123
124   if(Config1 == 1){
125     Descri = "g:5seg";
126     Dt_Medida = 5000;
127   }
128   if(Config1 == 2){
129     Descri = "g:1min";
130     Dt_Medida = 60000;
131   }
132   if(Config1 == 3){
133     Descri = "g:30min";
134     Dt_Medida = 1800000;
135   }
136   if(Config1 == 4){
137     Descri = "g:1h";
138     Dt_Medida = 3600000;
139   }
140    if(Config1 == 5){
141     Descri = "g:6h";
142     Dt_Medida = 62100000;
143   }
144
145   // CÁLCULO DE MÉDIA DO SENSOR (10 LEITURAS)
146   int ReadsSolo[10]; // Vetor para armazenar leituras (trocar o tamanho do vetor para
147   //aumentar leituras para média)
148   int SumReadsSolo = 0; // Variável para armezar soma
149   float NLeituras = (float)(sizeof(ReadsSolo) / sizeof(ReadsSolo[0]));
150
151   for(int n = 0; n < 6; n++){
152     for (int i = 0; i <= (NLeituras - 1); i++) { // efetuando as leituras
153       ReadsSolo[i] = analogRead(PinSolo[n]); //Leitura
154       SumReadsSolo = SumReadsSolo + ReadsSolo[i]; // soma
155     }
156
157   ReadSolo[n] = SumReadsSolo / NLeituras; // Cálculo da média
158   USolo[n] = 100.0 * (1.0 - (ReadSolo[n] / 1023.0)); // Umidade % (Linear)
159   SumReadsSolo = 0;
160   }
161
162    if (Dt_Anterior != Dt_Medida) {
163     if (MO == 0) {
164       Sd_OK(); // Função para mostrar informações quando houver cartão SD
165     if (MO == 1) {
166       Inform(); // Função para mostrar informações quando não houver cartão SD
167     }
```

```
168       Dt_Anterior = Dt_Medida;
169     }
170        int chk = DHT.read11(DHTPIN);
171
172     // CONFIGURAÇÃO DELAY COM POTENCIÔMETRO - SUBSTITUIR PELO BOTÃO
173     if (millis() - delay1 >= Dt_Medida) {
174       delay1 = millis();
175       if(MO == 1){
176         Inform();
177         Serial.print(t.Day());
178         Serial.print("/");
179         Serial.print(t.Month());
180         Serial.print("/");
181         Serial.print(t.Year());
182         Serial.print(";");
183         Serial.print(t.Hour());
184         Serial.print(":");
185         Serial.print(t.Minute());
186         Serial.print(":");
187         Serial.print(t.Second());
188         Serial.print(";");
189         Serial.print(DHT.temperature);
190         Serial.print(";");
191         Serial.print(DHT.humidity);
192         Serial.print(";");
193         Serial.print(ReadSolo[0]);
194         Serial.print(";");
195         Serial.print(ReadSolo[1]);
196         Serial.print(";");
197         Serial.print(ReadSolo[2]);
198         Serial.print(";");
199         Serial.print(ReadSolo[3]);
200         Serial.print(";");
201         Serial.print(ReadSolo[4]);
202         Serial.print(";");
203         Serial.print(ReadSolo[5]);
204         Serial.print(";");
205         Serial.print(USolo[0]);
206         Serial.print(";");
207         Serial.print(USolo[1]);
208         Serial.print(";");
209         Serial.print(USolo[2]);
210         Serial.print(";");
211         Serial.print(USolo[3]);
212         Serial.print(";");
213         Serial.print(USolo[4]);
214         Serial.print(";");
215         Serial.println(USolo[5]);
216
217       }
218       if(MO == 0){
219         if(SD.begin(PinSD_CS) == false){
220           Falha_G_SD();
221           Serial.println("Falha SD!");
222         }
223         if(SD.begin(PinSD_CS) == true){
224           Sd_OK();
225           MyFile = SD.open("Mydata.csv", FILE_WRITE);
226           if(MyFile){
227             Serial.println("Gravando!");
228             Sd_OK();
229             MyFile.print(t.Day());
230             MyFile.print("/");
231             MyFile.print(t.Month());
```

```
232             MyFile.print("/");
233             MyFile.print(t.Year());
234             MyFile.print(";");
235             MyFile.print(t.Hour());
236             MyFile.print(":");
237             MyFile.print(t.Minute());
238             MyFile.print(":");
239             MyFile.print(t.Second());
240             MyFile.print(";");
241             MyFile.print(DHT.temperature);
242             MyFile.print(";");
243             MyFile.print(DHT.humidity);
244             MyFile.print(";");
245             MyFile.print(ReadSolo[0]);
246             MyFile.print(";");
247             MyFile.print(ReadSolo[1]);
248             MyFile.print(";");
249             MyFile.print(ReadSolo[2]);
250             MyFile.print(";");
251             MyFile.print(ReadSolo[3]);
252             MyFile.print(";");
253             MyFile.print(ReadSolo[4]);
254             MyFile.print(";");
255             MyFile.print(ReadSolo[5]);
256             MyFile.print(";");
257             MyFile.print(USolo[0]);
258             MyFile.print(";");
259             MyFile.print(USolo[1]);
260             MyFile.print(";");
261             MyFile.print(USolo[2]);
262             MyFile.print(";");
263             MyFile.print(USolo[3]);
264             MyFile.print(";");
265             MyFile.print(USolo[4]);
266             MyFile.print(";");
267             MyFile.print(USolo[5]);
268             MyFile.println(";");
269             MyFile.close();
270           } else{
271             Serial.println("Nada de gravação");
272           }
273         }
274       }
275     }
276
277   if (MO_Ant != MO) {
278       if (MO == 1) {
279         Inform();
280       }
281       if (MO == 0) {
282         Sd_OK();
283       }
284       MO_Ant = MO;
285     }
286
287     delay(100);
288   }
```

A.2. FUNÇÕES AUXILIARES

A.2.1. Função para controle do modo de operação

