UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PATRÍCIA NÁGELA DA SILVA COSTA

DESENVOLVIMENTO DE UM SISTEMA DE HARDWARE E SOFTWARE PARA AQUISIÇÃO, PROCESSAMENTO E ENVIO DE DADOS METEOROLÓGICOS BASEADO EM ARDUINO

São Luís, MA

2019

PATRÍCIA NÁGELA DA SILVA COSTA

DESENVOLVIMENTO DE UM SISTEMA DE HARDWARE E SOFTWARE PARA AQUISIÇÃO, PROCESSAMENTO E ENVIO DE DADOS METEOROLÓGICOS BASEADO EM ARDUINO

Monografia apresentada ao Curso de Engenharia Elétrica da UFMA, como requisito para obtenção parcial do grau de BACHAREL em Engenharia Elétrica.

Orientador: Prof. Dr. Denivaldo Lopes.

São Luís, MA

2019

DESENVOLVIMENTO DE UM SISTEMA DE HARDWARE E SOFTWARE PARA AQUISIÇÃO, PROCESSAMENTO E ENVIO DE DADOS METEOROLÓGICOS BASEADQ EM ARDUINO

PATRÍCIA NÁGELA DA SILVA COSTA

Aprovada em: 07 / 01 / 2019

BANCA EXAMINADORA

Orientador Prof. Dr. Denivaldo Cícero Pavão Lopes
Departamento de Engenharia Elétrica – UFMA

1º Examinador Prof. Dr. Aristófanes Correa Silva
Departamento de Engenharia Elétrica – UFMA

2º Examinador Prof. Dr. Luciano Buonocore
Departamento de Engenharia Elétrica – UFMA

AGRADECIMENTOS

Primeiramente a Deus, por ter me concedido saúde e sabedoria durante o período de execução deste trabalho e por ser meu suporte em todas as dificuldades enfrentadas ao longo deste trajeto.

Aos meus pais, Francisca da Silva Costa e Luis Alves da Costa, por acreditarem em minha capacidade de concluir este curso, por me apoiarem na escolha desta profissão e por todos os esforços que fizeram a fim de me ajudar a chegar nesta etapa tão importante da minha vida.

Aos meus amigos, em especial, Dayanne Hedlen, que esteve comigo em praticamente todos os momentos deste percurso.

Ao professor Denivaldo Lopes, pela orientação, apoio e confiança depositada na implementação deste projeto.

A todos os professores do Departamento de Engenharia Elétrica da UFMA, por todo conhecimento prático e teórico transmitido.

Ao professor Aristófanes Correa, por todas os ensinamentos na disciplina Linguagem de Programação, os quais foram fundamentais para o desenvolvimento deste projeto.

Aos meus orientadores técnicos da ALUMAR, Janio Santos e Sávio Calixto, por todos os conhecimentos transmitidos durante o período de estágio nessa indústria.

A todas as pessoas que eu não mencionei, mas que de alguma forma, seja direta ou indiretamente contribuíram para que eu chegasse até aqui.

“Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.”

(Charles Chaplin)

RESUMO

Uma estação meteorológica é um equipamento que possibilita o monitoramento de variáveis climáticas. É possível coletar informações meteorológicas, tais como, precipitação de água, temperatura ambiente, pressão atmosférica, umidade relativa do ar, velocidade do vento, insolação, umidade do solo, dentre outras. Essas informações são úteis para se fazer uma previsão do tempo e para a caracterização do clima. Sensores eletrônicos específicos podem obter esses dados meteorológicos e transmiti-los para um microcontrolador que pode armazená-los em uma memória persistente ou enviá-los através da internet para um centro de armazenamento. Uma estação meteorológica automática possui vários sensores, por exemplo, um termostato para medir temperatura, barômetro para medir a pressão atmosférica, higrômetro para medir a umidade relativa, pirômetro para medir a insolação e um pluviômetro para medir a precipitação pluviométrica. No mercado brasileiro, há várias estações meteorológicas automáticas disponíveis, entretanto, por um custo elevado. Esse trabalho de conclusão de curso tem por objetivo desenvolver um sistema de *hardware/software* de uma estação meteorológica com as seguintes características: microprocessado, capacidade de armazenamento persistente dos dados coletados, capacidade de comunicação com a Internet e de baixo custo.

Palavras-chave: Estação meteorológica, Microcontrolador, Sensores, Internet das Coisas.

ABSTRACT

A weather station is an equipment that allows the monitoring of climatic variables. It is possible to collect meteorological information, such as precipitation of water, ambient temperature, atmospheric pressure, relative humidity, wind speed, sunshine, soil moisture, among others. This information is useful for forecasting weather and climate characterization. Specific electronic sensors can get these meteorological data and transmit them to a microcontroller that can store them in a persistent memory or send them over the internet to a storage center. An automatic weather station has several sensors, for example, a thermostat for measuring temperature, barometer for measuring atmospheric pressure, hygrometer for measuring relative humidity, pyrometer for measuring sunshine and a rain gauge for measuring rainfall. In the Brazilian market, there are several automatic weather stations available, however, at a high cost. This undergraduate thesis aims to develop a hardware/software system for a weather station with the following characteristics: microprocessor, persistent storage capacity of collected data, ability to communicate with the Internet and low cost.

Keywords: Weather station, Microcontroller, Sensors, Internet of Things.

LISTA DE ABREVIATURAS E SIGLAS

EEPROM	Electrically Erasable Programmable Read-Only Memory
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
INMET	Instituto Nacional de Meteorologia
IOT	Internet of Things
PHP	Hypertext Preprocessor
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
SCL	Serial Clock
SDA	Serial Data
SDO	Serial Data Output
SOC	System on a Chip
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver/Transmitter
UFMA	Universidade Federal do Maranhão
UV	Ultraviolet

LISTA DE FIGURAS E TABELAS

SUMÁRIO

1 INTRODUÇÃO

Os dados utilizados para análise do tempo e do clima são obtidos de várias fontes, dentre elas: estação meteorológica e fotos de satélite. Dentre as grandezas físicas obtidas através de uma estação meteorológica tem-se: precipitação de água, temperatura ambiente, pressão atmosférica, umidade relativa do ar, velocidade do vento, orientação aproximada do vento, insolação, duração da ação do sol, dentre outras.

A estação meteorológica automática é um equipamento formado por um conjunto de sensores eletrônicos que capturam, registram e enviam as informações meteorológicas para um sistema de processamento de dados. As variáveis medidas por esse equipamento são úteis para diversos setores, como por exemplo: geração de energia elétrica, aviação, pecuária, auxílio na definição de políticas públicas, a fim de evitar tragédias causadas por períodos de maior precipitação pluviométrica, bem como também na agricultura, já que algumas das informações coletadas atuam diretamente na produtividade de todas as plantações, de tal forma que através dos dados medidos pode-se gerenciar a temperatura atmosférica, precipitação da água, direção dos ventos e umidade do solo.

Entretanto, no mercado brasileiro, esse equipamento ainda se caracteriza por alto custo, tanto em relação a obtenção quanto a instalação, e, em geral, o acesso às informações coletadas pelo mesmo ainda é feito apenas localmente. As estações meteorológicas automatizadas mais simples estão sendo comercializadas em média pela soma de R$ 2.000,00, esse preço varia de acordo com as funcionalidades presentes.

Vários sensores de baixo custo foram desenvolvidos com o surgimento de tecnologias como o Arduino, Raspberry Pi 3, entre outros. Chips com WiFi embutido compatíveis com essas placas também foram disponibilizados na forma de circuitos de expansão as funcionalidades das mesmas, tornando-se uma das alternativas que possibilita o acesso as variáveis medidas pelos sensores em uma plataforma de armazenamento de dados na nuvem, por exemplo.

Dada a disponibilidade e o baixo custo dos recursos necessários para implementação de um sistema de *hardware/software* capaz de coletar, armazenar e enviar dados meteorológicos, esse trabalho propõe o estudo e desenvolvimento de

um sistema de monitoramento de dados meteorológicos automatizado capaz de auxiliar no controle climático.

1.1 Problemática

As estações meteorológicas fornecem dados para monitoramento de varáveis climáticas e meteorológicas, algumas dessas informações possuem relação direta com a produtividade de diversos setores e podem auxiliar, por exemplo, no controle dos mesmos. Em alguns processos industriais é necessário monitorar o calor, ventos, pragas, chuvas, entre outros, sendo os mesmos fatores que impactam no resultado final de suas operações. Em algumas regiões do Brasil a variável chuva causa inúmeros problemas, principalmente nos períodos de maiores índices pluviométricos, os mesmos afetam a população em geral, tanto os habitantes da periferia quanto os das cidades. Na zona urbana, uma das consequências direta dos períodos chuvosos é o aumento dos engarrafemos, enquanto que nas periferias um dos problemas mais graves enfrentados por seus habitantes são as inundações. Sendo assim, um equipamento que pode auxiliar os políticos, os gestores de indústrias na tomada de decisão quanto aos impactos causados pelas variáveis climáticas, são as estações meteorológicas, entretanto, no Brasil, esse produto ainda se caracteriza por um preço elevado, sendo que alguns só disponibilizam médias diárias, impossibilitando assim, o monitoramento dos dados em tempo real.

1.2 Solução Proposta

Por meio dos diversos sensores eletrônicos e módulos WiFi disponíveis no mercado que são compatíveis com o Arduino e que se caracterizam por um preço baixo, é possível desenvolver um sistema capaz de monitorar algumas variáveis meteorológicas. Sendo assim, essa monografia tem por objetivo desenvolver um sistema de *hardware*/*software* para aquisição, processamento e envio de dados meteorológicos a um baixo custo e que por sua vez encontra-se inserido no contexto de internet das coisas.

1.3 Objetivos

Os objetivos do presente trabalho estão organizados em geral e específicos.

1.3.1 Objetivo Geral

Desenvolver um sistema de *hardware/software* de uma estação meteorológica com as principais características: microprocessado, dotado de comunicação com a internet e de baixo custo.

1.3.2 Objetivos Específicos

Os objetivos específicos são:

- Estudar os diferentes tipos de sensores eletrônicos disponíveis no mercado a fim de verificar algumas características essenciais para inserção desses componentes no projeto, tais como: precisão, faixa de medição, tempo de resposta e valor de corrente padrão;
- Conceber, modelar, implementar e testar o *hardware* de uma estação meteorológica, utilizando como base o Arduino para leitura e tratamento dos dados detectados pelos sensores;
- Realizar testes em laboratório e no campo para verificação da precisão dos componentes utilizados;
- Utilizar o SoC ESP8266 como módulo WiFi para o Arduino, de tal forma a estabelecer comunicação do sistema implementado com a internet;
- Validar o armazenamento dos dados coletados em uma base de dados MySQL, mantido em nuvem utilizando o ThingSpeak;

1.4 Metodologia

As etapas abaixo foram executadas sequencialmente para o desenvolvimento dessa monografia:

1. Levantamento bibliográfico e estudo sobre os assuntos:
   a. Variáveis climáticas;
   b. Sistemas embarcados com a utilização do Arduino;
   c. Sensores de temperatura ambiente, umidade relativa do ar, umidade do solo, precipitação de água, pressão atmosférica, raio ultravioleta, principalmente pelo *datasheet* de cada um desses componentes;
   d. Tecnologia de comunicação WiFi;
   e. Estações meteorológicas automatizadas;

f. Redes de computadores;

g. Linguagem de programação PHP;

h. Banco de dados;

2. Montagem do *hardware*, com a ligação dos sensores e do módulo WiFi na plataforma Arduino baseada no microcontrolador ATmega2560;

3. Desenvolvimento do *software* para aquisição, processamento e envio dos dados meteorológicos, sendo os mesmos enviados para o ThingSpeak e para um banco de dados MySQL;

4. Realização de testes em laboratório e em campo do *hardware* e *software* implementado;

1.5 Apresentação dos Capítulos

Esta monografia está dividida em 5 capítulos, sendo o primeiro uma introdução em relação ao projeto proposto. No capítulo 2 se encontra a revisão e levantamento bibliográfico, no qual são abordadas todas as tecnologias utilizadas para o desenvolvimento do *hardware* e *software* da estação meteorológica implementada. No capítulo 3, é apresentada a solução proposta, no qual são descritas e detalhadas todas as etapas para a implementação do protótipo, incluindo o circuito eletrônico implementado e a explicação dos códigos utilizados e desenvolvidos. No capítulo 4 são mostrados os testes realizados para verificação da eficiência dos sensores e do módulo WiFi ESP8266, apresentando os cenários nos quais os mesmos foram inseridos para realização dos experimentos, sendo também mostrado o custo para a implementação desse projeto. No capítulo 5, é feita a conclusão desse trabalho e apresentadas as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este levantamento e revisão bibliográfica tem como objetivo apresentar a descrição de todas as ferramentas e componentes utilizados para a execução desse projeto.

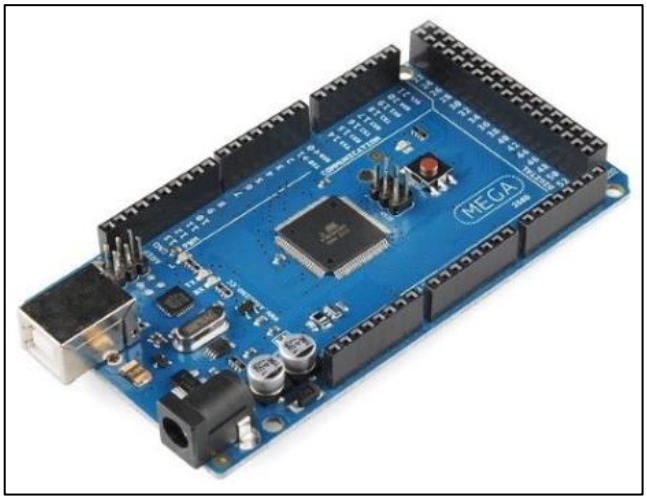
2.1 Arduino

O Arduino é uma plataforma de prototipagem eletrônica *open-source* (MCROBERTS, 2011). Pode ser utilizado para receber sinais de sensores e realizar o processamento das variáveis medidas.

A maior vantagem do Arduino sobre outras plataformas de desenvolvimento é a sua facilidade de utilização. Existe atualmente uma grande quantidade de pessoas utilizando essa plataforma, sendo que a maior parte desses programadores se agrupam em comunidades e estão dispostos a auxiliar outros desenvolvedores, através do compartilhamento tanto do código quanto do diagrama do circuito eletrônico utilizado em seu projeto (MCROBERTS, 2011).

Neste trabalho foi selecionado como componente base o Arduino Mega 2560. Na Figura 1, pode-se observar essa placa.

Figura 1: Arduino MEGA 2560

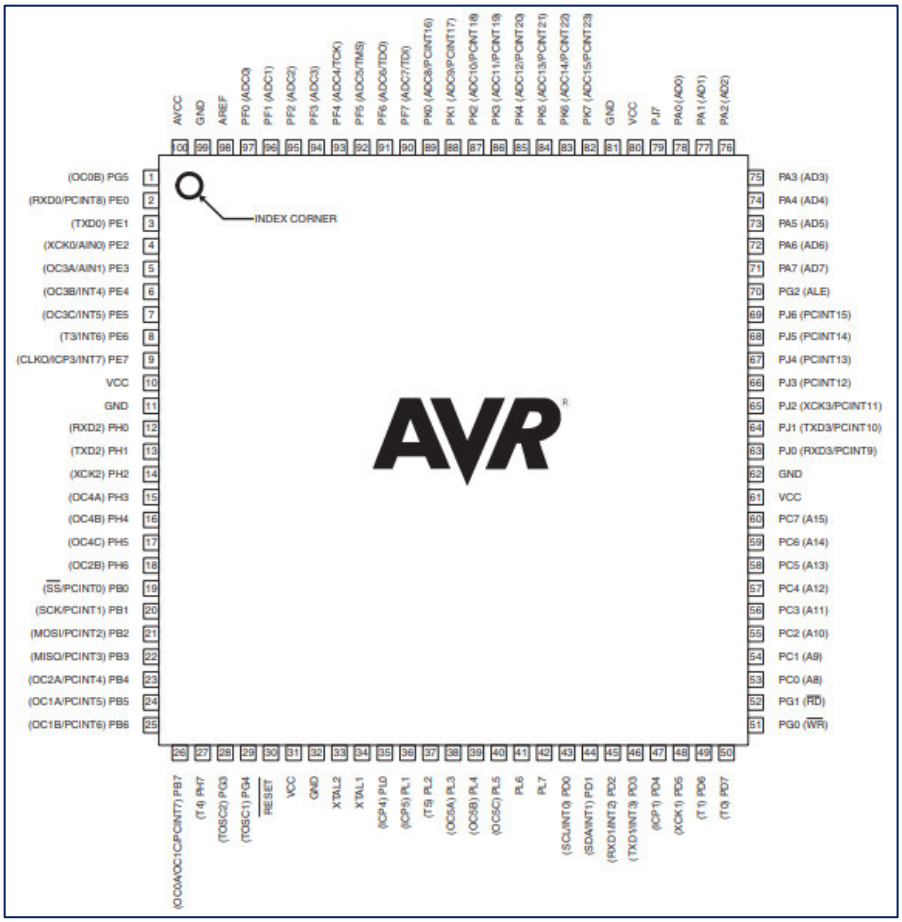


Fonte: http://www.filipeflop.com

Essa plataforma foi escolhida por conter os principais recursos necessários para a implementação da estação meteorológica de baixo custo proposta. A mesma é composta por 16 entradas analógicas, 54 pinos digitais que podem ser utilizados como entrada ou saída, 4 portas de comunicação serial UARTs, especificadas respectivamente como: Serial 0 nos pinos 0 (RX) e 1 (TX), Serial 1 nos pinos 19 (RX) e 18 (TX), Serial 2 nos pinos 17 (RX) e 16 (TX) e a porta Serial 3 nos pinos 15 (RX) e 14 (TX). Pode-se utilizar os pinos disponíveis para interrupções externas a fim de solicitar para o processador que uma tarefa específica seja priorizada e imediatamente executada. Essa placa possui 6 pinos que podem ser utilizados para interrupções externas, são os seguintes: 2, 3, 18, 19, 20 e 21 (ARDUINO, 2018).

O Arduino MEGA 2560 utiliza o microcontrolador ATmega2560. Algumas das principais características desse componente são: memória *flash* de 256 KB, 8 KB de memória SRAM e 4 KB de EEPROM (ATMEL, 2014). A configuração dos pinos do microcontrolador ATmega2560 é ilustrada na Figura 2.