```
bool Botao() {
#define TempoDebounce 50
  int DelayBotao = 0;
  int EstadoBotao;
  if ((millis() - DelayBotao) > TempoDebounce) {
    EstadoBotao = digitalRead(PinMOP);
    DelayBotao = millis();
  }
  return EstadoBotao;
}
```

A.2.2. Função para apresentar data e hora do RTC no LCD

```
void TimeTLCD(){
  lcd.print(t.Day());
  lcd.print("/");
  lcd.print(t.Month());
  lcd.print("/");
  lcd.print(t.Year());
  lcd.print(" ");
  lcd.print(t.Hour());
  lcd.print(":");
  lcd.print(t.Minute());
  lcd.print(":");
  lcd.println(t.Second());
}
```

A.2.3. Funções para apresentar falha do cartão SD

```
void Falha_G_SD() {
  lcd.clear();
  lcd.setCursor(10, 1);
  lcd.setFontSize(FONT_SIZE_LARGE);
  lcd.println("PPGBEA-UFRPE");
  lcd.setCursor(10, 4);
  lcd.setFontSize(FONT_SIZE_SMALL);
  TimeTLCD();
  lcd.setFontSize(FONT_SIZE_SMALL);
  lcd.println("SD failure!");
}
void Sd_NotT() {
  lcd.clear();
  lcd.setCursor(10,4);
  lcd.setFontSize(FONT_SIZE_LARGE);
  lcd.println("SD not tested!");
```

```
17    delay(600);
18  }
```

A.2.4. Função para display LCD

```
1   void Inform() {
2     lcd.clear();
3     lcd.setFontSize(FONT_SIZE_MEDIUM);
4     lcd.setCursor(10, 0);
5     lcd.println("PPGBEA-UFRPE");
6     lcd.setFontSize(FONT_SIZE_SMALL);
7     lcd.setCursor(2, 2);
8     lcd.println("Option: No SD Card!");
9     lcd.setCursor(2, 3);
10    TimeTLCD();
11    lcd.setFontSize(FONT_SIZE_SMALL);
12    lcd.setCursor(2, 4);
13    lcd.print(DHT.temperature);
14    lcd.print(";");
15    lcd.print(DHT.humidity);
16    lcd.print(";");
17    lcd.println(Descri);
18    lcd.setCursor(2, 6);
19    lcd.print("U1=");
20    lcd.print(round(USolo[0]));
21    lcd.print(";U2=");
22    lcd.print(round(USolo[1]));
23    lcd.print(";U3=");
24    lcd.println(round(USolo[2]));
25    lcd.setCursor(2,8);
26    lcd.print("U4=");
27    lcd.print(round(USolo[3]));
28    lcd.print(";U5=");
29    lcd.print(round(USolo[4]));
30    lcd.print(";U6=");
31    lcd.println(round(USolo[5]));
32  }
```

A.2.5. Função para logotipo

```
1   void logo() {
2     lcd.clear();
3     lcd.setCursor(10,4);
4     lcd.setFontSize(FONT_SIZE_LARGE);
5     lcd.println("PPGBEA-UFRPE");
6     delay(600);
7   }
```

A.2.6. Função para SD Card

```
void Sd_OK() {
  lcd.clear();
  lcd.setFontSize(FONT_SIZE_MEDIUM);
  lcd.setCursor(10, 0);
  lcd.println("PPGBEA-UFRPE");
  lcd.setFontSize(FONT_SIZE_SMALL);
  lcd.setCursor(2, 2);
  lcd.println("Option: SD recording!");
  lcd.setCursor(2, 3);
  TimeTLCD(); //(Aba_FDataHora)
  lcd.setFontSize(FONT_SIZE_SMALL);
  lcd.setCursor(2, 4);
  lcd.print(DHT.temperature);
  lcd.print(";");
  lcd.print(DHT.humidity);
  lcd.print(";");
  lcd.println(Descri);
  lcd.setCursor(2, 6);
  lcd.print("U1=");
  lcd.print(round(USolo[0]));
  lcd.print(";U2=");
  lcd.print(round(USolo[1]));
  lcd.print(";U3=");
  lcd.println(round(USolo[2]));
  lcd.setCursor(2,8);
  lcd.print("U4=");
  lcd.print(round(USolo[3]));
  lcd.print(";U5=");
  lcd.print(round(USolo[4]));
  lcd.print(";U6=");
  lcd.println(round(USolo[5]));
}
```

APÊNDICE B

COMPONENTES ELETRÔNICOS DA ESTAÇÃO

Figura B.1: Diodo Schottky e módulo microSD

(a) Diodo Schottky (b) Módulo para cartão MicroSD

Fonte: (ROBÓTICA, 2024)

Figura B.2: Módulos reguladores de tensão

(a) Módulo Regulador de Tensão Xl6009

(b) Módulo regulador de tensão e carregador de bateria

Fonte: (ROBÓTICA, 2024)

Figura B.3: Botão e ventilador

(a) Botão push do tipo pulsador

(b) Mini cooler

Fonte: (ROBÓTICA, 2024)

APÊNDICE C

CÓDIGOS R

C.1. IMPORTANDO E ORGANIZANDO OS DADOS

O código abaixo corresponde ao arquivo 01-setup.R que importa os dados que serão utilizados e os formata para melhor utilização.