Figura 2: Configuração dos pinos do Microcontrolador ATmega2560

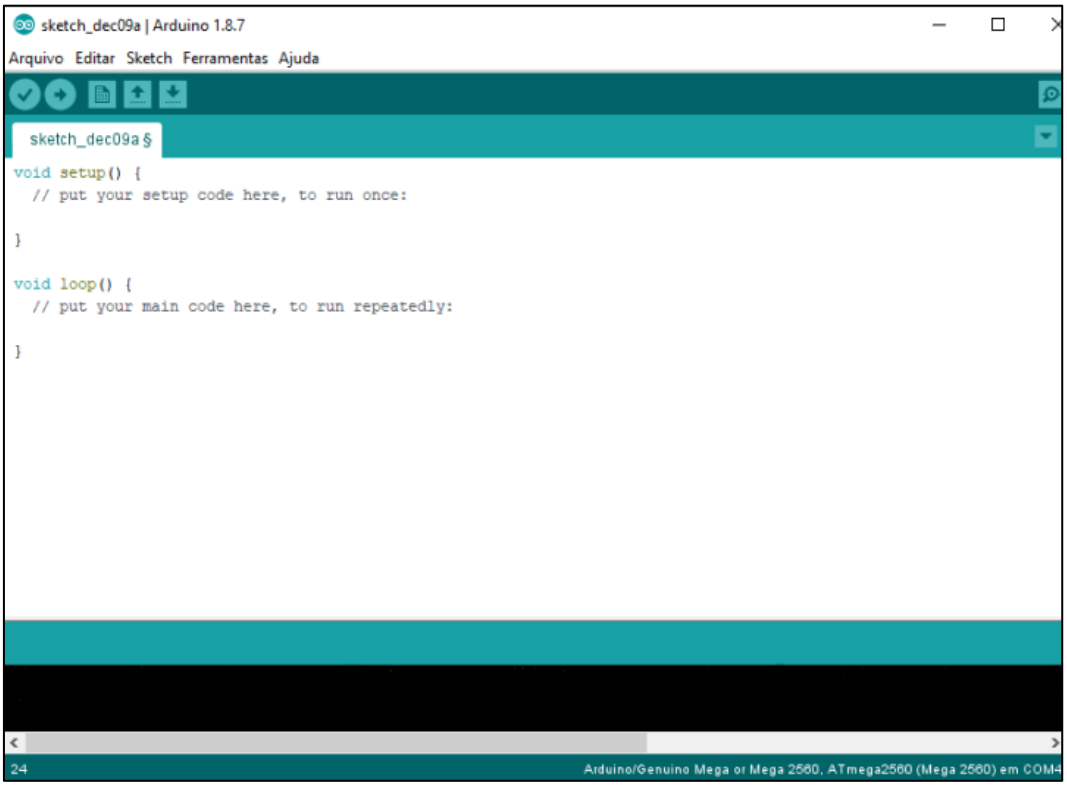


Fonte: Datasheet ATmega2560 (2014).

2.2 IDE do Arduino

A placa Arduino pode ser programada utilizando o IDE (*Integrated Development Environment*), conforme mostrado na Figura 3. Esse *software* permite escrever, salvar e compilar o código desenvolvido (BANZI, 2015), convertendo assim uma linguagem de alto nível em uma linguagem de máquina. Através do IDE do Arduino também é possível fazer o *upload* do código para o microcontrolador. Este *software* está disponível gratuitamente no site do Arduino (www.arduino.cc). Nesse ambiente é possível também fazer a inclusão de bibliotecas a fim de simplificar a implementação de aplicações, já que instruções específicas já foram desenvolvidas nesse espaço para um determinado dispositivo, como por exemplo, um sensor, um módulo WiFi, entre outros. O IDE do Arduino já vem com várias bibliotecas padrão, sendo que outras bibliotecas podem ser instaladas pelo próprio usuário nesse ambiente. O Arduino pode ser programado com a linguagem de programação Arduino que é baseada na linguagem *Wiring* (ARDUINO, 2018).

Figura 3: IDE do Arduino



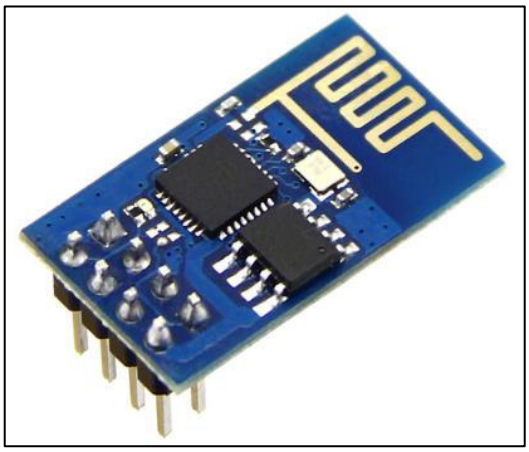
Fonte: Autor

2.3 ESP8266

O ESP8266 é um microcontrolador projetado pela Espressif Systems. Esse chip possui WiFi embutido e tamanho reduzido, características que fazem desse módulo um componente que pode ser incluso em aplicações de IoT (SCHWARTZ, 2016).

Atualmente existem vários modelos disponíveis no mercado (KOLBAN, 2016). Conforme mostrado na Figura 4, nesse trabalho foi utilizado o ESP-01, pois uma das diferenças entre o mesmo e os outros modelos é o número de pinos de entrada e saída disponíveis, como suas funções nesse projeto são se conectar a uma rede WiFi e enviar os valores dos sensores coletados pelo Arduino para uma plataforma de armazenamento de dados na nuvem e para o banco de dados MySQL, o número de GPIOs do módulo não foi considerado como condição para a implementação desse trabalho.

Figura 4: ESP8266 modelo ESP-01

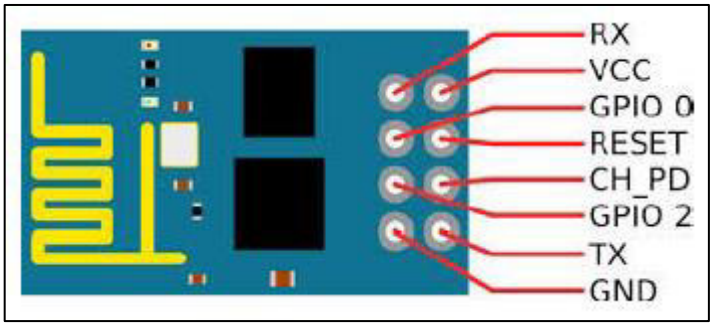


Fonte: http://www.filipeflop.com

Esse chip permite a conexão do Arduino com as redes WiFi baseadas no padrão IEEE 802.11 b/g/n. A tensão de operação é de 3,3 V, a arquitetura é RISC de 32 bits (ESPRESSIF, 2013). Possui 32 KBytes de RAM para instruções, 96 Kbytes de RAM para dados, uma memória *flash* de 512 KBytes (CURVELLO, 2015).

É possível verificar qual a versão do *firmware* que está gravado nessa placa e realizar o upgrade para uma nova versão (KOLBAN, 2016). A placa ESP-01 é composta por 8 pinos. Na Figura 5, pode-se observar a disposição de cada pino desse modelo do ESP8266.

Figura 5: Pinagem do ESP8266 modelo ESP01



Fonte: KOLBAN (2016, p. 36).

A descrição de cada um dos pinos desse chip pode ser visualizada na Tabela 1.

Tabela 1 - Descrição dos pinos da placa ESP8266 modelo ESP-01

Pino	Descrição
TX	Transmissão Serial
RX	Recepção Serial
CH_PD	*Enable*
RST	*Reset*
GPIO 0	Porta de entrada e saída de dados 0
GPIO 1	Porta de entrada e saída de dados 1
VCC	3,3 V
GND	Terra

Fonte: Adaptado de KOLBAN (2016).

O ESP8266 permite ser configurado como: cliente, ponto de acesso ou ambos (ESPRESSIF, 2018). Através do modo cliente o chip pode estabelecer uma conexão com a rede WiFi criada por um ponto de acesso e no modo ponto de acesso o mesmo pode criar sua própria rede.

2.4 Sensor de Temperatura ambiente e Umidade relativa do ar

A temperatura pode ser definida como o grau de agitação das moléculas, de tal forma que quanto mais agitadas as moléculas estiverem mais elevada será a temperatura. No caso da temperatura do ar, a mesma varia de um lugar para o outro e com o decorrer do tempo (AYOADE, 1996).

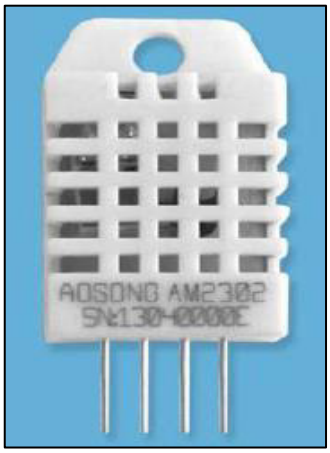
Alterações no aquecimento da superfície proporcionam mudanças no comportamento médio da atmosfera, afetando dessa forma a variável climática temperatura (VAREJÃO, 2005). Outros fatores também influenciam na distribuição

de temperatura em uma determinada localidade, tais como: a natureza dos ventos predominantes, as correntes oceânicas e o relevo (AYOADE, 1996).

A umidade é definida como a quantidade de água na forma de vapor contida na atmosfera. Dois dos índices de umidade utilizados são: umidade absoluta e umidade relativa. A umidade absoluta é expressa como a massa total de água num dado volume de ar, já a umidade relativa refere-se a razão entre o conteúdo real de umidade de uma amostra de ar e a quantidade máxima que o mesmo volume de ar pode conservar na mesma temperatura e pressão quando saturado (AYOADE, 1996).

Para realizar a medição de temperatura atmosférica e da umidade relativa do ar foi utilizado o sensor digital DHT22, conforme mostrado na Figura 6. Esse componente é composto por um sensor capacitivo de umidade e um termistor, sendo que os mesmos estão conectados a um microcontrolador de 8 bits (AOSONG, 2018).

Figura 6: Sensor DHT22



Fonte: AM2302 Product Manual (2018).

O sensor DHT22 permite realizar leituras de temperaturas na faixa de -40ºC a 80ºC com precisão de ± 0,5ºC, sua faixa de medição de umidade é de 0 a 100% com precisão de ± 2-5% (AOSONG, 2018). As principais características relacionadas a performance desse sensor constam na Tabela 2.

Tabela 2 - Características do sensor DHT22

Parâmetro	DHT22
Máxima corrente de operação	1,5 mA
Range de umidade	0 a 100% / ± 2-5%
Range de Temperatura	-40 a 80ºC / ± 0,5ºC
Taxa de Amostragem	0,5 Hz
Resolução	0,1

Fonte: Adaptado de AM2302 Datasheet (2018).

Para entender o funcionamento de um sensor de umidade capacitivo é necessário revisar o conceito de capacitor. Segundo Du (2015, p. 100), o capacitor é um componente eletrônico que pode armazenar energia na forma de um campo elétrico.

Existem diferentes tipos de configuração para esses componentes, no caso do capacitor formado por placas paralelas condutoras separadas por um dielétrico, a variável capacitância pode ser calculada através da Equação 1 (DU, 2015):

$$C = \frac{\varepsilon 0 \varepsilon r A}{d} \quad (1)$$

Onde,

C = Capacitância

$\varepsilon 0$ = Permissividade dielétrica do vácuo e vale 8,85. 10^{-12} C²/N.m²

εr = Permissividade relativa ou constante dielétrica do meio

A = Área das placas condutoras

d = Distância entre as placas condutoras

Para compreender o conceito de permissividade relativa é necessário entender a relação entre o dielétrico e as placas condutoras de um capacitor. Quando precisa-se obter capacitores com grande capacitância de acordo com a Equação 1, uma das alternativas é diminuir a distância (d) entre as placas paralelas que constituem esse componente e garantir que as mesmas não encostem uma na outra na presença de alguma vibração, por exemplo. Logo, a solução é colocar um material isolante entre as placas de tal forma que a inserção do mesmo irá alterar o valor da capacitância. Então, cada dielétrico está associado a uma determinada

propriedade, sendo a mesma definida como constante dielétrica ou permissividade relativa (ε_r). Essa variável pode ser calculada através da Equação 2 (DU, 2015):

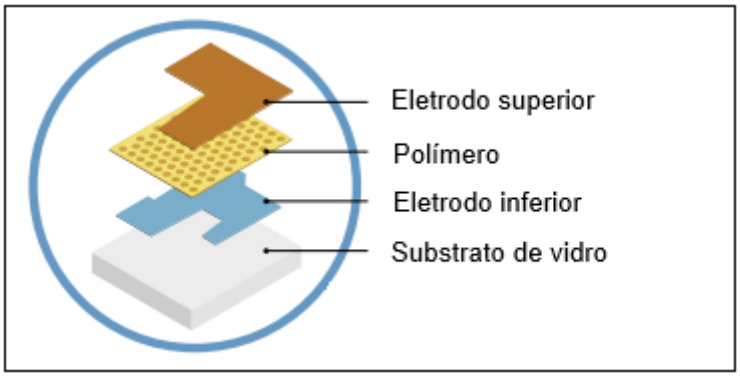
$$\varepsilon_r = \frac{\varepsilon_p}{\varepsilon_0} \quad (2)$$

Onde,

ε_p = Permissividade da substância dielétrica

O elemento sensor de umidade do DHT22 é do tipo capacitivo formado por duas placas condutoras paralelas separadas por um substrato sobre o qual é depositado uma película de polímero (ADAFRUIT, 2018). Nesses tipos de sensores, a película de polímero absorve ou libera vapor de água, quando a umidade relativa do ar no ambiente diminui ou aumenta, visto que as propriedades dielétricas da película mudam, como consequência da alteração da permissividade relativa equivalente (CASTRO, 2011). Logo, de acordo com a Equação 1, a capacitância também irá alterar. Segundo Du (2015, p. 101) sensores capacitivos são sensores variáveis, isto é, suas capacitâncias mudam durante o processo de detecção. Na Figura 7 pode ser visualizada a estrutura interna de um sensor de umidade capacitivo com película fina de polímero.

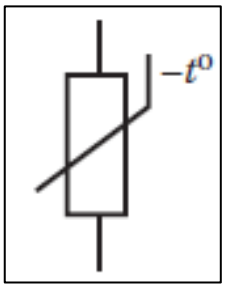
Figura 7: Estrutura interna de um sensor de umidade capacitivo com película fina de polímero



Fonte: Adaptado de HUMICAP Sensors Datasheet (2018).

O elemento sensor de temperatura do DHT22 é um termistor NTC (*Negative Temperature Coefficient*), esse dispositivo eletrônico diminui a sua resistência elétrica com o aumento da temperatura (DU, 2015). Na Figura 8 pode ser visualizado o símbolo do circuito para um termistor do tipo NTC.

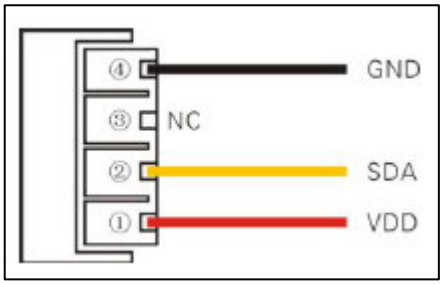
Figura 8: Símbolo do circuito de um termistor NTC (com sinal negativo)



Fonte: Du (2015, p. 48)

O sensor DHT22 é composto por 4 pinos. O pino 1 (VDD) refere-se à alimentação do sensor que pode variar de 3,3 a 5,5 V, sendo o recomendado alimentá-lo com uma tensão de 5 V. O pino 2 (SDA) é utilizado para estabelecer a comunicação entre o sensor e o microcontrolador, o pino 3 (NC) não deve ser conectado e o pino 4 deve ser conectado ao *ground* do microcontrolador (AOSONG, 2018). Na Figura 9, pode-se visualizar a disposição desses pinos.

Figura 9: Pinagem do sensor DHT22

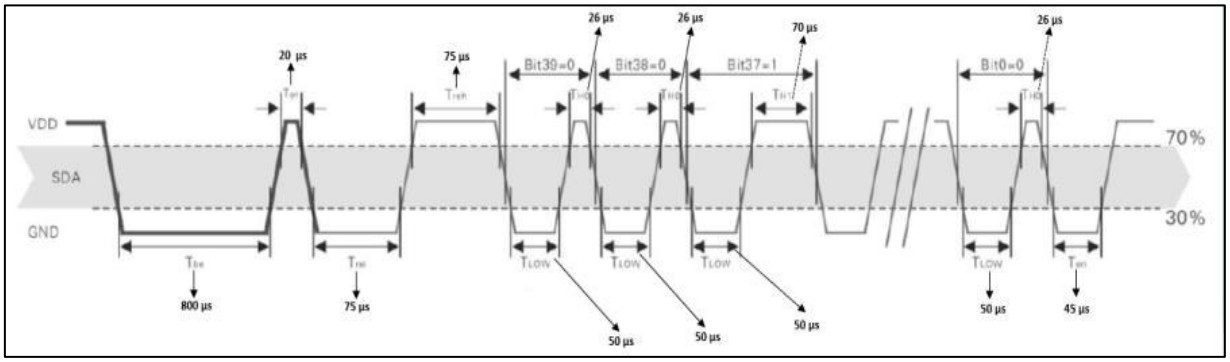


Fonte: AM2302 Product Manual (2018).

O DHT22 utiliza um protocolo próprio para comunicação, semelhante ao desenvolvido pela *Dallas Semiconductor Corp.* que atualmente é a *Maxim Integrated Products*. A comunicação é do tipo *one wire*, ou seja, os dados medidos pelo sensor são enviados digitalmente através de um único fio para o microcontrolador. Esse protocolo é baseado na estrutura mestre-escravo, de tal forma que o sensor responderá somente quando o microcontrolador realizar uma requisição (AOSONG, 2018).

A comunicação inicia com o dispositivo mestre (microcontrolador) enviando um sinal em nível 0 para o barramento de dados por no mínimo 800 µs, após esse tempo, o dispositivo mestre envia um sinal em nível 1 para o barramento por pelo menos 20 µs. Esta sequência funciona como um sinal de partida para o sensor. Após receber esse sinal, o dispositivo *slave* (nesse caso será o sensor de temperatura e umidade DHT22) responde enviando 0 para o barramento por no mínimo 75 µs, em seguida envia 1 para o barramento por pelo menos 75 µs, nesse período o sensor testa novamente a temperatura, a umidade relativa do ambiente e registra os dados. Após a finalização do sinal de resposta, antes de iniciar o envio de cada bit do valor digital, o sensor envia um sinal em nível 0 para o barramento por cerca de 50 µs. Em seguida o sensor inicia o envio dos 40 bits de tal forma que para esse tipo de comunicação o tempo de duração do pulso em nível lógico ALTO (1) determina o estado do bit enviado, ou seja, 0 ou 1. Se o pulso em nível 1 permanecer nesse estado por cerca de 26 a 28 µs, indica que o bit enviado possui valor 0, entretanto, um pulso em nível 1 com duração de 70 µs indica que o bit enviado possui valor 1. Após ter concluído a transmissão de todos os bits o sensor envia um sinal em nível 0 para o barramento por pelo menos 45 µs e entra no modo de *sleep* automaticamente (AOSONG, 2018). Na Figura 10, pode-se observar o processo de comunicação entre o disposto mestre e o sensor DHT22, assim como a duração de cada etapa associada ao mesmo.

Figura 10: Etapas e duração da comunicação entre dispositivo mestre e escravo



Fonte: Adaptado de AM2302 Product Manual (2018).

O sensor DHT22 envia a informação do valor referente a umidade relativa e temperatura ambiente em um conjunto formado por 40 bits, de tal forma que os primeiros 16 bits correspondem ao valor da umidade relativa, os próximos 16 bits são referentes ao valor da temperatura ambiente e os últimos 8 bits representam a

paridade, esse byte é necessário para verificar se não houve nenhum erro na transmissão dos bytes de umidade e temperatura para o microcontrolador. É necessário ressaltar que o valor em decimal correspondente aos 16 bits enviados tanto para representar o valor da umidade quanto da temperatura correspondem a 10 vezes os valores reais dessas variáveis, para exemplificar este ponto, pode-se considerar que o microcontrolador recebeu os 40 bits mostrados abaixo (AOSONG, 2018):

0000 0010 1001 0010 0000 0001 0000 1101 1010 0010

Sabe-se que os primeiros 16 bits correspondem ao valor de umidade relativa, portanto, transformando de binário para decimal o conjunto de 2 bytes, 0000 0010 1001 0010, encontra-se o valor 658 em decimal. Os 16 bits seguintes correspondem ao valor de temperatura ambiente, logo, transformando de binário para decimal o conjunto de 2 bytes, 0000 0001 0000 1101, encontra-se o valor 269 em decimal, sendo que esses valores correspondem a 10 vezes o valor real dessas variáveis, de tal forma que dividindo ambos os valores por 10, encontra-se o resultado final para essas medições que corresponde respectivamente a 65,8% e 26,9ºC .

2.5 Sensor de Pressão Atmosférica

Assim como todos os corpos o ar tem peso. A pressão atmosférica corresponde ao peso da coluna de ar sobre a área correspondente (TORRES; MACHADO, 2008). Tanto a temperatura quanto a latitude e a longitude provocam variação na pressão atmosférica (AYOADE, 1996).

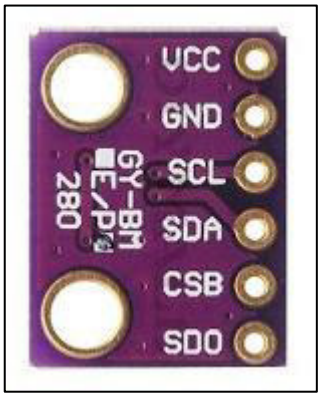
> A temperatura faz variar a pressão atmosférica por que o calor “dilata” o ar, tornando-o mais leve e determinando, por consequência, uma menor pressão do ar sobre a superfície (baixa pressão), ou seja, para uma mesma condição de altitude entre dois pontos quaisquer, a pressão sofrerá variação desde que a temperatura entre esses dois pontos seja diferente. (TORRES; MACHADO, 2008, p.38).

Para realizar a medição de pressão atmosférica foi utilizado o sensor digital BMP280 desenvolvido pela *Bosh Sensortec*. Esse sensor mede pressão absoluta em pascal, sendo que pode efetuar leituras na faixa entre 300 hPa a 1100 hPa, essa faixa corresponde a uma variação de altitude de + 9000 a – 500 m. A precisão das medições é de ± 0.12 hPa. A tensão de operação é de 3,3 V e o consumo de

corrente é de aproximadamente 2,7 µA. Esse sensor funciona com interfaces SPI e I2C (SENSORTEC, 2015).

O sensor BMP280 é composto por 6 pinos, o pino de alimentação (Vcc), o *ground* (GND) e os pinos de comunicação entre dispositivos SPI que são: SCL (Serial Clock), SDA (Serial Data), CSB (Chip Select Bar) e SDO (Serial Data Output). Os nomes padrões para os pinos de interface SPI são respectivamente: MOSI (Master Out- Slave In), MISO (Master In-Slave Out), SCLK (Clock Signal), SS (Select Signal) (LEENS, 2009). Na Figura 11, mostra-se o sensor BMP280 e seus respectivos pinos.

Figura 11: Sensor BMP280 e pinagem



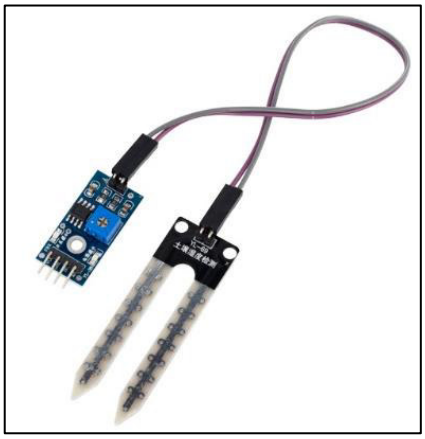
Fonte: www.arduinoecia.com.br

2.6 Sensor de Umidade do Solo

Monitorar a umidade do solo significa não somente inibir gastos com água ou energia elétrica, mas como também evitar a ocorrência de doenças em uma determinada plantação. Com os dados de umidade do solo é possível implementar um sistema para controlar o processo de irrigação (BANDERALI, 2010).

O sensor de umidade do solo utilizado nesse projeto possui o componente LM393 como comparador interno. É um sensor do tipo resistivo. Pode-se ajustar a sensibilidade do mesmo através do potenciômetro presente em sua estrutura. É composto por uma saída digital e outra analógica. A partir da saída analógica é possível controlar diversos tipos de umidade do solo. A tensão de alimentação é de 3,3 a 5 V (THOMSEN, 2016). Na Figura 12, mostra-se o sensor de umidade do solo utilizado nesse trabalho.

Figura 12: Sensor de umidade do solo



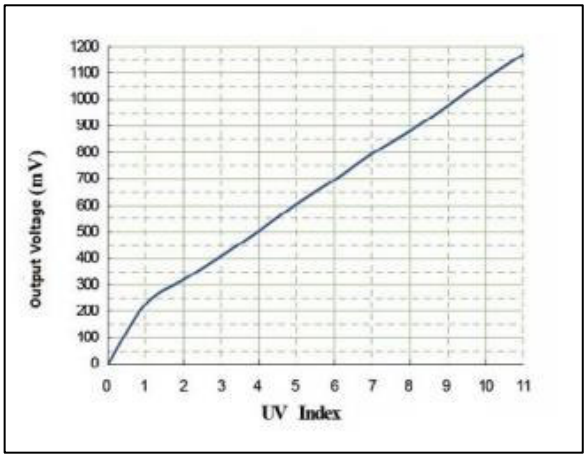
Fonte: www.filipeflop.com

2.7 Sensor de Índice UV

A radiação ultravioleta é uma das emitidas pelo sol, corresponde a radiação eletromagnética com um comprimento de onda menor do que o da luz visível. Em relação a região do espectro em que se encontra, a mesma está na faixa de comprimento de onda de 100 a 400 nm (SANTOS, 2010). Dependendo do tipo de raio UV, alguns dos danos que os mesmos podem causar aos seres humanos são: envelhecimento precoce, enfraquecimento do sistema imunológico do corpo, cegueira e até mesmo câncer de pele (EPA, 2017).

O sensor de índice UV utilizado nesse trabalho utiliza o chip UVM-30A, possui três pinos, o pino de alimentação VCC, o qual deve ser alimentado com 5 V, o pino GND e um pino de saída analógica. Esse sensor tem uma precisão de ± 1 UV. Conforme o índice UV, o mesmo gera uma tensão em uma faixa específica. Pode detectar raios UV com comprimento de onda entre 200~370 nm. O consumo de corrente padrão é de aproximadamente 0,06 mA e o máximo consumo pode ser de 0,1 mA (WILTRONICS, 2018). A Figura 13 mostra as faixas de tensões possíveis geradas por esse sensor e o valor de índice UV correspondente as mesmas.

Figura 13: Faixas de tensões possíveis e valor de índice UV do Sensor UVM-30A



Fonte: UVM-30A Datasheet (2018).

2.8 Pluviômetro Arduino

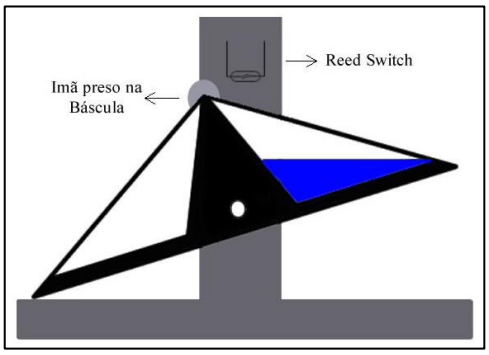
Precipitação segundo Torres e Machado (2008, p. 53) é o processo pelo qual a água condensada na atmosfera atinge a superfície terrestre, sob a forma líquida (chuva ou chuvisco) ou sólida (granizo, saraiva ou neve).

> No sistema internacional de medidas, a unidade de medida representativa da pluviosidade é o milímetro (mm), desta maneira, uma pluviosidade de 1 milímetro capturado pelo Pluviômetro Arduino equivale a um volume de 1 litro de água da chuva por metro quadrado (STRAUB, 2018, p.2).

Em sua estrutura interna, o pluviômetro utilizado nesse projeto possui uma báscula que se movimenta sempre que 0,25 mm de água é identificado pela mesma. Há também um *reed switch*, a função do mesmo no pluviômetro é detectar se houve movimento da báscula (STRAUB, 2018).

O *reed switch* é um interruptor formado por duas lâminas, sendo que sem a presença de um campo magnético essas lâminas encontram-se separadas nesse dispositivo, entretanto, sempre que há um campo magnético próximo ao mesmo, as lâminas se atraem, unem-se e possibilitam a passagem de corrente elétrica (HSI, 2013). Na Figura 14, pode-se visualizar a estrutura interna de um pluviômetro de báscula automático, através da mesma é possível observar que junto a báscula é conectado um imã, logo, sempre que o mesmo passa pelo interruptor um pulso é enviado para o pino digital do Arduino (STRAUB, 2018).

Figura 14: Estrutura interna do pluviômetro automático de báscula



Fonte: usinainfo.com.br

2.9 ThingSpeak

De acordo com seus desenvolvedores o ThingSpeak é considerado um *web service* gratuito que permite armazenar, visualizar e analisar dados em tempo real. É considerado uma plataforma de armazenamento de dados na nuvem e que está inserido no contexto de IoT. Os valores podem ser visualizados instantaneamente, ou seja, a visualização dos dados não se limita somente a médias, por exemplo.

2.10 Banco de Dados

Um banco de dados é composto por quatro partes, a base de dados, um sistema gerenciador de banco de dados (SGBD), uma linguagem de exploração (linguagem de acesso aos dados) e programas adicionais, como por exemplo, otimizadores de dados (ELMASRI; NAVATHE, 2011). Em relação aos bancos de dados gratuitos, pode-se considerar o MySQL como uma das soluções mais populares, o mesmo pertence a *Oracle*.

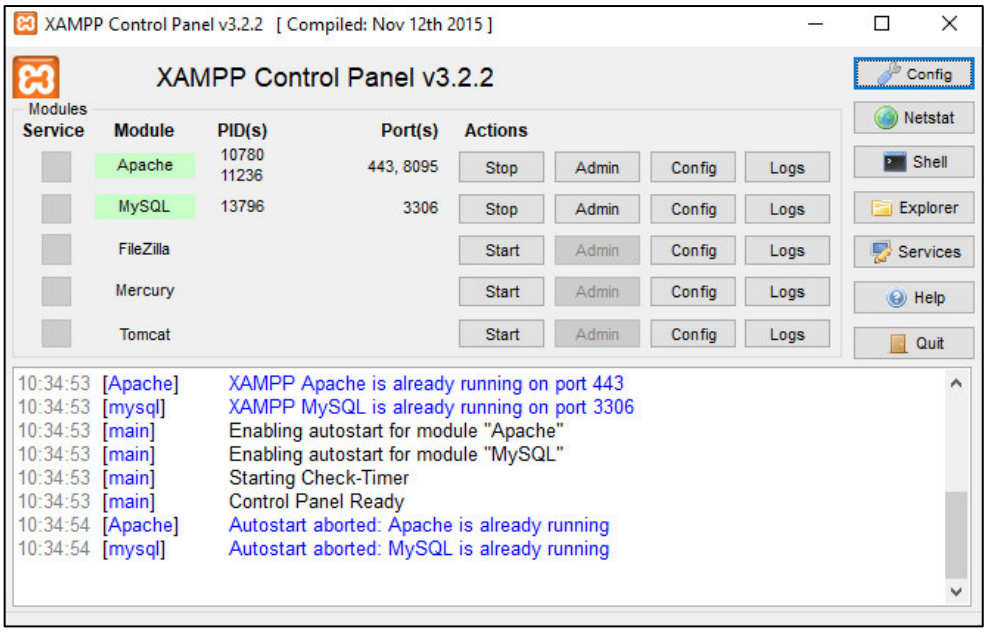
2.11 PHP

O PHP (*Hypertext Preprocessor*) é considerado uma linguagem de *script* de uso geral e por sua vez é *open source*. Bases de dados armazenadas em diversos tipos de bancos de dados relacionais podem ser acessados através de PHP, tal como, o MySQL, já que os mesmos podem ser acessados com linguagem de *scripting*, logo, pode-se por exemplo, estabelecer a conexão de uma determinada aplicação com um banco de dados utilizando PHP. Os interpretadores para PHP são fornecidos gratuitamente e escritos na linguagem C, de modo que estão disponíveis na maioria das plataformas de computador (ELMASRI; NAVATHE, 2011, p.325).

2.12 XAMPP

Para manipulação e acesso de um banco de dados com uma linguagem de desenvolvimento *web*, como o PHP, é necessário a execução da mesma em um servidor *web*, como por exemplo, o Apache. Há disponível para download diretamente do site oficial (https://www.apachefriends.org/pt_br/download.html), uma plataforma que facilita a configuração do ambiente necessário para enviar informações para um banco de dados e manipulá-las, caso seja necessário. Essa ferramenta é o XAMPP, o qual refere-se a um pacote de instalação com uma série de programas inclusos, tais como: o Apache (servidor HTTP), MySQL (servidor de banco de dados) e o PHP (MADUREIRA JUNIOR; SACILOTTI, A. C.; SACILOTTI, R., 2017). Na Figura 15, pode-se visualizar o painel de controle do XAMPP.

Figura 15: Painel de Controle do XAMPP



Fonte: Autor

Nesse trabalho, foi instalado o XAMPP com a inicialização do programa Apache a fim de transformar o computador local em um servidor *web*. Foi criado um banco de dados MySQL e uma tabela em uma ferramenta *open source* utilizada para administração de banco de dados, o phpMyAdmin, a base de dados foi nomeada de estacao_meteorologica e a tabela de tabelaestacaometeorologica.

3 IMPLEMENTAÇÃO DA ESTAÇÃO METEOROLÓGICA

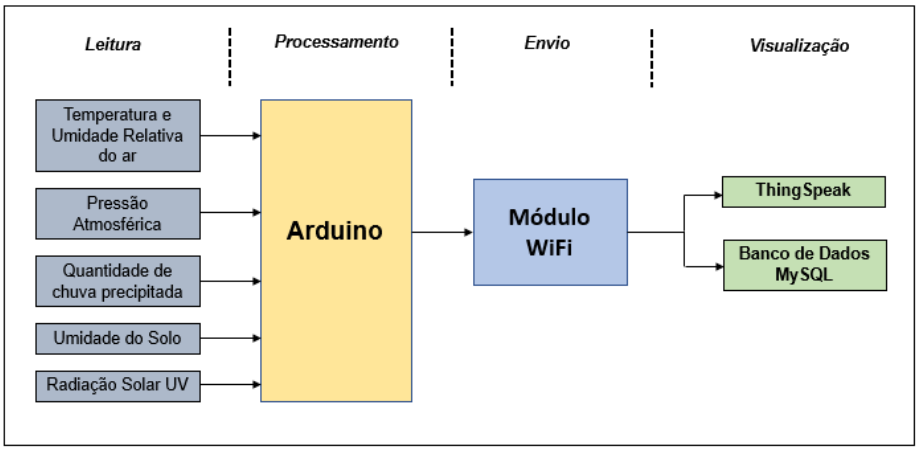
Este projeto propõe o desenvolvimento de um sistema de *hardware*/*software* de uma estação meteorológica. A implementação do mesmo foi baseada nas etapas abaixo:

- Definição do modelo do protótipo da estação meteorológica;
- Montagem do *hardware* da estação meteorológica;
- Criação de um banco de dados e de uma tabela utilizando o phpMyAdmin como interface;
- Implementação do *software* no IDE do Arduino;
- Desenvolvimento do *script* em PHP;

3.1 Modelo Proposto

O protótipo proposto e implementado para o sistema de *hardware/software* da estação meteorológica pode ser visualizado na Figura 16. Após a inserção dos sensores para captura dos dados meteorológicos, o Arduino irá processar esses dados e a partir do módulo WiFi os mesmos serão enviados para uma plataforma de armazenamento de dados na nuvem, o ThingSpeak e para um *web service* que os inserirá em um banco de dados MySQL.

Figura 16: Modelo proposto



Fonte: Autor

3.2 Montagem do *Hardware*

A implementação do circuito de *hardware* da estação meteorológica foi baseada na ligação de cada componente selecionado com o Arduino e foi dividida nas etapas abaixo:

Etapa 1: Instalação do sensor de temperatura e umidade relativa do ar DHT22;

Etapa 2: Instalação do sensor de pressão atmosférica BMP280;

Etapa 3: Instalação do sensor de umidade do solo;

Etapa 4: Instalação do sensor de índice UV;

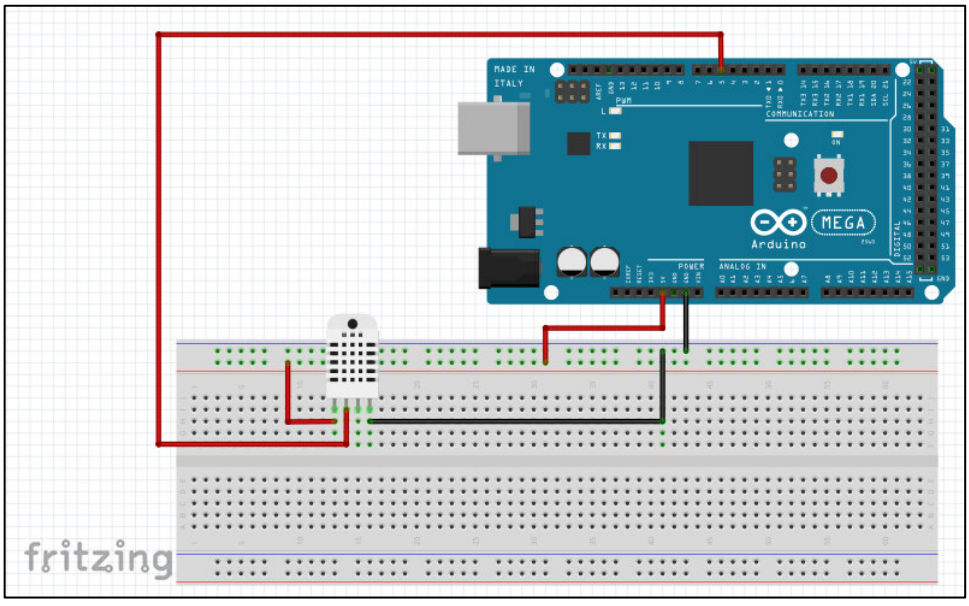
Etapa 5: Instalação do pluviômetro;

Etapa 6: Instalação do SoC com módulo WiFi ESP8266 ESP-01;

3.2.1 Instalação do sensor de temperatura e umidade relativa do ar DHT22

Nesta etapa foram realizadas as conexões do sensor DHT22 com o Arduino. Como dito no Capítulo 2, o sensor DHT22 é formado por 4 pinos, sendo que nesse projeto, o pino VCC foi conectado a alimentação de 5 V do Arduino, o pino de dados foi conectado ao pino digital 5 do Arduino e o pino de terra foi conectado ao GND do Arduino. Na Figura17, pode-se visualizar essas ligações.