```
library(readr)
library(reshape2)
library(lubridate)
library(xlsx)
library(readxl)
library(ggplot2)
library(dplyr)
library(scales)
library(nortest)
library(corrplot)
library(car)
library(lmtest)
library(dunn.test)
#Import airport station data
#dados_aeroporto <- read_excel("data/dados_aeroporto.xlsx")
dados_aeroporto <- read_excel("data/aeroporto.xlsx",
                          skip = 6)
dados_aeroporto <- dados_aeroporto[,-c(3,4,6:13)]
names(dados_aeroporto) <- c("data","temperatura","umidade_do_ar")
#Formatting the date columnn for a datahora column
dados_aeroporto$datahora <- as.POSIXct(strptime(dados_aeroporto$data,
                     format="%d.%m.%Y %H:%M"),
            format="%Y-%m-%d %H:%M")
#creating hora and data columns with values for hour and date
dados_aeroporto$hora <- format(dados_aeroporto$datahora, format="%H:%M")
dados_aeroporto$data <- format(dados_aeroporto$datahora, format="%Y-%m-%d")
#Creating a dataframe just to when rained in AEROPORTO
rain <- read_excel("data/aeroporto.xlsx",
           skip = 6)
rain <- rain[,c(1,9)]
names(rain) <- c("data","fenomenoatual")
rain$datahora <- as.POSIXct(strptime(rain$data,
                                                format="%d.%m.%Y %H:%M"),
                                          format="%Y-%m-%d %H:%M")
#creating hora and data columns with values for hour and date
```

```
rain$hora <- format(rain$datahora, format="%H:%M")
rain$data <- format(rain$datahora, format="%Y-%m-%d")
#CEMADEN Rain Data
data_cemaden_nov <- read_delim("data/data_cemaden_nov.csv",
                               delim = ";", escape_double = FALSE,
                               col_types = cols(valorMedida = col_double()),
                               locale = locale(decimal_mark = ",",
                                               grouping_mark = ""),
                               trim_ws = TRUE)
data_cemaden_dez <- read_delim("data/data_cemaden_dez.csv",
                               delim = ";", escape_double = FALSE,
                               col_types = cols(valorMedida = col_double()),
                               locale = locale(decimal_mark = ",",
                                               grouping_mark = ""),
                               trim_ws = TRUE)
data_cemaden <- rbind(data_cemaden_nov,data_cemaden_dez)
data_cemaden <- data_cemaden |> filter(nomeEstacao=='Dois Irmãos')
data_cemaden <- data_cemaden[,-c(1:6)]
data_cemaden <- data_cemaden |>
        filter(datahora >ymd_hms("2021-11-25 00:00:00",tz="America/Sao_Paulo")
               &datahora <ymd_hms("2021-12-18 00:00:00",tz="America/Sao_Paulo"))
data_cemaden$datahora <- trunc(data_cemaden$datahora, units="hours")
data_cemaden <- data_cemaden |> group_by(datahora) |>
    summarize(valorMedida=sum(valorMedida))
data_cemaden$datahora <- as.POSIXct(data_cemaden$datahora)
data_cemaden <- data_cemaden |> filter(valorMedida>0)
data_cemaden$cema <- 65
data_cemaden$cema_temp <- 24
#Import data for the single pot test
data_single_pot_test <- read_delim("data/data_single_pot_test.CSV",
                                   delim = ";", escape_double = FALSE,
                                   col_names = FALSE, trim_ws = TRUE)
#Add head, melt and fix date
names(data_single_pot_test)<-c('data','hora','temperatura','umidade_do_ar','solo1',
                          'solo2','solo3','solo4','solo5','solo6',
                          'solo1Relativa','solo2Relativa','solo3Relativa',
                          'solo4Relativa','solo5Relativa','solo6Relativa','nada')
data_single_pot_test <- melt(data_single_pot_test, id.vars = c('data','hora'),
                    measure.vars = c('temperatura','umidade_do_ar','solo1',
                                     'solo2','solo3','solo4','solo5','solo6',
                                     'solo1Relativa','solo2Relativa',
                                     'solo3Relativa','solo4Relativa',
                                     'solo5Relativa','solo6Relativa'),
                    variable.name = "medida",value.name='valor')
data_single_pot_test$data <- as.Date(parse_date_time(data_single_pot_test$data,
                                                     orders="dmy"))
#Import data for the full test
data_full_test <- read_delim("data/data_full_test.CSV",
                                    delim = ";", escape_double = FALSE,
                               col_names = FALSE, trim_ws = TRUE)
#Add head, melt and fix date
names(data_full_test)<-c('data','hora','temperatura','umidade_do_ar','solo1',
                                 'solo2','solo3','solo4','solo5','solo6',
                                 'solo1Relativa','solo2Relativa','solo3Relativa',
                                 'solo4Relativa','solo5Relativa','solo6Relativa',
                          'nada')
data_full_test <- melt(data_full_test, id.vars = c('data','hora'),
                               measure.vars = c('temperatura','umidade_do_ar','solo1',
                                                'solo2','solo3','solo4','solo5',
                                                'solo6','solo1Relativa',
                                                'solo2Relativa','solo3Relativa',
                                                'solo4Relativa','solo5Relativa',
                                                'solo6Relativa'),
                               variable.name = "medida",value.name='valor')
```

```
data_full_test$data <- as.Date(parse_date_time(data_full_test$data,
                                               orders="dmy"))
data_full_test$datahora <- as.POSIXct(strptime(paste(data_full_test$data,
                                               data_full_test$hora),
                                               format="%Y-%m-%d %H:%M"),
                                          format="%Y-%m-%d %H:%M")
#Import the regression test data and renaming
data_regression_test <- read_delim("data/data_regression_test.CSV",
                                   delim = ";", escape_double = FALSE,
                                   trim_ws = TRUE)
names(data_regression_test)<-c("ordem","nf",'data','hora','temperatura',
                               'umidade_do_ar','solo1','solo2','solo3','solo4',
                               'solo5','solo6')
#melting
data_regression_test <- melt(data_regression_test, id.vars = c("ordem",'data',
                                                               'hora'),
                         measure.vars = c('solo1','solo2','solo3','solo4',
                                          'solo5','solo6'),
                         variable.name = "medida",value.name='valor')
#organizing date and hour
data_regression_test$data <- as.Date(parse_date_time(data_regression_test$data,
                                               orders="dmy"))
#Creating a column datahora with date and hour
data_regression_test$datahora <- as.POSIXct(strptime(
        paste(data_regression_test$data,data_regression_test$hora),
        format="%Y-%m-%d %H:%M"),format="%Y-%m-%d %H:%M")
#Creating the blocks that were used in the trial
bloco1 <- c("solo1",'solo2','solo4')
bloco2 <- c("solo3",'solo5','solo6')
#Creating a column with the respective sensor in their blocks
data_regression_test <- mutate(data_regression_test,
                               blocos =ifelse(medida %in% bloco1, "bloco1",
                                              "bloco2"))
#Filtering
data_regression_test <- filter(data_regression_test,datahora> "2023-05-18 17:56:00")
#Organizing by block
data_regression_test <- arrange(data_regression_test,blocos)
#Creating the column umidade_real with the humidity measured by the analog hygrometer
data_regression_test$umidade_real <- c(rep(rep(c(20,35,50,60,0),c(6,6,6,6,6)),times=3),
                                   rep(rep(c(0,20,35,50,60),c(6,6,6,6,6)),times=3))
```

C.2. DADOS DE TEMPERATURA

O código abaixo corresponde ao arquivo 02-temperature.R, que faz a organização dos dados voltados para temperatura, com os comandos para os testes utilizados, geração das tabelas e gráficos.

```
#Filtering the data only for the temperature
temperatura <- filter(data_full_test,medida %in% c('temperatura')&
                              data < dmy("18-12-2021"))

#Truncate the hora column to only show the hour digit so it can bet summarized
temperatura$hora <- format(trunc(temperatura$datahora, units="hours"),
                           format="%H:%M")
#Grouping by data and hour, them summarizing the valor columnn
temperatura <- temperatura %>% group_by(data,hora) %>%
```

```
        summarize(valor=valor[1])
#Creating the datahora columnn once again for plotting purporses
temperatura$datahora <- as.POSIXct(strptime(paste(temperatura$data,
                                                  temperatura$hora),
                                            format="%Y-%m-%d %H:%M"),
                                   format="%Y-%m-%d %H:%M")
#Joining temperatura and dados_aeroporto by datahora column
temperatura_final <- inner_join(temperatura,dados_aeroporto[,-c(1,3,5)],
                                by=join_by(datahora))
#Summarizing aeroporto to show minimum, average and max temperature by day
summary_aeroporto <- temperatura_final %>% group_by(data) %>%
        summarize(min=min(temperatura),media=round(mean(temperatura),digits=2),
                  max=max(temperatura))
summary_aeroporto$station <- rep("aeroporto",each=24)
#Summarizing station to show minimum, average and max temperature by day
summary_ufrpe <- temperatura_final %>% group_by(data) %>%
        summarize(min=min(valor),media=round(mean(valor),digits=2),
                  max=max(valor))
summary_ufrpe$station <- rep("ufrpe",each=24)
#Creating a xlsx fuke for using in tables for summary
write.xlsx(inner_join(summary_ufrpe,summary_aeroporto,
                      by=join_by(data)),"results/summary_temperature.xlsx" )
#putting together both dataframes and melting
summary_final <- rbind(summary_aeroporto,summary_ufrpe)
summary_final <- melt(summary_final, id.vars = c('data','station'),
                      measure.vars = c('min','media','max'),
                      variable.name = "variavel",value.name='valor')
summary_final$station <- as.factor(summary_final$station)
summary_final$variavel <- as.factor(summary_final$variavel)
#Starting tests