Figura 17: Esquema de ligação do sensor DHT22

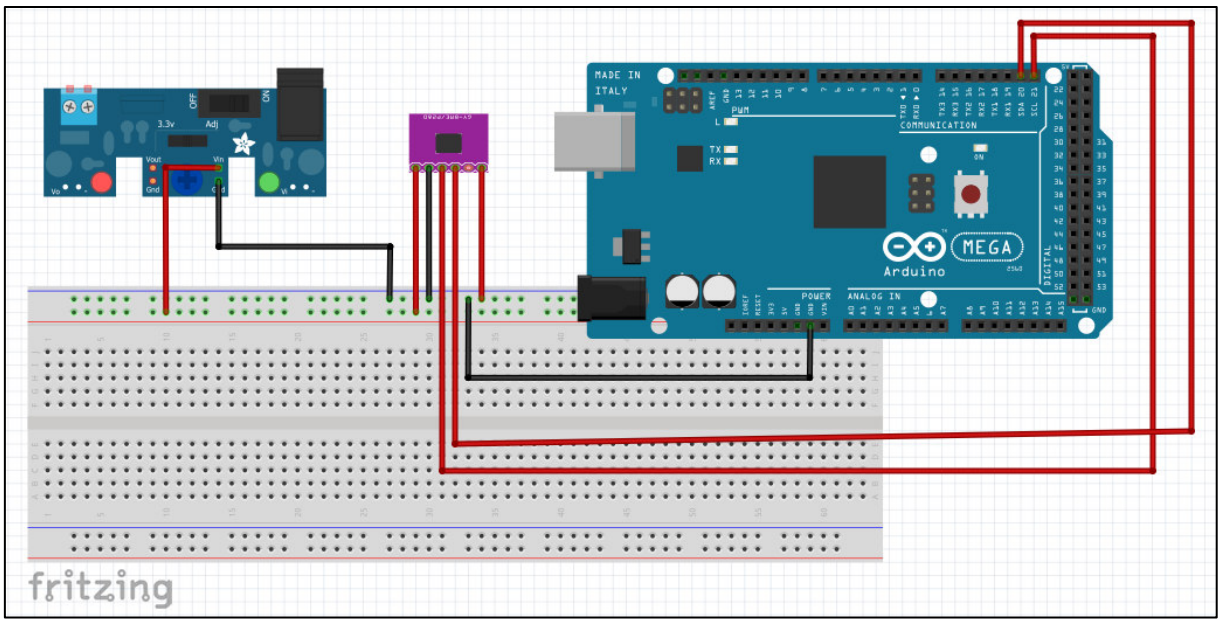


Fonte: Autor

3.2.2 Instalação do sensor de pressão atmosférica BMP280

Nesta etapa foram realizadas as conexões do sensor de pressão atmosférica BMP280 com o Arduino. Conforme mencionado no Capítulo 2, o sensor BMP280 é formado por 6 pinos, sendo que de acordo com os pinos utilizados pode-se conectar o módulo ao Arduino usando uma interface I2C ou SPI. Nesse projeto foi escolhido o protocolo I2C, portanto, os pinos utilizados foram SCL, SDA, VCC e o pino SDO, cuja finalidade é selecionar o endereço I2C. Ao configurar o pino SDO em nível baixo, o endereço I2C corresponde a 0x76, caso seja configurado em nível alto, o endereço I2C corresponde a 0x77. As ligações desse dispositivo com o Arduino foram as seguintes: o pino VCC foi conectado a uma fonte de alimentação externa chaveada de 3,3 V, o pino GND foi ligado ao GND do Arduino, o pino SDA do módulo foi conectado ao SDA do Arduino, que corresponde ao pino D20 e o pino SCL do módulo foi ligado ao SCL do Arduino, que corresponde ao pino D21. O pino SDO foi configurado em nível lógico alto, logo, foi alimentado com 3,3 V. Na Figura 18, pode-se visualizar essas ligações.

Figura 18: Esquema de ligação do sensor BMP280

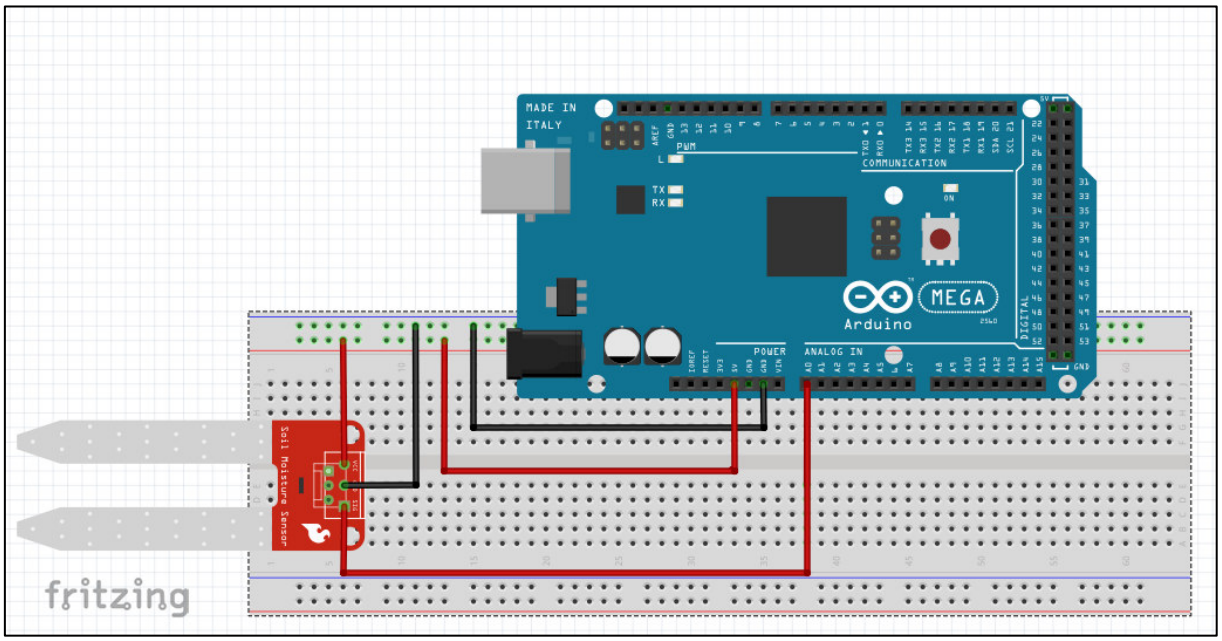


Fonte: Autor

3.2.3 Instalação do sensor de umidade do solo

Para medir a umidade do solo foi conectado ao sistema de *hardware* o sensor mencionado no Capítulo 2 desse trabalho, conforme dito anteriormente, esse sensor é formado por 6 pinos, sendo que dois desses pinos são utilizados para conectar a haste de metal (não há polaridade para essa ligação). Para determinação da umidade do solo foi escolhido o sinal analógico. As ligações foram as seguintes: o pino A0 do módulo no A0 do Arduino, a alimentação VCC do módulo no pino de 5 V do Arduino e o GND do dispositivo no GND do Arduino. Na Figura19, pode-se visualizar essas ligações.

Figura 19: Esquema de ligação do sensor de umidade do solo

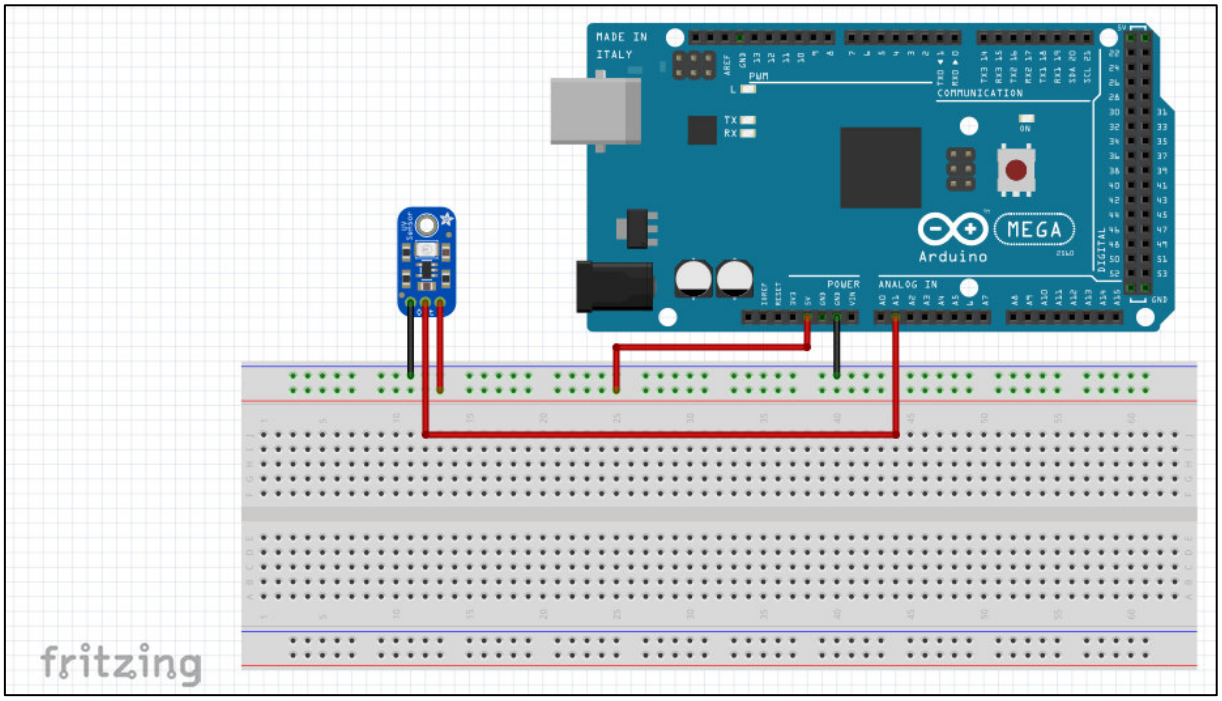


Fonte: Autor

3.2.4 Instalação do sensor de índice UV

Com o objetivo de obter o nível de radiação solar ultravioleta foi instalado o sensor UVM-30A ao *hardware*. Conforme dito no Capítulo 2, esse sensor é formado por 3 pinos. As conexões do mesmo com o Arduino foram as seguintes: VCC do módulo no pino de alimentação de 5 V do Arduino, a saída analógica do sensor ao A1 do Arduino e o GND ao GND do Arduino. Na Figura 20, pode-se visualizar essas ligações.

Figura 20: Esquema de ligação do sensor de índice UV

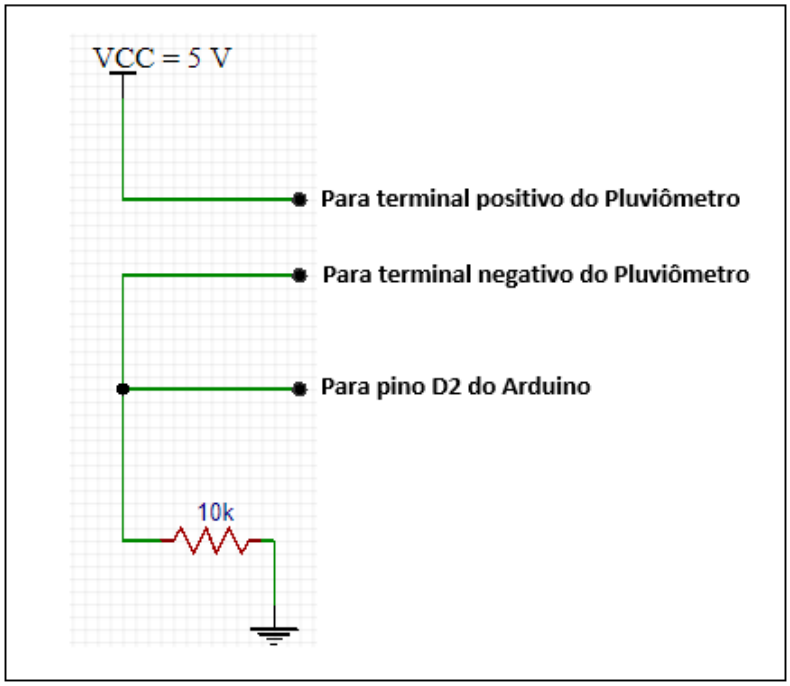


Fonte: Autor

3.2.5 Instalação do pluviômetro

Nesta etapa foi adicionado o pluviômetro de báscula digital, a fim de medir a quantidade de chuva precipitada. Dos pinos digitais do Arduino MEGA 2560 que estão habilitados para interrupções, o selecionado para gerar a rotina de interrupção foi o digital 2. Conforme mostrado na Figura 21, um resistor de *pull down* de 10 kΩ é conectado com o digital 2 do Arduino, o terminal positivo do pluviômetro é conectado a alimentação de 5 V do Arduino e o terminal negativo do pluviômetro ao digital 2 do Arduino.

Figura 21: Esquema de ligação do pluviômetro de báscula

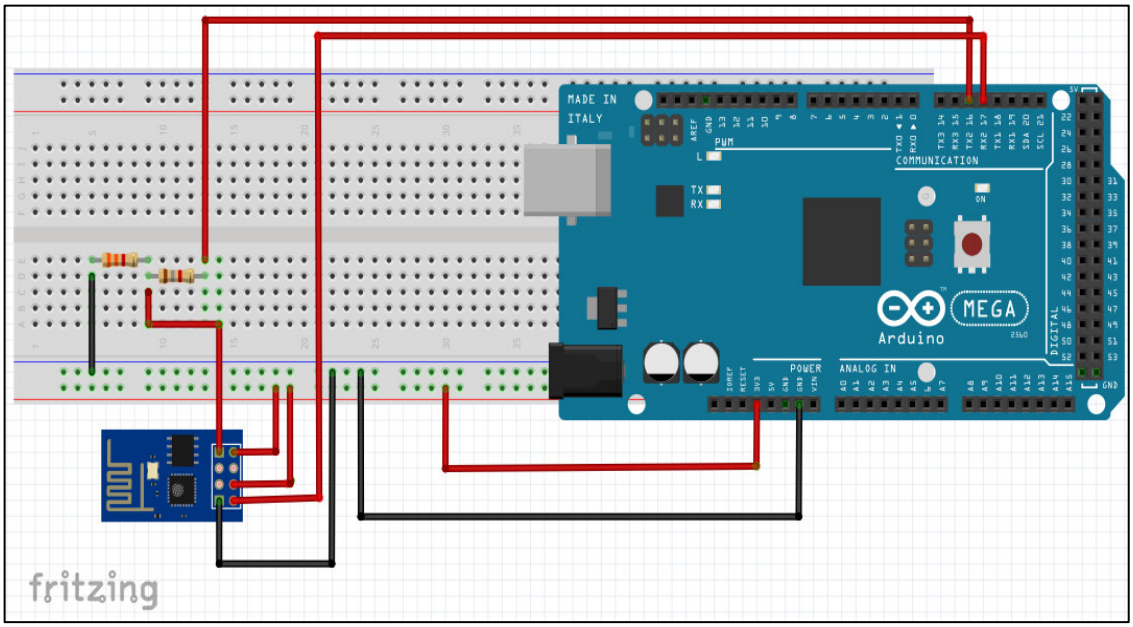


Fonte: Autor

3.2.6 Instalação do microcontrolador com módulo WiFi ESP8266

Nesta etapa foi adicionado ao protótipo o módulo WiFi ESP8266 modelo ESP-01. As ligações do ESP8266 no Arduino foram as seguintes: o pino VCC do módulo na alimentação de 3,3 V do Arduino, o GND do módulo conectado ao GND do Arduino, o pino CH_PD ligado a alimentação de 3,3 V do Arduino. Para estabelecer a comunicação de dados entre o Arduino e o ESP8266, o pino RX do módulo deve ser conectado ao TX do Arduino e o TX do módulo deve ser ligado ao RX do microcontrolador, entretanto, deve-se atentar que todas as conexões desse módulo WiFi precisam considerar a tensão do mesmo que é de 3,3 V. Logo, na conexão entre o pino RX do ESP8266 e o TX do Arduino foi adicionado um divisor de tensão formado por dois resistores (3,3 kΩ e 1,8 kΩ) de tal forma a garantir uma tensão de 3,3 V no pino RX do ESP8266. Para esse projeto a porta serial do Arduino MEGA 2560 escolhida foi a 2, representada pelos pinos 17 (RX) e 16 (TX). Na Figura 22, pode-se visualizar as ligações mencionadas.

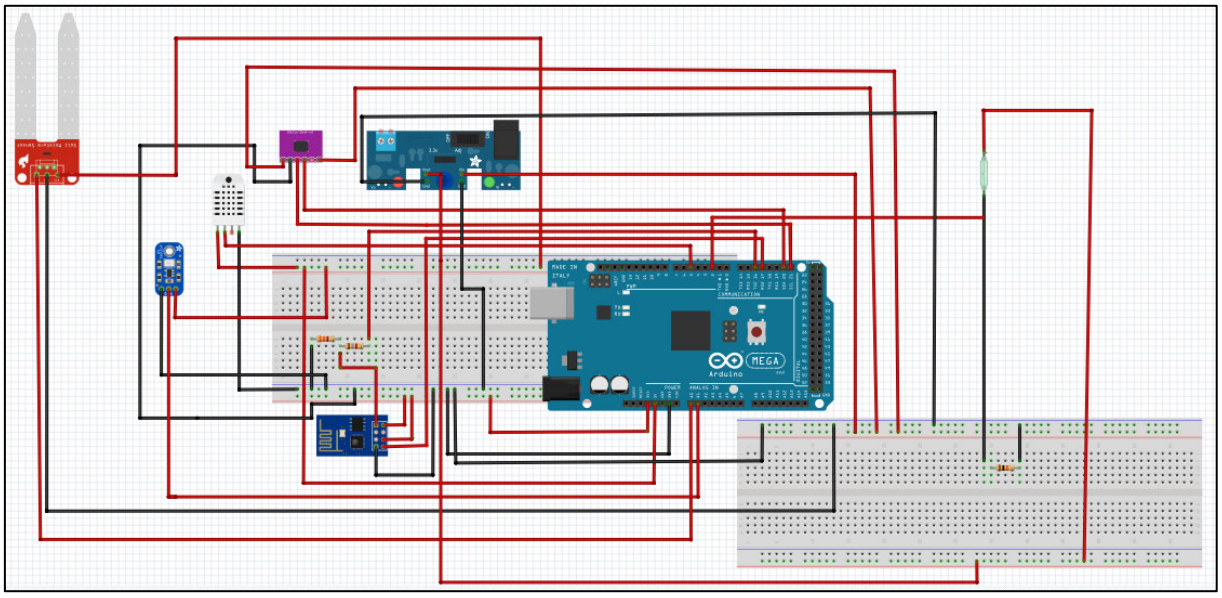
Figura 22: Esquema de ligação do módulo WiFi ESP8266 ESP-01



Fonte: Autor

O circuito final do sistema com o microcontrolador, o módulo WiFi e os sensores pode ser visualizado na Figura 23.

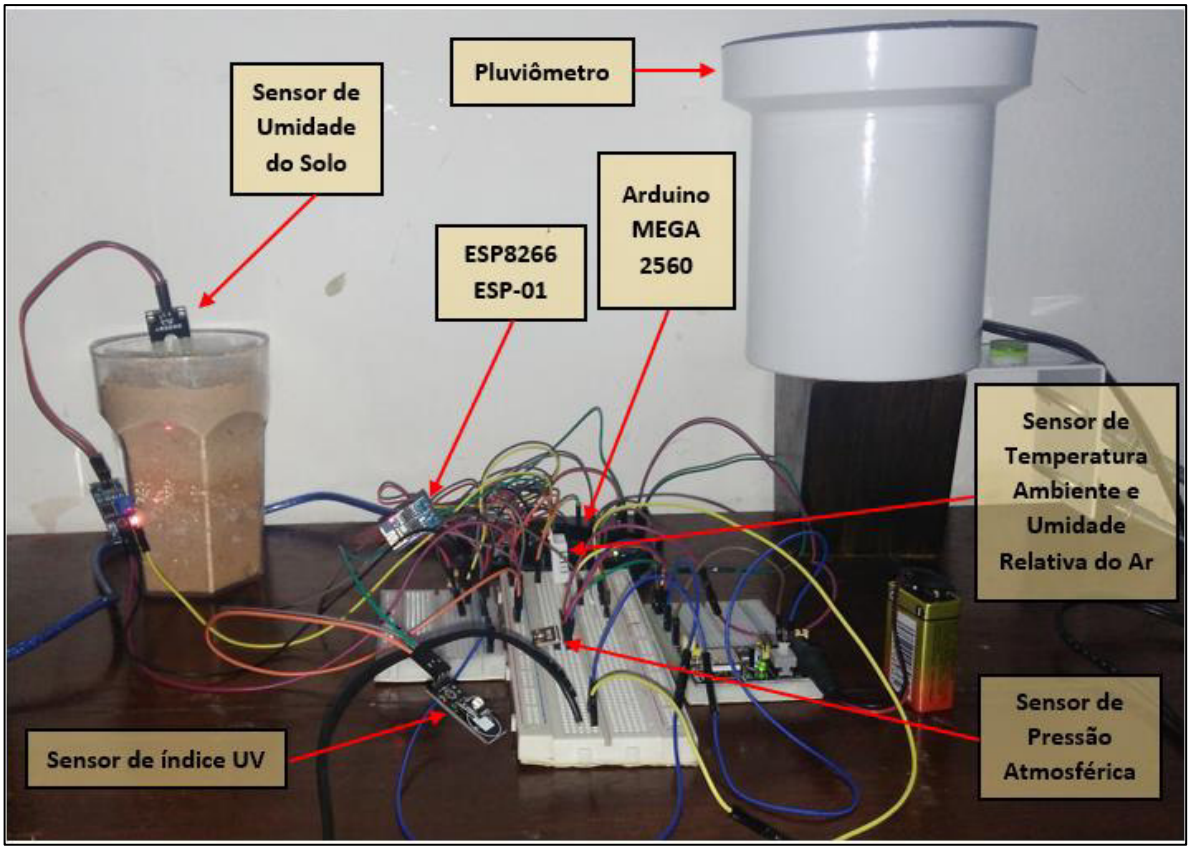
Figura 23: Circuito final do *hardware* da Estação Meteorológica



Fonte: Autor

Na Figura 24, pode-se visualizar o protótipo desenvolvido para a estação meteorológica.

Figura 24: *Hardware* implementado para a Estação Meteorológica



Fonte: Autor

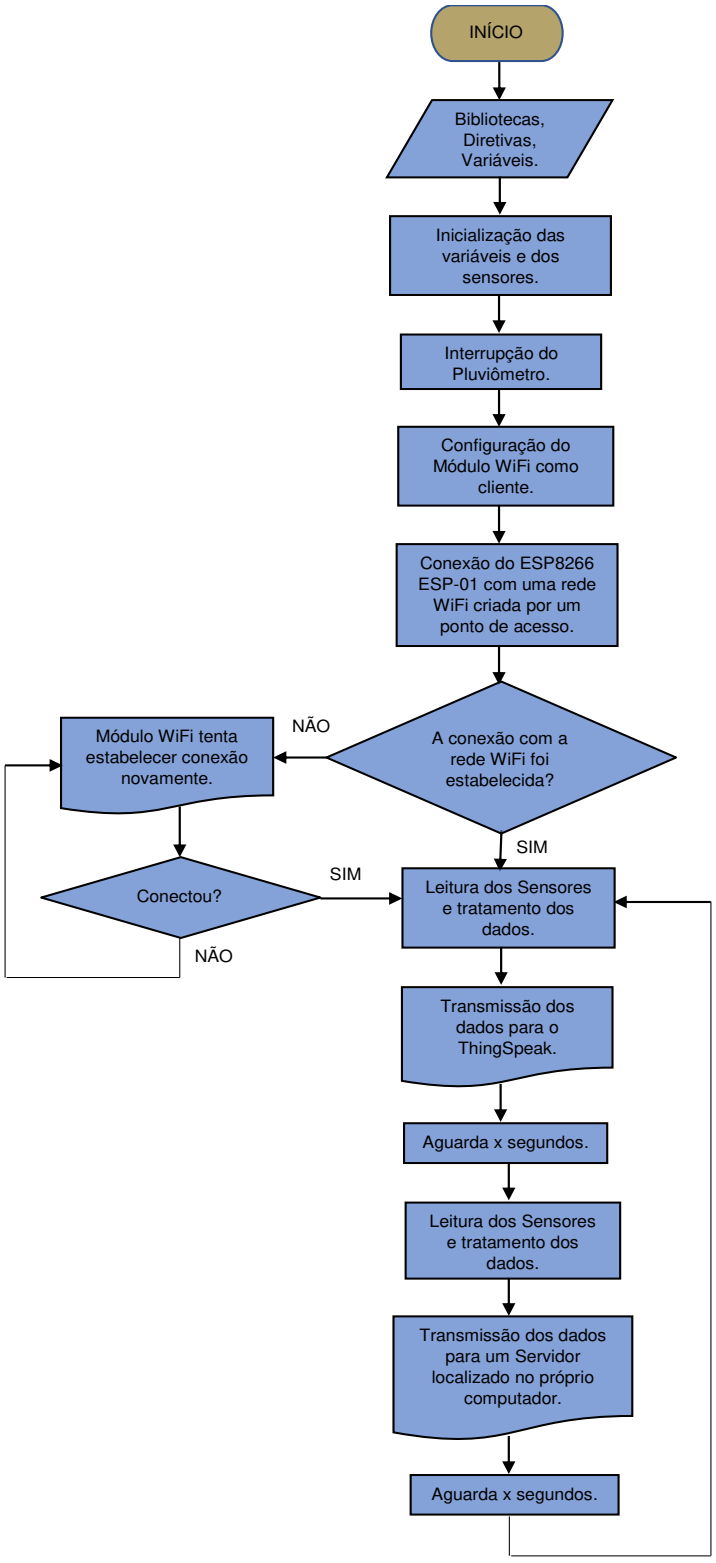
3.3 Desenvolvimento do *Software*

O *software* desenvolvido para a estação meteorológica foi dividido em duas etapas:

- Implementação do código no IDE do Arduino, sendo que as funções do mesmo são: leitura, processamento e envio dos dados meteorológicos para o ThingSpeak e para um servidor *web* localizado no próprio computador;
- Implementação do *script* em PHP no *Notepad ++*, as funções do mesmo são: obter e inserir os valores dos sensores em uma tabela localizada em um banco de dados MySQL. Esse *script* irá executar em um servidor http localizado no próprio computador, o Apache;

Através do fluxograma mostrado na Figura 25, pode-se visualizar a sequência de tarefas realizadas no código desenvolvido no IDE do Arduino.

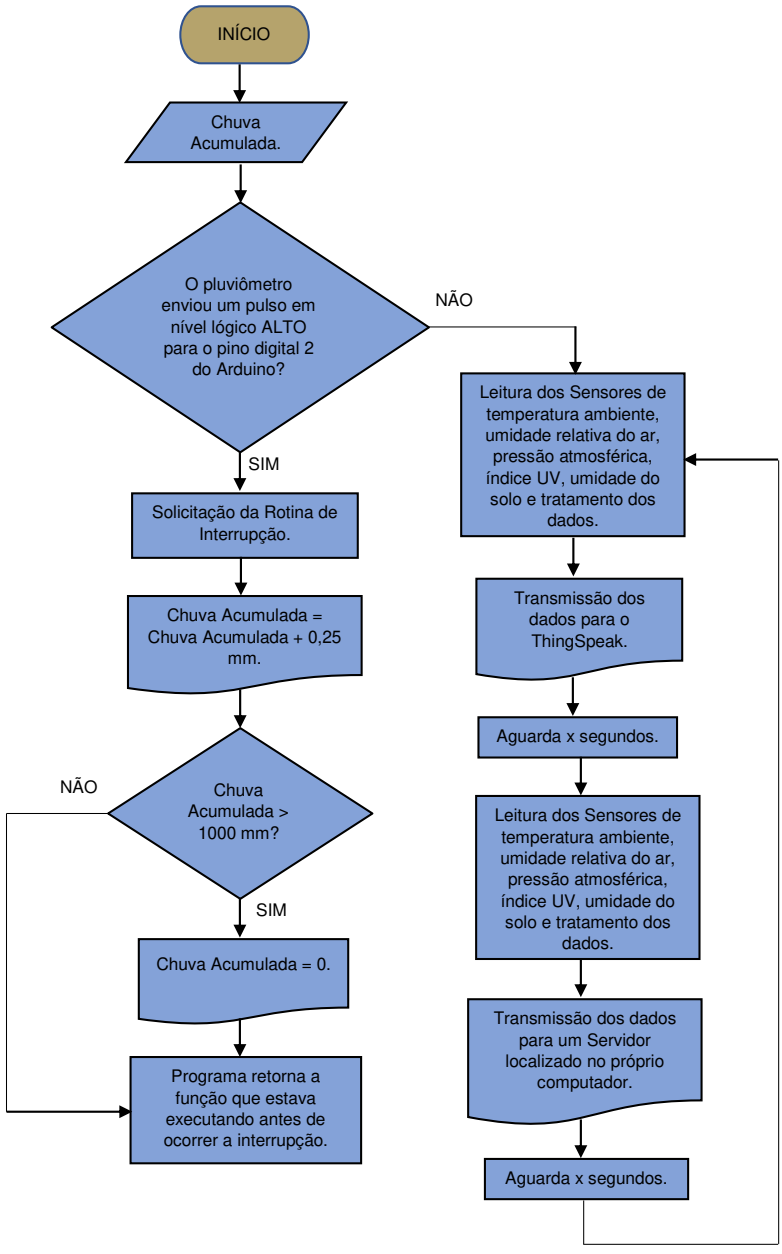
Figura 25: Fluxograma do código desenvolvido no IDE do Arduino



Fonte: Autor

Conforme dito no Capítulo 2, o pluviômetro utilizado nesse trabalho envia um pulso para o pino digital do Arduino sempre que 0,25 mm de chuva é identificado pela báscula que compõe o mesmo, logo, foi necessário utilizar um dos pinos de interrupção externa disponível no microcontrolador, de tal forma que sempre que um pulso em nível lógico ALTO for enviado pelo pluviômetro, o código imediatamente irá adicionar 0,25 a um contador de chuva acumulada. Na Figura 26, pode-se visualizar o fluxograma da sequência de tarefas executadas no código implementado no IDE do Arduino, as mesmas são referentes a rotina de interrupção do pluviômetro.

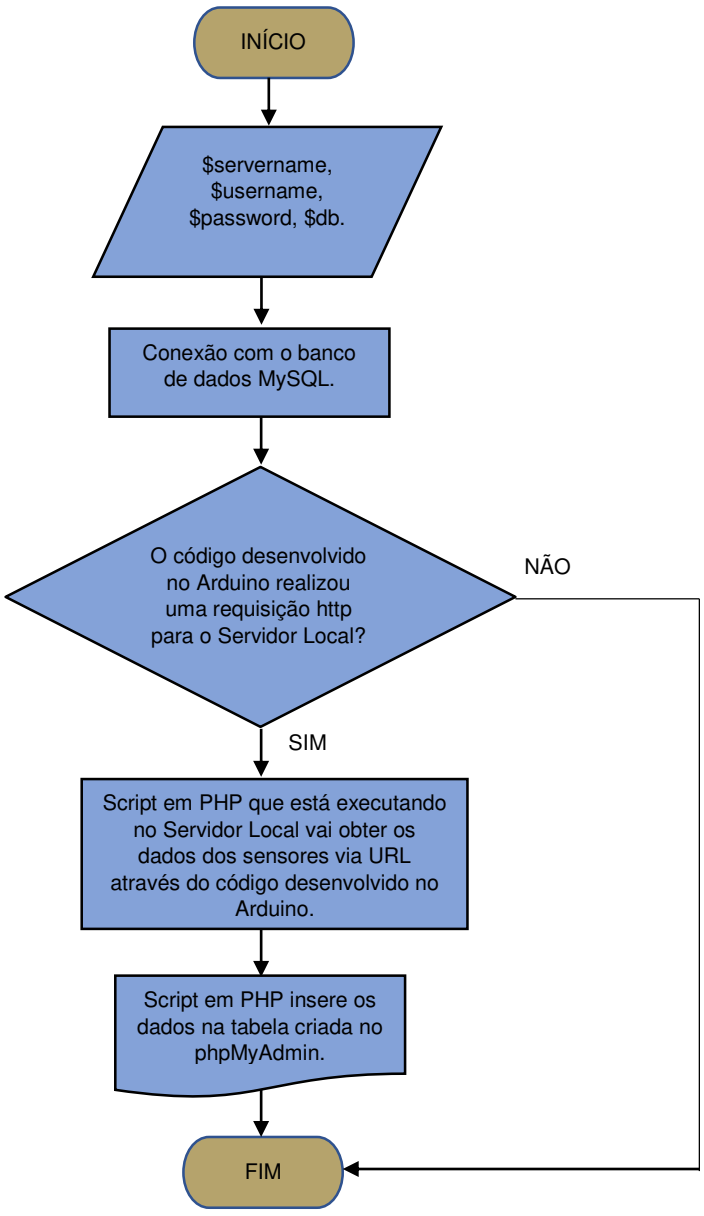
Figura 26: Fluxograma da rotina de interrupção do Pluviômetro



Fonte: Autor

Na Figura 27, pode-se visualizar o fluxograma que representa a sequência de tarefas relacionadas com o *script* em PHP implementado.

Figura 27: Fluxograma do *script* em PHP



Fonte: Autor

A seguir serão descritas detalhadamente todas as etapas relacionadas com a implementação do *software* para a Estação Meteorológica.

3.3.1 Inclusão das Bibliotecas

Conforme dito no Capítulo 1 é possível incluir no ambiente de programação do Arduino bibliotecas implementadas por outros desenvolvedores, dessa forma a programação torna-se menos complexa. Para esse código as seguintes bibliotecas foram utilizadas:

- DHT.h: possibilita utilizar as funções do sensor DHT22;
- Wire.h: permite a comunicação de dispositivos que utilizam o I2C como protocolo de comunicação;
- Adafruit_Sensor.h: possibilita utilizar as funções do sensor BMP280;
- Adafruit_BMP280.h: possibilita utilizar as funções do sensor BMP280.

A Figura 28 mostra o trecho do código com a inclusão das bibliotecas.

Figura 28: Bibliotecas utilizadas no código desenvolvido no IDE do Arduino

```
#include <DHT.h>  // Biblioteca para utilizar as funções do sensor DHT22
#include <Wire.h> // Biblioteca para permitir a comunicação de dispositivos que utilizam I2C
#include <Adafruit_Sensor.h> //Biblioteca para utilizar as funções do sensor BMP280
#include <Adafruit_BMP280.h> //Biblioteca para utilizar as funções do sensor BMP280
```

Fonte: Autor

3.3.2 Inicialização dos Sensores e Configuração do Pino de Interrupção

Nesse trecho do código é verificado se os pinos do sensor BMP280 foram conectados corretamente, é ativado o resistor de *pull up* referente ao pino configurado para acionar a interrupção do pluviômetro e também é inicializado o sensor DHT22. A Figura 29 mostra o trecho do código com a inicialização dos sensores e configuração do pino de interrupção do pluviômetro.

Figura 29: Inicialização de sensores e ativação do resistor de PULL UP

```
 // Verificando a conexão do sensor BMP280

Serial.println("Testando modulo BMP280");

 if (!sensor_bmp.begin())
 {
   Serial.println("Sensor não encontrado. Verificar as conexoes!");
   while (1);
 }

//

 pinMode(pinochuva, INPUT_PULLUP); // configura pinochuva como entrada e com resistor de
                                   // de PULLUP ativado
 dht.begin();          // inicializando sensor DHT22
```

Fonte: Autor

3.3.3 Conexão do ESP8266 modelo ESP01 com uma rede WiFi

Na função *setup* do código implementado no IDE do Arduino, a conexão do módulo com a rede WiFi é estabelecida. A porta serial escolhida para a comunicação entre o Arduino e o módulo foi a Serial2, como pode-se visualizar no código abaixo a velocidade de comunicação estabelecida foi de 115200 bps. O comando AT +CWMODE=1 configura o ESP01 como cliente para que o mesmo possa estabelecer a conexão com uma rede WiFi criada por um ponto de acesso, já o comando AT +CWJAP estabelece a conexão do módulo com a rede que possui a senha e o SSID informados no início do código. A Figura 30 mostra o trecho do código responsável pela conexão do ESP8266 modelo ESP01 com uma rede WiFi.

Figura 30: Conexão do ESP8266 modelo ESP01 com uma rede WiFi

```
Serial2.begin(115200);  // Configurando baud rate entre o Arduino e o ESP8266 ESP-01

Serial2.println("AT+CWMODE=1");   // Definindo o ESP8266 como cliente

RespostaCom(1000);  // Chamada da função RespostaCom

Serial2.println("AT+CWJAP=\""+ssid+"\",\""+password+"\"");  // Comando para conectar o ESP8266 ESP-01
                                                           // ao Access Point através do SSID
                                                           // e da senha informada no início do programa
RespostaCom(5000); // Chamada da função RespostaCom

if (DEBUG) Serial.println("Completo");   // Imprime no Serial Monitor que a conexão foi estabelecida
```

Fonte: Autor

3.3.4 Definição do *attachInterrupt* na função Setup

Na função *setup* também foi configurado o *attachInterrupt*. O primeiro parâmetro do *attachInterrupt* corresponde ao número do pino que irá acionar a interrupção externa, o segundo refere-se a função que será acionada sempre que ocorrer uma interrupção e o terceiro corresponde ao modo que a interrupção será acionada. No caso da configuração do circuito desse pluviômetro como mostrado no Capítulo 2, quando as lâminas do *reed switch* se atraem o circuito é fechado e um pulso em nível lógico ALTO é enviado para o pino digital do Arduino, logo, o modo de acionamento da interrupção deverá ser o RISING, ou seja, quando o valor do pino digital do Arduino muda de nível lógico baixo para nível lógico alto. A Figura 31 mostra o trecho do código com a definição do *attachInterrupt* na função *setup*.

Figura 31: Definição do attachInterrupt

```
attachInterrupt(digitalPinToInterrupt(pinochuva), qtdchuva, RISING); // Interrupção do pluviômetro
```

Fonte: Autor

3.3.5 Função de Interrupção do Pluviômetro

Sempre que houver uma interrupção a função qtdchuva será acionada e por conseguinte uma outra função também será solicitada com o objetivo de armazenar em um contador a quantidade de chuva acumulada. Para evitar contagem desnecessária uma verificação de tempo decorrido foi implementada no código. A variável **<agora>** armazena o tempo decorrido em microsegundos desde que o processador foi inicializado e a variável **<auxAgora>** armazena o último tempo decorrido em microsegundos no qual ocorreu uma interrupção do pluviômetro. Portanto, a função que contém o contador que acumula a quantidade de chuva só será solicitada se a diferença entre essas variáveis for maior do que 200000 microsegundos. A Figura 32 mostra o trecho do código com a função de interrupção do pluviômetro.