#Temperatura Total
tempOverall <- data.frame(c(temperatura$valor,temperatura_final$temperatura),
                          rep(c("ufrpe","aeroporto"),each=550),
                          stringsAsFactors = TRUE)
names(tempOverall) <- c("temp","station")
tempStats <- data.frame(dados=character(0),teste =character(0),pvalor=numeric(0))
#Creating a dataset with all the test results
tempStats <- rbind(tempStats,c("Total",
                               shapiro.test(temperatura_final$valor)$method,
                               shapiro.test(temperatura_final$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Total",shapiro.test(temperatura_final$temperatura)$method,
                     shapiro.test(temperatura_final$temperatura)$p.value))
tempStats <- rbind(tempStats,
                   c("Total",wilcox.test(temp~station,data=tempOverall)$method,
                     wilcox.test(temp~station,data=tempOverall)$p.value))
#Temperatura Minima
tempStats <- rbind(tempStats,
                   c("Mínima",shapiro.test(filter(summary_final,station=='aeroporto'
                                                  & variavel=='min' )$valor)$method,
                   shapiro.test(filter(summary_final,station=='aeroporto' &
                                                 variavel=='min' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Mínima",shapiro.test(filter(summary_final,station=='ufrpe'
                                                  & variavel=='min' )$valor)$method,
                     shapiro.test(filter(summary_final,station=='ufrpe' &
                                                   variavel=='min' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Mínima",
                     wilcox.test(valor~station,
                                 data=filter(summary_final,variavel=='min' ))$method,
                     wilcox.test(valor~station,data=filter(summary_final,
                                                           variavel=='min' ))$p.value))
```

```r
#Temperatura Media
tempStats <- rbind(tempStats,
                   c("Média",shapiro.test(filter(summary_final,station=='aeroporto'
                                                 & variavel=='media' )$valor)$method,
                     shapiro.test(filter(summary_final,station=='aeroporto' &
                                                 variavel=='media' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Média",shapiro.test(filter(summary_final,station=='ufrpe'
                                                 & variavel=='media' )$valor)$method,
                     shapiro.test(filter(summary_final,station=='ufrpe' &
                                                 variavel=='media' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Média",
                     wilcox.test(valor~station,
                                 data=filter(summary_final,variavel=='media' ))$method,
                     wilcox.test(valor~station,data=filter(summary_final,
                                                           variavel=='media' ))$p.value))
#Temperatura Max
tempStats <- rbind(tempStats,
                   c("Máxima",shapiro.test(filter(summary_final,station=='aeroporto'
                                                 & variavel=='max' )$valor)$method,
                     shapiro.test(filter(summary_final,station=='aeroporto' &
                                                 variavel=='max' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Máxima",shapiro.test(filter(summary_final,station=='ufrpe'
                                                 & variavel=='max' )$valor)$method,
                     shapiro.test(filter(summary_final,station=='ufrpe' &
                                                 variavel=='max' )$valor)$p.value))
tempStats <- rbind(tempStats,
                   c("Máxima",
                     wilcox.test(valor~station,
                                 data=filter(summary_final,variavel=='max' ))$method,
                     wilcox.test(valor~station,data=filter(summary_final,
                                                           variavel=='max' ))$p.value))
names(tempStats) <- c("data","method","pvalue")
tempStats$pvalue <- as.numeric(tempStats$pvalue)
write.xlsx(tempStats,"results/stats_temperature.xlsx")


tempStats$pvalue <- round(tempStats$pvalue,digits=4)
str(tempStats$pvalue)

#Graphics
pdf("results/temperature_period1.pdf",width=12,height=8)
ggplot(data=temperatura_final, aes(x=datahora))+
        geom_line(aes(y=valor,colour='UFRPE'),size=1)+
        geom_line(aes(y=temperatura,colour='Aeroporto'),size=1)+
        scale_x_datetime(limits = as.POSIXct(c("2021-11-25 00:00","2021-12-06 00:00")),
                         date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Data", y = "Temperatura",color='Estação')
dev.off()

pdf("results/temperature_period2.pdf",width=12,height=8)
ggplot(data=temperatura_final, aes(x=datahora))+
        geom_line(aes(y=valor,colour='UFRPE'),size=1)+
        geom_line(aes(y=temperatura,colour='Aeroporto'),size=1)+
        scale_x_datetime(limits = as.POSIXct(c("2021-12-06 00:00","2021-12-17 00:00")),
                 date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Data", y = "Temperatura",color='Estação')
dev.off()
```

C.3. DADOS DE UMIDADE DO AR

O código abaixo corresponde ao arquivo 03-air_humidity.R, que faz a organização dos dados voltados para umidade do ar, com os comandos para os testes utilizados, geração das tabelas e gráficos.