Figura 32: Função de Interrupção do Pluviômetro

```
// Função solicitada por uma interrupção do pluviômetro
void qtdchuva(){

   agora = micros();
   tempoult= abs(agora - auxAgora);

   if(tempoult> 200000L){  // o L é necessário, pois a variável tempoult
                           // foi declarada como long
   IncmmChuva();          // chamada da função com contador de chuva acumulada
   auxAgora = agora;
   }

}
```

Fonte: Autor

3.3.6 Função com contador que armazena chuva acumulada

Essa função é solicitada para adicionar 0,25 mm a variável mmChuva se ocorrer uma interrupção e se o tempo entre interrupções for maior do que 200000 microsegundos. Na Figura 33, pode-se visualizar a função com o contador que armazena a chuva acumulada.

Figura 33: Função Chuva Acumulada

```
// Função para armazenar chuva acumulada e zerar contador caso
// a quantidade de chuva acumulada seja > 1000 mm

void IncmmChuva(){

mmChuva =   mmChuva + 0.25;  // contador para armazenar chuva acumulada

 if (mmChuva > 1000)
    mmChuva = 0;     // zera contador mmChuva caso > 1000 mm
}
```

Fonte: Autor

3.3.7 Leitura e tratamento dos dados recebidos

A leitura e tratamento dos dados dos sensores é realizada na função *loop*. Essa tarefa é executada duas vezes. A primeira para enviá-los para o ThingSpeak e a segunda para enviá-los para o *web service* localizado no próprio computador. O código completo da função loop pode ser visualizado no Apêndice A desse trabalho. Na Figura 34, pode-se visualizar a leitura e tratamento dos dados meteorológicos detectados pelos sensores.

Figura 34: Leitura e tratamento dos dados recebidos

```
void loop() {

float t = 0;     // declaração da variável para receber o valor da temperatura
float h = 0;     // declaração da variável para receber o valor de umidade
float an_InUv = 0; // declaração da variável para receber o valor do pino analógico de índice UV
float I_UV  = 0; // Recebe Valor de índice UV após tratamento dos dados
float mVolts = 0; // Recebe valor em mV, correspondente ao resultado do valor
                  //  recebido do conversor analógico digital para mV
float p = 0; // recebe valor de pressão atmosférica

p = sensor_bmp.readPressure(); // Leitura do valor de pressão
t = dht.readTemperature();     // Leiturra do valor de temperatura ambiente
h = dht.readHumidity();        // Leitura do valor de umidade relativa do ar

// Início verificação do valor de índice Uv

an_InUv = analogRead(analog); // Ler valor presente no pino analog configurado como
                              // entrada analógica

    // Coversor A/D do microcontrolador MEGA 2560 possui 10 bits de resolução
    // logo o valor retornado por an_InUv estará na faixa entre 0 e 1023
    // de acordo com o valor presente no pino.
    // Valor de tensão de referência para o conversor A/D = 5 V

    mVolts = (an_InUv * (5.0 / 1023.0)) * 1000;  // conversão do cálculo do conversor analógico
                                                 //  digital para mV
   // Fim cálculo conversão para mV
```

Fonte: Autor

3.3.8 Envio dos dados para o ThingSpeak

Nesse trecho do código inicialmente é aberta uma conexão HTTP com o ThingSpeak. Através do comando "AT+CIPSTART= \ "TCP\ ", \ "" ; "184.106.153.149"; "\ ", 80"; aponta-se para a URL dessa plataforma. Através da instrução GET os dados são enviados para o canal que possui a chave especificada no início do código. Na Figura 35, pode-se visualizar o trecho do código responsável pelo envio dos dados climáticos para o ThingSpeak.

Figura 35: Envio dos dados para o ThingSpeak

```
boolean EscreverThingSpeak(float value1, float value2, float value3, int value4, double value5, float value6){
  String cmd = "AT+CIPSTART=\"TCP\",\"";  // String para iniciar uma conexão TCP com um endereço,
                                          //  neste caso será para o IP da plataforma ThingSpeak
  cmd += "184.106.153.149";      // IP do servidor ThingSpeak
  cmd += "\",80";                // Número da porta de destino dos dados no endereço 184.106.153.149
  Serial2.println(cmd); // imprime string cmd na porta Serial2
                        //(selecionada para comunicação entre o Arduino e o ESP8266)
  if (DEBUG) Serial.println(cmd);
  if(Serial2.find("Error")){   // Verifica se no Buffer de entrada da Serial2 tem uma string = "Error",
                               // se sim, retorna TRUE,caso contrário, retorna FALSE
  if (DEBUG) Serial.println("AT+CIPSTART error"); // imprime "AT+CIPSTART error" no Serial
                                                  //Monitor caso tenha encontrado a string Error na Serial2
  return false;
  }

  String getStr = "GET /update?api_key=";   // String que ao final será composta pela instrução GET,
                                            // chave do canal criado para receber os dados meteorológicos
                                            // e os valores dos sensores recebidos e processados pelo
                                            // Arduino
  getStr += apiKey; // chave do canal
  getStr +="&field1=";
  getStr += String(value1);  // corresponde ao valor de temperatura
  getStr +="&field2=";
  getStr += String(value2); // corresponde ao valor de umidade relativa do ar
  getStr +="&field3=";
  getStr += String(value3); // corresponde ao valor de índice UV
  getStr +="&field4=";
  getStr += String(value4);  // corresponde ao valor de umidade do solo
  getStr +="&field5=";
  getStr += String(value5); // corresponde ao valor de chuva acumulada
  getStr +="&field6=";
  getStr += String(value6); // corresponde ao valor de Pressão Atmosférica
  // ...
  getStr += "\r\n\r\n";

  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length()); // Nesta etapa a string cmd é composta pelo comando AT+CIPSEND=
                                  // e o tamanho da string getStr
  Serial2.println(cmd);         // Imprime na porta Serial2 a string cmd
  if (DEBUG)  Serial.println(cmd);  // imprime no Serial Monitor a string cmd
```

Fonte: Autor

3.3.9 Envio dos dados para um *web service* localizado no próprio computador

Nesse trecho do código é aberta uma conexão com o *local host* que está sendo utilizado através do XAMPP, portanto, deve ser informado para a string cmd o endereço IP do *local host*. Através da função GET os dados são enviados para o diretório especificado em getStr. Na Figura 36, pode-se visualizar o trecho do código responsável pelo envio dos dados climáticos para um *web service* localizado no próprio computador.

Figura 36: Envio dos dados para um *web service* localizado no próprio computador

```
boolean httpGet(float t, float u, float i, int us, double mC, float p)
{

String cmd = "AT+CIPSTART=\"TCP\",\""; // String para iniciar uma conexão TCP com um endereço,
                                       // neste caso será para o IP do localhost que está
                                       // sendo utilizado através do XAMPP = endereço IPv4
  cmd += "192.168.0.8";               // IP do localhost
  cmd += "\",8095";                   // Número da porta de destino dos dados no endereço
                                      // IP do localhost
  Serial2.println(cmd);               // Imprime string cmd na porta Serial2
                                      // (selecionada para comunicação entre o Arduino e o ESP8266)
  if (DEBUG) Serial.println(cmd);     // Imprime no Serial Monitor a string cmd
  if(Serial2.find("Error")){          // Verifica se no Buffer de entrada da Serial2 tem uma string = "Error",
                                      // se sim, retorna TRUE,caso contrário, retorna FALSE
    if (DEBUG) Serial.println("AT+CIPSTART error"); // imprime "AT+CIPSTART error" no Serial
                                                   //Monitor caso tenha encontrado a string Error na Serial2
    return false;
  }

  delay(1000);

// GET STRING

String getStr = "GET /arduino/salvardados.php?";  //String que ao final será composta pela instrução GET,
                                                  // pasta arduino que foi criada no XAMPP,
                                                  // o arquivo com script desenvolvido em php que irá
                                                 // inserir os valores dos sensores  para uma tabela criada
                                                // no banco de dados e os valores das medições dos sensores
getStr += "T=";
getStr += String(t); // corresponde ao valor de temperatura
getStr +="&Um=";
getStr += String(u); // corresponde ao valor de umidade relativa do ar
getStr +="&IndUV=";
getStr += String(i); // corresponde ao valor de índice UV
getStr +="&UmSolo=";
getStr += String(us);  // corresponde ao valor de umidade do solo
getStr +="&mmChuva=";
getStr += String(mC);  // corresponde ao valor de chuva acumulada
getStr +="&P=";
getStr += String(p);  // corresponde ao valor de pressão atmosférica
getStr += "\r\n\r\n";
Serial.println(getStr);
```

Fonte: Autor

3.3.10 *Script* em PHP

O *script* em PHP tem a função de obter os dados dos sensores via uma URL, os quais são enviados através do código desenvolvido no IDE do Arduino. Esses dados são inseridos em uma tabela criada em um banco de dados MySQL no phpMyAdmin. Através da declaração armazenada em $sql será inserido nos campos informados (`T`, `Um`, `IndUV`, `UmSolo`, `mmChv`, `P`) na tabela estação meteorológica, os valores dos dados diretamente de uma URL que possui o endereço desse arquivo em PHP. Na Figura 37, pode-se visualizar o *script* desenvolvido em linguagem de programação PHP.

Figura 37: *Script* em PHP

```php
<?php // Delimitador do código em PHP
      // dessa forma o servidor web entende que trata-se de um código
      // escrito em PHP
// Declaração de Variáveis
$servername = "localhost";  // endereço do servidor que o
                            // banco de dados está hospedado
$username = "root";    // nome do usuário cadastrado no banco de dados, sendo que
                      // quando a instalação do MySQL é realizada através
                     // do XAMPP, o usuário é criado por padrão como root
$password = "";     // por padrão quando a instalação do MySQL é realizada através
                   // do XAMPP não há senha para acessar o banco de dados
$db="estacao_meteorologica"; // Recebe o nome do banco de dados que será
                             // realizada a conexão
$conn = mysqli_connect($servername, $username, $password, $db); // A função mysqli_connect
                                                                // estabele a conexão com o
                                                                // banco de dados. A variável
                                                                // $conn recebe o retorno da conexão
                                                                // se a conexão for estabelecida a variável
                                                                // $conn retorna TRUE

if (!$conn) {                                          // Caso o valor retornado
                                                       // seja diferente de verdadeiro
                                                       // a função  mysqli_connect_error()
                                                       // irá retornar  uma string representando
                                                       // o último erro que aconteceu com a última
                                                       // chamada a função mysqli_connect()
die("Connection failed: " . mysqli_connect_error());
}
$sql="INSERT INTO `tabelaestacaometeorologica` (`T`, `Um`, `IndUV`, `UmSolo`, `mmChv`, `P`)
VALUES('".$_GET['T']."', '".$_GET['Um']."', '".$_GET['IndUV']."','".$_GET['UmSolo']."', '".
$_GET['mmChv']."', '".$_GET['P']."')";

$qry = mysqli_query($conn ,$sql); // Realizando uma consulta
                                  // no banco de dados
?>
```

Fonte: Autor

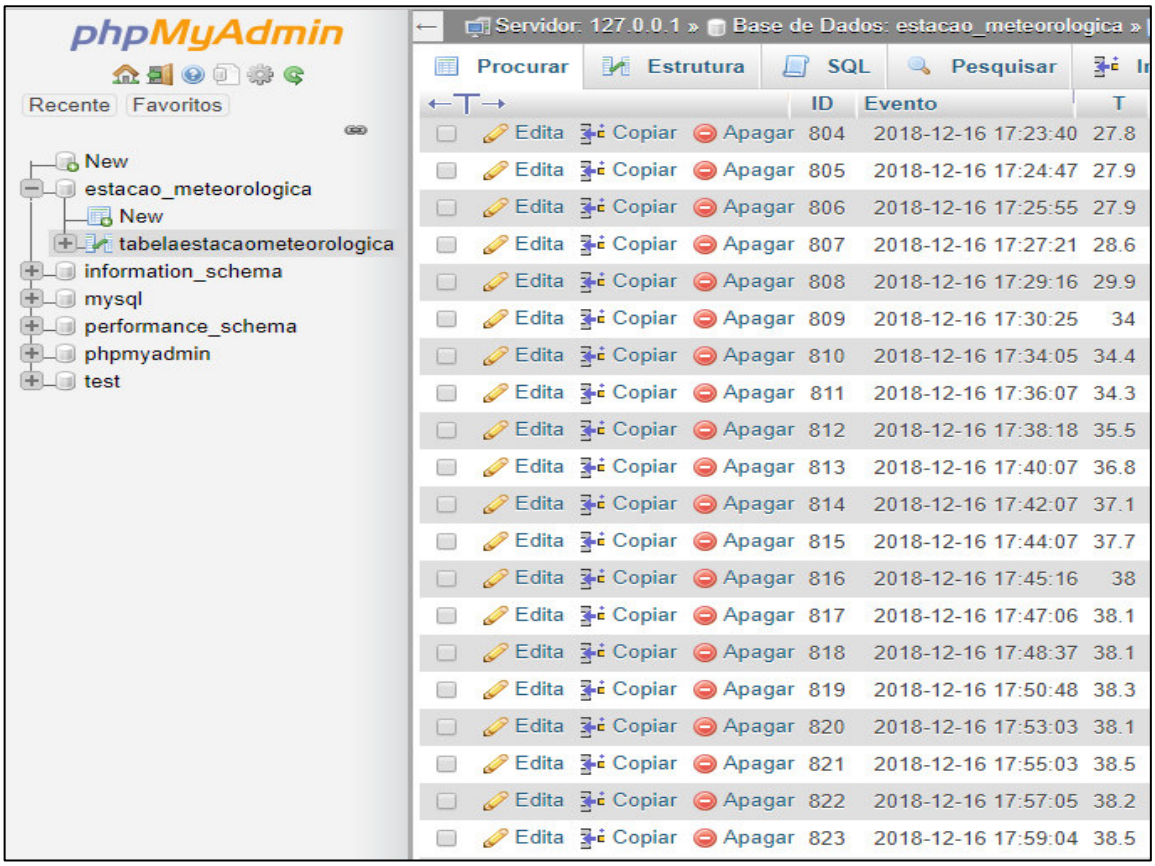
4 EXPERIMENTOS E RESULTADOS

Neste capítulo serão mostrados os testes realizados a fim de validar a eficiência e tempo de resposta dos sensores utilizados para o desenvolvimento desse projeto, assim como também verificar se não há perda de comunicação entre o ESP8266 modelo ESP-01 e a rede WiFi. Ao final do capítulo, mostra-se o custo para a implementação desse projeto.

4.1 Teste do Sensor de Temperatura Ambiente e Umidade Relativa do Ar

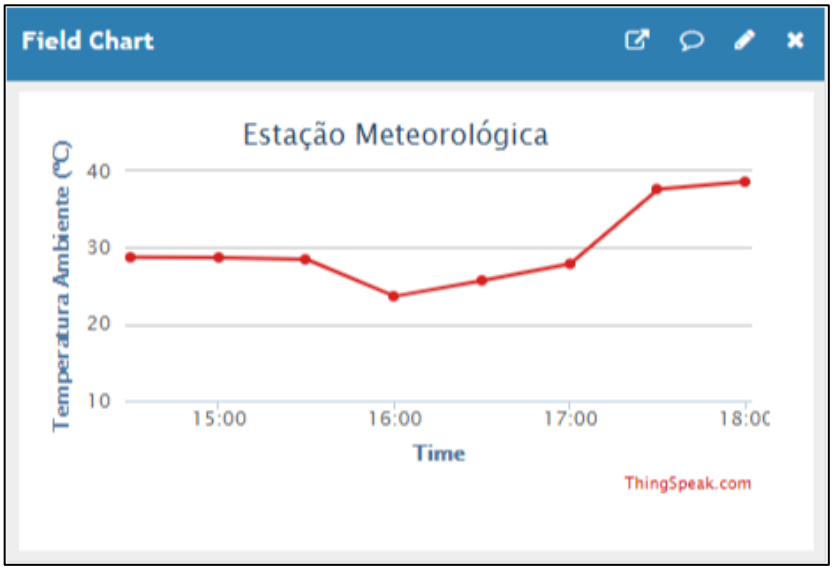
Para testar a variável temperatura foi aproximado um ferro de solda do sensor DHT22. Como mostra-se na Figura 38, rapidamente o sensor respondeu ao aumento de temperatura. Na Figura 39 é possível visualizar médias de 30 min antes e depois de aproximar o ferro de solda do sensor, a aproximação foi realizada a partir das 17 horas e 30 minutos.

Figura 38: Dados do sensor de temperatura armazenados na base de dados



Fonte: Autor

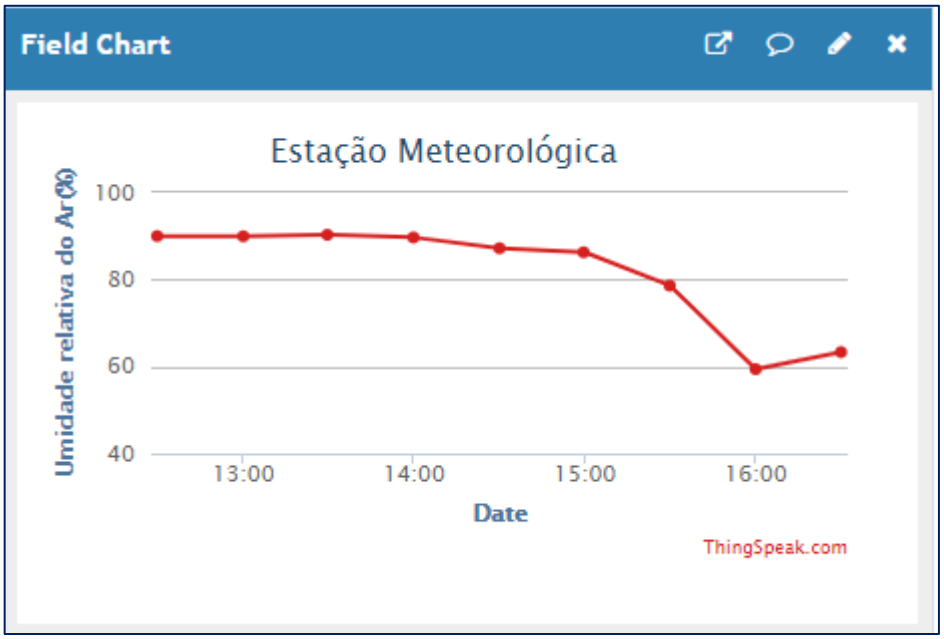
Figura 39: Dados do sensor de temperatura no ThingSpeak



Fonte: Autor

Para testar a variável umidade relativa do ar um dos testes realizados foi colocar o sistema em um ambiente com ar-condicionado, sabe-se que o mesmo retira a umidade do ar, portanto, é possível verificar a resposta do sensor para um ambiente com ar-condicionado e em um ambiente sem esse equipamento. Na Figura 40, pode-se visualizar o gráfico no ThingSpeak recebendo a resposta do DHT22 a uma variação de umidade.