```
air_umidity <- filter(data_full_test,
                      medida %in% c('umidade_do_ar')  & data <
                              dmy("18-12-2021"))
#Truncate the hora column to only show the hour digit so it can bet summarized
air_umidity$hora <- format(trunc(air_umidity$datahora, units="hours"),
                           format="%H:%M")
#Grouping by data and hour, them summarizing the valor columnn
air_umidity <- air_umidity %>% group_by(data,hora) %>%
        summarize(valor=valor[1])
#Creating the datahora columnn once again for plotting purporses
air_umidity$datahora <- as.POSIXct(strptime(paste(air_umidity$data,
                                                  air_umidity$hora),
                                            format="%Y-%m-%d %H:%M"),
                                   format="%Y-%m-%d %H:%M")
#Joining temperatura and dados_aeroporto by datahora column
air_umidity_final <- inner_join(air_umidity,dados_aeroporto[,-c(1,2,5)],
                                by=join_by(datahora))
#Summarizing aeroporto to show minimum, average and max temperature by day
summary_aeroporto <- air_umidity_final %>% group_by(data) %>%
        summarize(min=min(umidade_do_ar),
                  media=round(mean(umidade_do_ar),digits=2),
                  max=max(umidade_do_ar))
summary_aeroporto$station <- rep("aeroporto",each=24)
#Summarizing station to show minimum, average and max temperature by day
summary_ufrpe <- air_umidity_final %>% group_by(data) %>%
        summarize(min=min(valor),media=round(mean(valor),digits=2),
                  max=max(valor))
summary_ufrpe$station <- rep("ufrpe",each=24)
#Creating a xlsx fuke for using in tables for summary
write.xlsx(inner_join(summary_ufrpe,summary_aeroporto,
                      by=join_by(data)),"results/summary_humidity.xlsx" )
#putting together both dataframes and melting
summary_final <- rbind(summary_aeroporto,summary_ufrpe)
summary_final <- melt(summary_final, id.vars = c('data','station'),
                      measure.vars = c('min','media','max'),
                      variable.name = "variavel",value.name='valor')

summary_final$station <- as.factor(summary_final$station)
summary_final$variavel <- as.factor(summary_final$variavel)
#Starting tests
#Temperatura Total
humidityOverall <- data.frame(c(air_umidity$valor,air_umidity_final$umidade_do_ar),
                          rep(c("ufrpe","aeroporto"),each=550),
                          stringsAsFactors = TRUE)
names(humidityOverall) <- c("temp","station")
humidityStats <- data.frame(dados=character(0),teste =character(0),pvalor=numeric(0))
#Creating a dataset with all the test results
humidityStats <- rbind(humidityStats,c("Total",
                               shapiro.test(air_umidity_final$valor)$method,
```

```
                                shapiro.test(air_umidity_final$valor)$p.value))
humidityStats <- rbind(humidityStats,c("Total",
                                shapiro.test(air_umidity_final$umidade_do_ar)$method,
                                shapiro.test(air_umidity_final$umidade_do_ar)$p.value))
humidityStats <- rbind(humidityStats,c("Total",wilcox.test(temp~station,data=humidityOverall)$method,
                                wilcox.test(temp~station,data=humidityOverall)$p.value))
#Temperatura Minima
humidityStats <- rbind(humidityStats,
                    c("Mínima",shapiro.test(filter(summary_final,station=='aeroporto' &
                                                     variavel=='min' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='aeroporto' &
                                                   variavel=='min' )$valor)$p.value))
humidityStats <- rbind(humidityStats,
                    c("Mínima",shapiro.test(filter(summary_final,station=='ufrpe' &
                                                     variavel=='min' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='ufrpe' &
                                                   variavel=='min' )$valor)$p.value))
humidityStats <- rbind(humidityStats,c("Mínima",wilcox.test(valor~station,
                                                 data=filter(summary_final,
                                                             variavel=='min' ))$method,
                                wilcox.test(valor~station,data=filter(summary_final,
                                                                      variavel=='min' ))$p.value))
#Temperatura Media
humidityStats <- rbind(humidityStats,
                    c("Média",shapiro.test(filter(summary_final,station=='aeroporto' &
                                                    variavel=='media' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='aeroporto' &
                                                   variavel=='media' )$valor)$p.value))
humidityStats <- rbind(humidityStats,
                    c("Média",shapiro.test(filter(summary_final,station=='ufrpe' &
                                                     variavel=='media' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='ufrpe' &
                                                   variavel=='media' )$valor)$p.value))
humidityStats <- rbind(humidityStats,c("Média",wilcox.test(valor~station,
                                                   data=filter(summary_final,
                                                               variavel=='media' ))$method,
                                wilcox.test(valor~station,data=filter(summary_final,
                                                                      variavel=='media' ))$p.value))
#Temperatura Max
humidityStats <- rbind(humidityStats,
                    c("Máxima",shapiro.test(filter(summary_final,station=='aeroporto' &
                                                     variavel=='media' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='aeroporto' &
                                                   variavel=='max' )$valor)$p.value))
humidityStats <- rbind(humidityStats,
                    c("Máxima",shapiro.test(filter(summary_final,station=='ufrpe' &
                                                     variavel=='max' )$valor)$method,
                      shapiro.test(filter(summary_final,station=='ufrpe' &
                                                   variavel=='max' )$valor)$p.value))
humidityStats <- rbind(humidityStats,c("Máxima",wilcox.test(valor~station,
                                                 data=filter(summary_final,
                                                             variavel=='max' ))$method,
                                wilcox.test(valor~station,data=filter(summary_final,
                                                                      variavel=='max' ))$p.value))
names(humidityStats) <- c("data","method","pvalue")
humidityStats$pvalue <- as.numeric(humidityStats$pvalue)
write.xlsx(humidityStats,"results/stats_humidity.xlsx")