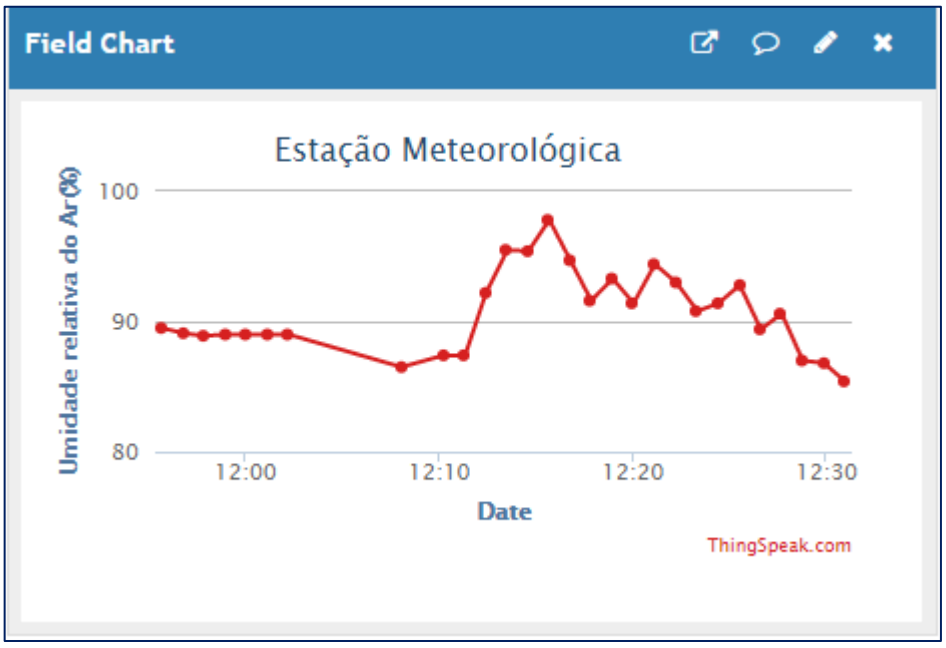
Figura 40: Dados do sensor de umidade em ambiente com ar-condicionado



Fonte: Autor

O sensor de umidade relativa do ar também foi exposto a um outro cenário, foi adicionado ao local em que o mesmo estava um umidificador de ar. A partir das 12 horas e 12 minutos o umidificador foi ligado. Na figura 41, pode-se visualizar que o sensor rapidamente respondeu ao aumento da umidade do ambiente em que estava inserido.

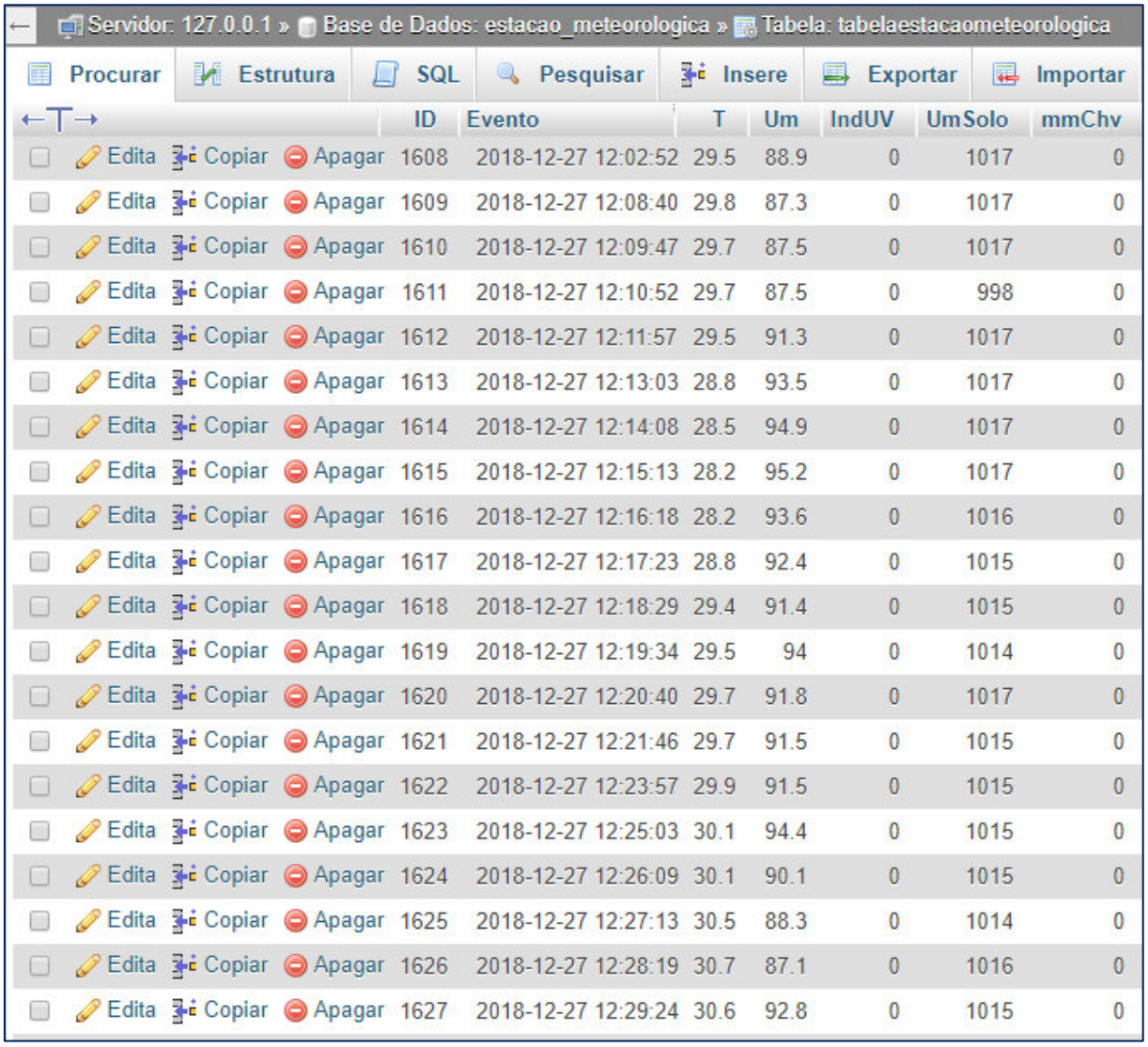
Figura 41: Dados do sensor de umidade em ambiente com umidificador de ar



Fonte: Autor

A resposta do sensor DHT22 a variação de umidade relativa do ar também pode ser visualizada na tabela criada no banco de dados MySQL na coluna nomeada como **Um**, conforme mostrado na Figura 42.

Figura 42: Dados do sensor de umidade armazenados na base de dados



Servidor: 127.0.0.1 » Base de Dados: estacao_meteorologica » Tabela: tabelaestacaometeorologica

Procurar | Estrutura | SQL | Pesquisar | Insere | Exportar | Importar

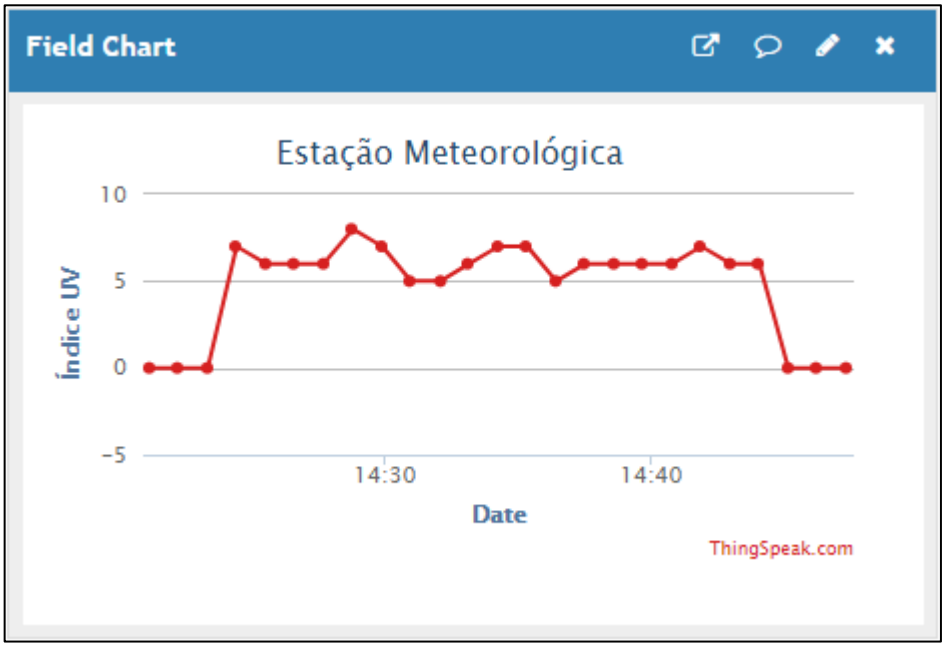
	ID	Evento	T	Um	IndUV	UmSolo	mmChv
Edita Copiar Apagar	1608	2018-12-27 12:02:52	29.5	88.9	0	1017	0
Edita Copiar Apagar	1609	2018-12-27 12:08:40	29.8	87.3	0	1017	0
Edita Copiar Apagar	1610	2018-12-27 12:09:47	29.7	87.5	0	1017	0
Edita Copiar Apagar	1611	2018-12-27 12:10:52	29.7	87.5	0	998	0
Edita Copiar Apagar	1612	2018-12-27 12:11:57	29.5	91.3	0	1017	0
Edita Copiar Apagar	1613	2018-12-27 12:13:03	28.8	93.5	0	1017	0
Edita Copiar Apagar	1614	2018-12-27 12:14:08	28.5	94.9	0	1017	0
Edita Copiar Apagar	1615	2018-12-27 12:15:13	28.2	95.2	0	1017	0
Edita Copiar Apagar	1616	2018-12-27 12:16:18	28.2	93.6	0	1016	0
Edita Copiar Apagar	1617	2018-12-27 12:17:23	28.8	92.4	0	1015	0
Edita Copiar Apagar	1618	2018-12-27 12:18:29	29.4	91.4	0	1015	0
Edita Copiar Apagar	1619	2018-12-27 12:19:34	29.5	94	0	1014	0
Edita Copiar Apagar	1620	2018-12-27 12:20:40	29.7	91.8	0	1017	0
Edita Copiar Apagar	1621	2018-12-27 12:21:46	29.7	91.5	0	1015	0
Edita Copiar Apagar	1622	2018-12-27 12:23:57	29.9	91.5	0	1015	0
Edita Copiar Apagar	1623	2018-12-27 12:25:03	30.1	94.4	0	1015	0
Edita Copiar Apagar	1624	2018-12-27 12:26:09	30.1	90.1	0	1015	0
Edita Copiar Apagar	1625	2018-12-27 12:27:13	30.5	88.3	0	1014	0
Edita Copiar Apagar	1626	2018-12-27 12:28:19	30.7	87.1	0	1016	0
Edita Copiar Apagar	1627	2018-12-27 12:29:24	30.6	92.8	0	1015	0

Fonte: Autor

4.2 Teste do Sensor de Índice UV

Para testar a resposta do sensor de índice UV, o mesmo foi exposto a principal fonte de radiação ultravioleta, a luz do sol. Como pode-se visualizar na Figura 43, rapidamente o sensor respondeu a essa exposição. Após ser retirado de qualquer cenário com a presença de uma fonte de luz ultravioleta, rapidamente o mesmo indicou 0 de radiação UV.

Figura 43: Dados do sensor de índice UV

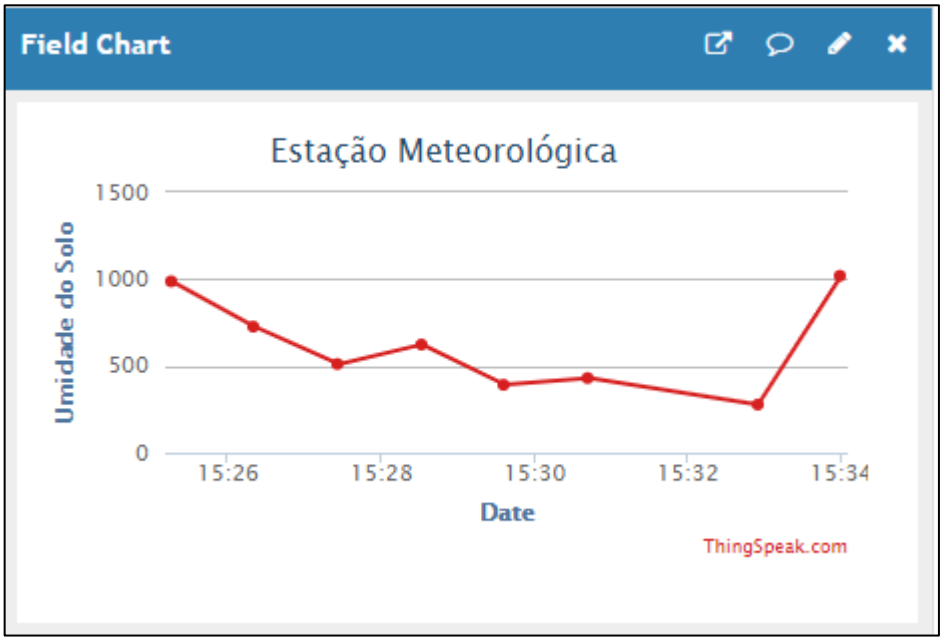


Fonte: Autor

4.3 Teste do Sensor de Umidade do Solo

Para testar a resposta do sensor de umidade do solo foi acrescentado água ao solo gradativamente, de tal forma que como este sensor é formado por uma haste com dois eletrodos, ao aplicar uma determinada corrente, pode-se estimar a umidade do solo. Quanto mais úmido o mesmo estiver maior será a corrente entre os eletrodos, já que uma maior quantidade de água será absorvida, entretanto, menor será o valor inteiro retornado por meio do pino analógico do Arduino que estiver conectado com esse sensor e esse valor estará entre 0 e 1023, já que o Arduino possui conversor A/D com resolução de 10 bits. Na Figura 44, pode-se visualizar o gráfico no ThingSpeak recebendo a resposta do higrômetro para diferentes níveis de umidade.

Figura 44: Dados do sensor de umidade do solo

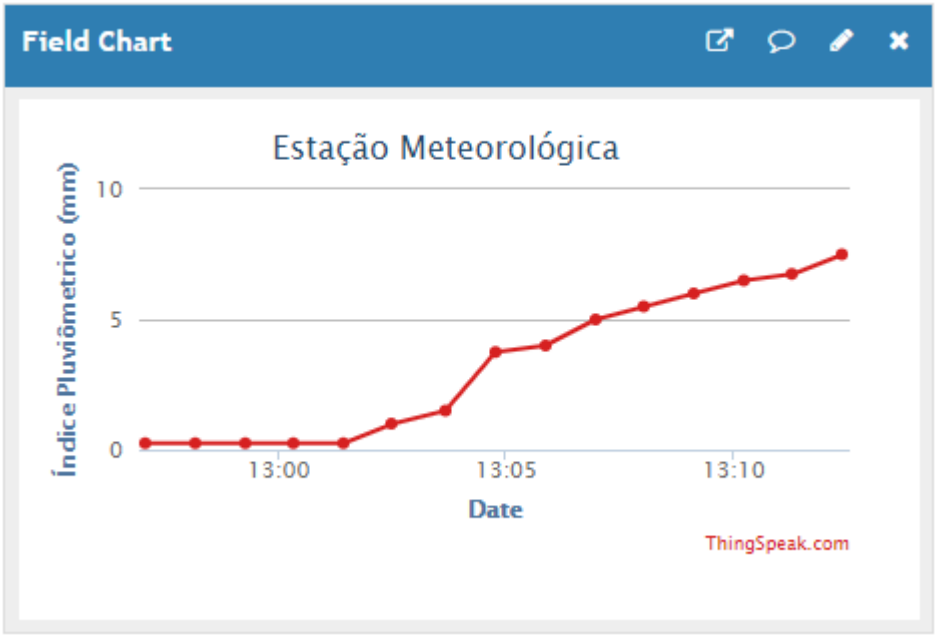


Fonte: Autor

4.4 Teste do Pluviômetro

A fim de testar a resposta do Pluviômetro foi adicionado 0,25 mm ao mesmo gradativamente, de tal forma que foi possível comprovar que os 0,25 mm são adicionados a variável de chuva acumulada somente quando há um movimento da báscula. Na Figura 45, pode-se visualizar o gráfico no ThingSpeak recebendo a resposta do pluviômetro ao longo do teste.

Figura 45: Dados do pluviômetro

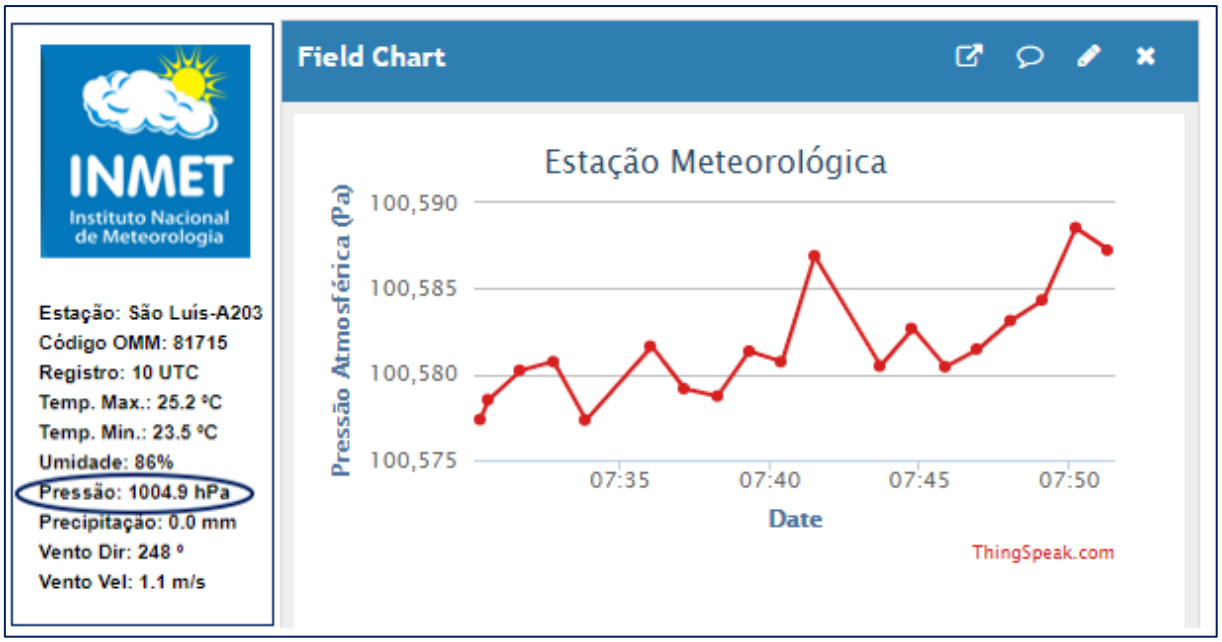


Fonte: Autor

4.5 Teste do Sensor de Pressão Atmosférica

Não há como simular a resposta do sensor BMP280 a uma variação da pressão atmosférica no ambiente em que o *hardware*/*software* da estação meteorológica implementada está instalado no momento, pois, sabe-se que ao nível do mar a pressão atmosférica sempre será da ordem de 1 atm, que corresponde a 101325 Pa, sendo assim, a pressão atmosférica medida pelo sensor foi de aproximadamente 100 570 Pa. Pode-se comparar este valor com o informado pelo INMET (Instituto Nacional de Meteorologia) para uma estação meteorológica automática também localizada em São Luís, o valor informado foi de 100490 Pa. Logo, pode-se comprovar que o sensor está indicando um resultado compatível com os dados do INMET. Na Figura 46, pode-se visualizar o valor de pressão atmosférica apresentado pelo INMET e o gráfico no ThingSpeak recebendo a resposta do sensor BMP280.