#Graphics
pdf("results/humidity_period1.pdf",width=12,height=8)
ggplot(data=air_umidity_final, aes(x=datahora))+
        geom_line(aes(y=valor,colour='UFRPE'),size=1)+
        geom_line(aes(y=umidade_do_ar,colour='Aeroporto'),size=1)+
        scale_x_datetime(limits = as.POSIXct(c("2021-11-25 00:00","2021-12-06 00:00")),
```

```
                        date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Data", y = "Umidade do ar (Relativa)",color='Estação')
dev.off()

pdf("results/humidity_period2.pdf",width=12,height=8)
ggplot(data=air_umidity_final, aes(x=datahora))+
        geom_line(aes(y=valor,colour='UFRPE'),size=1)+
        geom_line(aes(y=umidade_do_ar,colour='Aeroporto'),size=1)+
        scale_x_datetime(limits = as.POSIXct(c("2021-12-06 00:00","2021-12-17 00:00")),
                        date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Data", y = "Umidade do ar (Relativa)",color='Estação')
dev.off()
```

C.4. ANALISANDO TEMPERATURA E UMIDADE DO AR

O código abaixo corresponde ao arquivo 04-Temperature_Air.R que faz diversas análises com os dados de temperatura e umidade do ar.

```
temperatura_umidade <- inner_join(air_umidity_final[,-c(1,2)],
                                  temperatura_final[,-c(1,2)],
                                  by=join_by(datahora))
names(temperatura_umidade) <- c("umidade_ufrpe","datahora",
                                "umidade_aeroporto","temperatura_ufrpe",
                                "temperatura_aeroporto")

## Filtering rain dataframe to show the periods that rained in Aeroporto
rain <- filter(rain,is.na(fenomenoatual)==FALSE)
rain <- rain |> select(datahora,fenomenoatual)
###Creating a excel table with information about rainy days
write.xlsx(rain,'results/rainydays.xlsx')
## Creating constant values for chuva to be used in humidity and temperature
rain$chuva <- 70
rain$chuva_temp <- 25
## Executing normality tests
temperatura_umidade <- temperatura_umidade[,c(2,1,3,4,5)]
for (i in 2:5){
        print(shapiro.test(unlist(as.vector(temperatura_umidade[,c(i)]))))
}
correlations <- cor(temperatura_umidade[,-c(1)])
pdf("results/correlations.pdf",width=12,height=8)
corrplot(correlations,type='lower',addCoef.col = 'white',method='square',
         tl.srt = 45, order='AOE',diag=T,tl.cex = 2,number.cex=2,tl.pos = 'd')
dev.off()
#Plot graphs with indication of rainy days
temperatura_chuva <- left_join(temperatura_umidade,
                               rain[,c(1,3,4)],
                               by=join_by(datahora))
temperatura_chuva <- left_join(temperatura_chuva,data_cemaden,
                              by=join_by(datahora))

```

```
pdf("results/humidity_rain_period1.pdf",width=12,height=8)
ggplot(data=temperatura_chuva, aes(x=datahora))+
        geom_line(aes(y=umi_apt,colour='Aeroporto'),size=1)+
        geom_line(aes(y=umi_sta,colour='UFRPE'),size=1)+
        geom_point(aes(y=chuva,colour='Chuva-Aero'),size=3)+
        geom_point(aes(y=cema,colour='Chuva-DI'),size=3)+
        scale_x_datetime(limits = as.POSIXct(c("2021-11-25 00:00","2021-12-06 00:00")),
                         date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
              labs(x="Hora", y = "Umidade",color='Variável')
dev.off()
pdf("results/humidity_rain_period2.pdf",width=12,height=8)
ggplot(data=temperatura_chuva, aes(x=datahora))+
        geom_line(aes(y=umi_apt,colour='Aeroporto'),size=1)+
        geom_line(aes(y=umi_sta,colour='UFRPE'),size=1)+
        geom_point(aes(y=chuva,colour='Chuva-Aero'),size=3)+
        geom_point(aes(y=cema,colour='Chuva-DI'),size=3)+
        scale_x_datetime(limits = as.POSIXct(c("2021-12-06 00:00","2021-12-17 00:00")),
                         date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
              theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Umidade",color='Variável')
dev.off()
pdf("results/temperature_rain_period1.pdf",width=12,height=8)
ggplot(data=temperatura_chuva, aes(x=datahora))+
        geom_line(aes(y=temp_apt,colour='Aeroporto'),size=1)+
        geom_line(aes(y=temp_sta,colour='UFRPE'),size=1)+
        geom_point(aes(y=chuva_temp,colour='Chuva-Aero'),size=3)+
        geom_point(aes(y=cema_temp,colour='Chuva-DI'),size=3)+
        scale_x_datetime(limits = as.POSIXct(c("2021-11-25 00:00","2021-12-06 00:00")),
                         date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Temperatura",color='Variável')
dev.off()
pdf("results/temperature_rain_period2.pdf",width=12,height=8)
ggplot(data=temperatura_chuva, aes(x=datahora))+
        geom_line(aes(y=temp_apt,colour='Aeroporto'),size=1)+
        geom_line(aes(y=temp_sta,colour='UFRPE'),size=1)+
        geom_point(aes(y=chuva_temp,colour='Chuva-Aero'),size=3)+
        geom_point(aes(y=cema_temp,colour='Chuva-DI'),size=3)+
        scale_x_datetime(limits = as.POSIXct(c("2021-12-06 00:00","2021-12-17 00:00")),
                         date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
              theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Temperatura",color='Variável')
dev.off()
```

C.5. HIGRÔMETROS EM UM ÚNICO POTE

O código abaixo corresponde ao arquivo 05-moisture_single_pot.R que executa organiza os dados para os testes estatísticos envolvendo os higrômetros em pote único.

```
sensores <- c('solo1Relativa','solo2Relativa','solo3Relativa',
              'solo4Relativa','solo5Relativa','solo6Relativa')

moisture_single <- filter(data_single_pot_test,
                medida %in% sensores)
moisture_single$datahora <- as.POSIXct(strptime(paste(moisture_single$data,
                                                      moisture_single$hora),
                                                format="%Y-%m-%d %H:%M"),
                                       format="%Y-%m-%d %H:%M")
moisture_single <- moisture_single %>% group_by(datahora,medida) %>%
        summarize(valor=valor[1])
#Creates file stats_moisture with the information about the tests
sink("results/stats_moisture.txt")
#Performs Shapiro-Wilk test
for (i in sensores){
        cat("Para sensor ",i)
        print(shapiro.test(filter(moisture_single,sensores==i)$valor))
}
#Peforms Kruskal Wallis Test
kruskal.test(valor~medida,data=moisture_single)
dunn.test(x=moisture_single$valor,g=moisture_single$medida,list=TRUE)
sink()
pdf("results/moisture_single.pdf",width=12,height=8)
ggplot(data=moisture_single, aes(x=datahora, y=valor,group=medida))+
        geom_line(aes(color=medida),size=2)+
        theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Umidade do solo",color='Sensor')+
        geom_point()
dev.off()
```

C.6. HIGRÔMETROS EM POTES DIFERENTES

O código abaixo corresponde ao arquivo 06-moisture_all.R que analisa os dados dos higrômetros quando estes foram colocados em diferentes potes e deixados para funcionar.

```
sensores <- c('solo1Relativa','solo2Relativa','solo3Relativa',
              'solo4Relativa','solo5Relativa','solo6Relativa')

moisture_single <- filter(data_singl#Defining sensors that will be used in filtering
sensores <- c('solo1Relativa','solo2Relativa','solo3Relativa',
              'solo4Relativa','solo5Relativa','solo6Relativa')

moisture_pots <-  filter(data_full_test,
                         medida %in% sensores &
                                 data < dmy("03-12-2021"))
moisture_pots$hora <- format(trunc(as.POSIXct(moisture_pots$datahora,
                                              format="%Y-%m-%d %H:%M"),
                                   units="hours"),format="%H:%M")
moisture_pots$datahora <- as.POSIXct(strptime(paste(moisture_pots$data,
                                                    moisture_pots$hora),
                                              format="%Y-%m-%d %H:%M"),
                                     format="%Y-%m-%d %H:%M")
#Summarizing
moisture_pots <- moisture_pots %>% group_by(datahora,medida) %>%
```

```
        summarize(valor=mean(valor))
pdf("results/moisture_all.pdf",width=12,height=8)
ggplot(data=moisture_pots, aes(x=datahora, y=valor,group=medida))+
        geom_line(aes(color=medida),size=1)+
        theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Umidade do solo",color='Sensor')+
        scale_x_datetime(limits = as.POSIXct(c("2021-11-25 00:00","2021-12-01 00:00")),
                 date_breaks = "1 day", date_labels = "%d/%m")+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')
dev.off()
```

C.7. TESTE DOS HIGRÔMETROS

O código abaixo corresponde ao arquivo 07-moisture_calib.R que faz uma análise dos dados referente à tentativa de calibração dos sensores utilizando um higrômetro analógico.

```
#Creating graphics
pdf("results/moisture_calib_test.pdf",width=12,height=8)
ggplot(data=data_regression_test,
       aes(x=datahora, y=valor,group=medida))+
        geom_point(aes(color=medida),size=3)+
        scale_y_continuous(breaks=c(0,20,35,50,60))+
        theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Hora", y = "Umidade Medida",colour='Sensor')
dev.off()
pdf("results/moisture_calib_sensors.pdf",width=12,height=8)
ggplot(data=data_regression_test, aes(x=umidade_real, y=valor))+
        geom_point(aes(color=medida),size=3)+
        facet_grid(.~medida)+
        scale_x_continuous(breaks=c(0,20,35,50,60))+
        scale_y_continuous(breaks=c(0,20,35,50,60))+
        theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),
              text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Umidade real", y = "Umidade Medida",colour='Sensor')
dev.off()
pdf('results/moisture_calib_reg.pdf')
ggplot(data=data_regression_test, aes(x=umidade_real, y=valor))+
        geom_point(col='blue',size=2)+
        stat_smooth(method = "lm", col = "red")+
        scale_x_continuous(breaks=c(0,20,35,50,60))+
        scale_y_continuous(breaks=c(0,20,35,50,60))+
        theme(text = element_text(size = 20),legend.position = 'bottom')+
        labs(x="Umidade real", y = "Umidade Medida")
dev.off()
#Regression and writing test results
regTotal <- lm(valor~umidade_real,data_regression_test)
sink("results/regression_summary.txt")
shapiro.test(data_regression_test$valor)
shapiro.test(data_regression_test$umidade_real)
summary(regTotal)
sink()
```