Figura 46: Dados do Sensor de pressão atmosférica e valor de pressão atmosférica informado pelo INMET

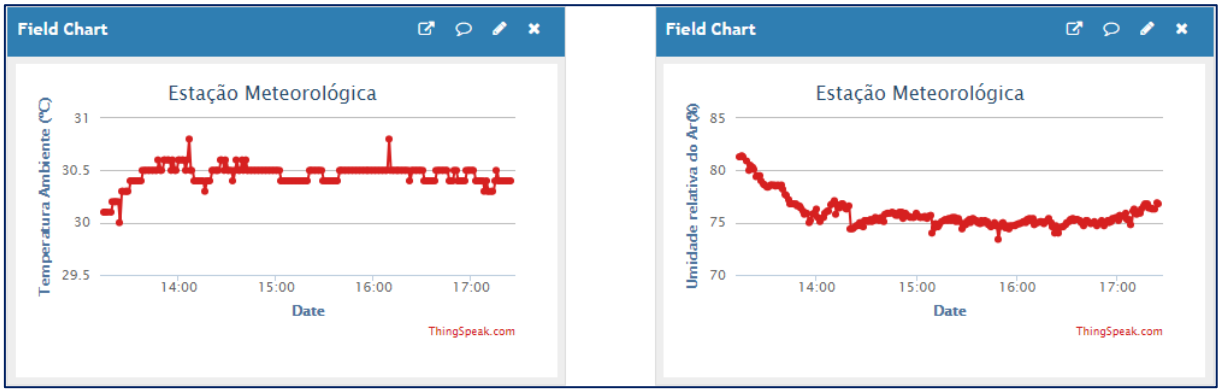


Fonte: Autor

4.6 Teste do módulo WiFi ESP8266 modelo ESP-01

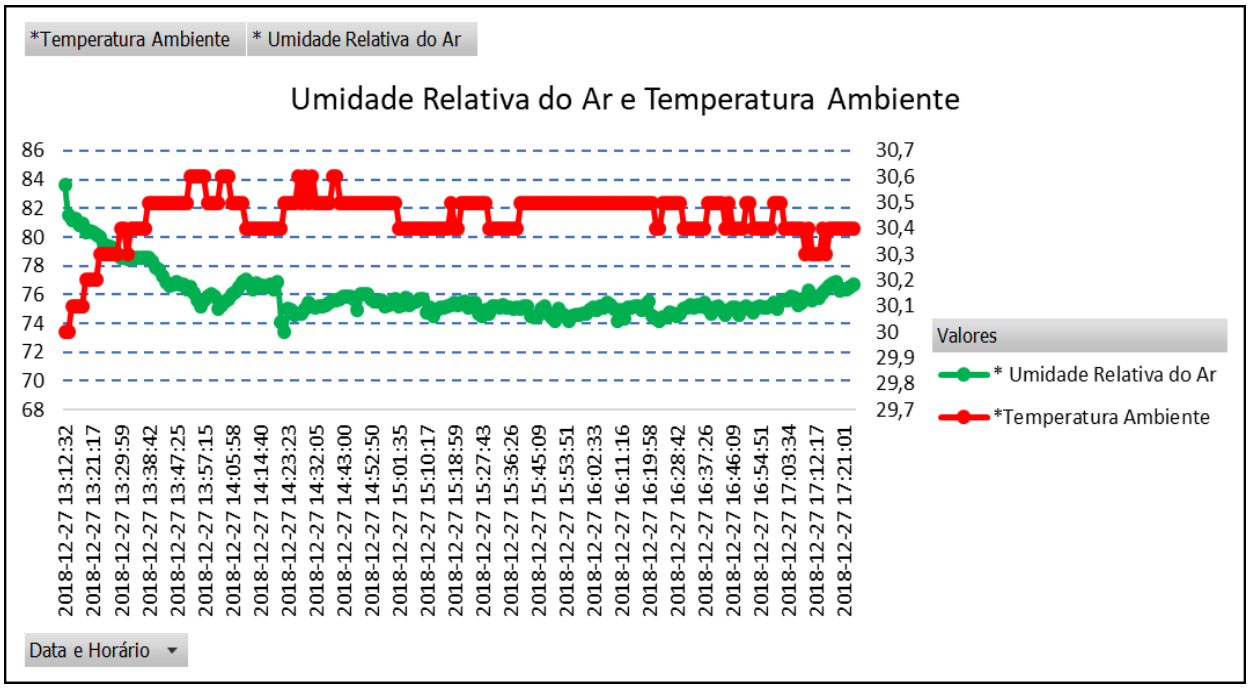
Para verificar a eficiência do módulo WiFi selecionado para implementação desse projeto, o sistema foi deixado ligado por um certo período de tempo. Como pode-se visualizar nas Figuras 47 e 48, os dados foram enviados tanto para o ThingSpeak quanto para o banco de dados criado, sem perda de conexão. O sistema ficou ligado por aproximadamente 4 horas ininterruptas.

Figura 47: Dados meteorológicos no ThingSpeak



Fonte: Autor

Figura 48: Dados meteorológicos do banco de dados

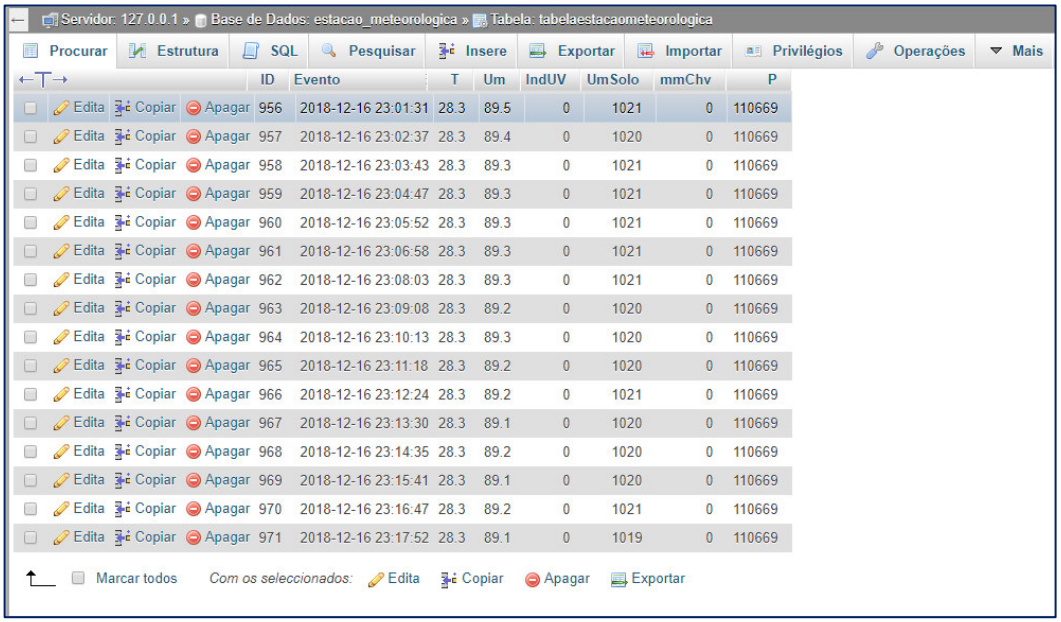


Fonte: Autor

4.7 Tabela criada no Bando de dados MySQL

Na Figura 49, pode-se visualizar a estrutura completa da tabela criada no phpMyAdmin, o parâmetro T corresponde a temperatura ambiente, Um, refere-se a umidade relativa do ar, IndUv ao índice de radiação ultravioleta, UmSolo a umidade do solo, mmChv aos mm de chuva acumulada e P corresponde a pressão atmosférica.

Figura 49: Tabela criada no phpMyAdmin



Fonte: Autor

4.8 Custo do Projeto

Na Tabela 3, pode-se visualizar o custo de cada componente utilizado para o desenvolvimento desse projeto.

Tabela 3 - Componentes utilizados no projeto

Componente	Quantidade	Valor
Arduino Mega 2560 R3	1	R$ 70,00
Sensor de Umidade do Solo	1	R$ 9,90
Sensor de índice UV	1	R$ 44,90
Sensor BMP280	1	R$ 18,00
Sensor DHT22	1	R$ 44,90
Pluviômetro	1	R$ 150,00
Módulo WiFi ESP8266 ESP-01	1	R$ 26,00
Protoboard 830 pontos	1	R$ 22,00
Resistores	3	R$ 0,30
Módulo Fonte Ajustável	1	R$ 12,00
Jumpers	96	R$ 24,00
Total		R$ 422,00

Fonte: Autor

5 CONCLUSÃO

Este projeto teve como principal objetivo a implementação de um protótipo para aquisição, processamento e envio de dados meteorológicos para uma plataforma de armazenamento de dados na nuvem, de tal forma que essas informações pudessem ser acessadas remotamente de qualquer lugar que tivesse acesso à internet. Uma outra finalidade do mesmo, foi aprender como implementar um banco de dados e utilizar a linguagem de programação PHP para inserir e manipular as informações na base de dados criada.

O ponto de partida para a decisão de implementar esse sistema foram as observações realizadas em ralação aos impactos causados devido a falta de monitoramento de algumas variáveis meteorológicas em muitas regiões do Brasil. Além disso, alguns dos dados que podem ser coletados por uma estação meteorológica possuem uma relação direta com a produtividade de diversos setores. Logo, convém por exemplo, monitorar a umidade do solo a fim de reduzir os gastos com água, energia elétrica, controlar o processo de irrigação. É também viável, monitorar o índice de radiação ultravioleta, já que essa variável está associada até mesmo com o aparecimento de câncer de pele, convém também acompanhar os dados de umidade relativa do ar, pois, tanto uma umidade muito alta quanto muito baixa pode causar problemas respiratórios. Da mesma maneira é praticável o monitoramento da precipitação acumulada, utilizando como referencial o fato de que grande parte dos desastres naturais estão associados com a variável chuva.

Pode-se perceber que a partir dos resultados evidenciados no Capítulo 4, os objetivos foram alcançados. Os sensores escolhidos para o desenvolvimento do projeto apresentaram precisão e tempo de resposta conforme o esperado. O módulo WiFi selecionado também se caracterizou como um componente eficiente, visto que para o período de tempo analisado, a conexão com o ponto de acesso não foi perdida, de tal forma que as informações foram enviadas em tempo real tanto para uma plataforma de armazenamento de dados na nuvem quanto para um servidor localizado no próprio computador do usuário e o custo para implementar o sistema proposto também foi baixo.

O aprendizado de como criar um banco de dados também foi comprovado, já que foi possível verificar no Capítulo 4 que os dados foram adequadamente

inseridos em uma tabela criada no phpMyAdmin através da interação entre a linguagem de desenvolvimento *web* PHP e o servidor *web*.

5.1 Sugestões para trabalhos futuros

Como sugestão para trabalhos futuros, é viável adicionar outros sensores ao *hardware* implementado, visto que como foi utilizada a placa Arduino MEGA 2560 como componente base, muitos pinos analógicos e digitais ficaram disponíveis para conectar outros sensores eletrônicos, como por exemplo: um pirômetro, heliógrafo ou um anemômetro. Uma outra sugestão é a criação de um produto, visto que nesse projeto foi desenvolvido o *hardware* e *software*, no qual o circuito permaneceu montado no protoboard. Pode-se comparar os resultados dos sensores com instrumentos de referência, também chamados de padrões, de tal forma a validar se os mesmos estão indicando valores na faixa de precisão informada pelos fabricantes nos *datasheets*, ou seja, essa proposta refere-se a realizar a calibração desses componentes.

REFERÊNCIAS

ADAFRUIT. **AM2302 Datasheet**. United States, 2018. Disponível em <https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf>. Acesso em: 28 junho 2018.

AOSONG. **AM2302 Product Manual**. China, 2018. Disponível em <https://img.filipeflop.com/files/download/Datasheet_DHT22_AM2302.pdf>. Acesso em: 08 abril 2018.

ARDUINO. **Arduino Mega**. 2018. Disponível em <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>. Acesso em: 08 abril 2018.

ARDUINO. **Introduction**. 2018. Disponível em <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 08 de abril de 2018.

ARDUINO E CIA. **Use o Sensor BMP280 para medir temperatura**. 2017. Disponível em <https://www.arduinoecia.com.br/2017/04/bmp280-pressao-temperatura-altitude.html>. Acesso em: 23 maio 2018.

ATMEL. **Datasheet ATmega2560**. United States, 2014.

AYOADE, J. **Introdução à Climatologia para os Trópicos**. (Santos, Trad.). São Paulo: Editora Bertrand Brasil S.A. 4 ed. 1996.

BANDERALI, M. **A importância do monitoramento da umidade do solo na agricultura**. 2010. Disponível em <http://www.diadecampo.com.br/zpublisher/materias/Materia.asp?secao=Gest%E3o&id=21142>. Acesso em: 30 maio 2018.

BANZI, M. **Primeiros Passos com o Arduino**. São Paulo: Editora Novatec. 2 ed. 2015.

CASTRO, S. **Sensores de Umidade: Caracterização e Desenvolvimento de Dispositivo Eletrônico**. Dissertação, Instituto Federal de Itajubá, 2011.

CURVELLO, A. **Apresentando o módulo ESP8266**. 2015. Disponível em <https://www.embarcados.com.br/modulo-esp8266/>. Acesso em: 21 maio 2018.

DU, W. **Resistive, Capacitive, Inductive, and Magnetic Sensor Technologies**. Florida: CRC Press, 2015.

ELMASRI, R.; NAVATHE, S. **Sistemas de banco de dados**. (Vieira, Trad.). São Paulo: Editora Pearson. 6 ed. 2011.

EPA. **Health Effects of UV Radiation**. 2017. Disponível em <https://www.epa.gov/sunsafety/health-effects-uv-radiation>. Acesso em: 27 maio 2018.

ESPRESSIF, S. **ESP8266 AT Instruction Set**. Shangai, 2018.

ESPRESSIF, S. **ESP8266 DataSheet**. Shangai, 2013.

FILIPEFLOP. **Módulo WiFi ESP8266 ESP-01**. [on-line] Disponível em <https://www.filipeflop.com/produto/modulo-wifi-esp8266-esp-01/>. Acesso em: 21 maio 2018.

FILIPEFLOP. **Placa Arduino MEGA 2560**. [on-line] Disponível em <https://www.filipeflop.com/produto/placa-mega-2560-r3-cabo-usb-para-arduino/>. Acesso em: 08 abril 2018.

FILIPEFLOP. **Sensor de umidade do solo**. [on-line] Disponível em <https://www.filipeflop.com/produto/sensor-de-umidade-do-solo-higrometro/?gclid=EAIaIQobChMI-Kbd9dTA3wIVwQSRCh2NCwSwEAAYASAAEgL5-fD_BwE>. Acesso em: 21 maio 2018.

HSI, S. **Reed Switch Application Notes**. United States, 2013. Disponível em<http://www.hsisensing.com/wp-content/uploads/2016/03/HSI_Reed_Switch_Applicaton_Notes_v12_2013.pdf.> Acesso em: 10 junho 2018.

KOLBAN, N. **Kolban's Book on ESP32 & ESP8266**. 2016. Disponível em <https://www.handsontec.com/pdf_learn/ESP8266_ESP32-oct2016.pdf>. Acesso em: 24 de maio de 2018.

LEENS, Frederic. **An Introduction to I2C and SPI Protocols**. 2009. Disponível em <https://ieeexplore.ieee.org/document/4762946/authors>. Acesso em: 23 maio 2018.

MADUREIRA JUNIOR, J. R.; SACILOTTI, A. C.; SACILOTTI, R. **Apostila - Desenvolvimento web com PHP**. 2017. Disponível em < http://www.perse.com.br/novoprojetoperse/BSU_Data/Books/N1506651597223/Amostra.pdf>. Acesso em: 21 maio 2018.

MCROBERTS, M. **Arduino Básico**. (Zanolli, R, Trad.). São Paulo: Editora Novatec, 1 ed. 2011.

SANTOS, João. **RADIAÇÃO ULTRAVIOLETA: ESTUDO DOS ÍNDICES DE RADIAÇÃO, CONHECIMENTO E PRÁTICA DE PREVENÇÃO A EXPOSIÇÃO NA REGIÃO ILHÉUS/ ITABUNA-BAHIA**. Dissertação, Universidade Estadual de Santa Cruz, Ilhéus, 2010.

SCHWARTZ, M. **Internet of things with ESP8266**. Reino Unido: Editora Packt Publishing. 1 ed. 2016.

SENSORTEC, B. **BMP280 Datasheet**. Alemanha, 2015. Disponível em < https://www.bosch-sensortec.com/bst/products/all_products/bmp280>. Acesso em: 20 abril de 2018

STRAUB, M. G. **Pluviômetro Arduino – Um sensor de chuva para Estação Meteorológica**. 2018. Disponível em <https://blog.usinainfo.com.br/pluviometro-arduino-um-sensor-de-chuva-para-estacao-meteorologica/>. Acesso em: 25 maio 2018.

THOMSEN, A. **Monitore sua planta usando Arduino**. 2016. Disponível em < https://www.filipeflop.com/blog/monitore-sua-planta-usando-arduino/>. Acesso em: 29 maio 2018.

TORRES, F.T.P; MACHADO, P. J.O. **Introdução à CLIMATOLOGIA**. Ubá: Editora Geographica, 2008.

VAISALA. **HUMICAP Sensors Datasheet**. Finlândia, 2018. Disponível em https://www.vaisala.com/sites/default/files/documents/HUMICAP-Sensors-Datasheet-B211748EN.pdf >. Acesso em: 26 julho 2018.

VAREJÃO, M. **Meteorologia e Climatologia**. Pernambuco: Versão Digital; 3 ed., 2005.

WILTRONICS. **UVM-30A Datasheet**. Austrália, 2018. Disponível em < https://www.wiltronics.com.au/wp-content/uploads/datasheets/ARD2-2062.pdf>. Acesso em: 21 maio 2018.

APÊNDICES

APÊNDICE A – Código desenvolvido no IDE do Arduino

```
//*********************************************************************************************
// DESCRIÇÃO DA APLICAÇÃO: Código para aquisição, processamento e envio
// dos dados meteorológicos do Arduino através do ESP8266 ESP-01 para uma
// plataforma de armazenamento de dados na nuvem, o ThingSpeak e para um
// web server que irá inserir as informações meteorológicas em um banco
// de dados MySQL.
// Última atualização em: 24/07/2018
// Autores: Patrícia Costa e exemplos do canal WR Kits, disponíveis em:
// www.youtube.com/channel/UCazAvTtoRlOrFDWDJDB2DKQ
//
//*********************************************************************************************
// Início do Programa
// Inclusão das Bibliotecas

#include <DHT.h>  // Biblioteca para utilizar as funções do sensor DHT22
#include <Wire.h> // Biblioteca para permitir a comunicação de dispositivos que
//utilizam I2C
#include <Adafruit_Sensor.h> // Biblioteca para utilizar as funções do sensor // BMP280
#include <Adafruit_BMP280.h> //Biblioteca para utilizar as funções do sensor BMP280

// DIRETIVAS para sensor DHT22

#define DHTPIN 5        // Diretiva que estabelece qual pino do Arduino está conectado ao
// pino de dados do sensor, neste caso é o pino 5
#define DHTTYPE DHT22 // Diretiva que estabelece que o sensor utilizado é o DHT22

//DIRETIVAS para ranges de umidade do solo

#define    Nivel_1      169   //Nomeando o valor 169 como Nivel_1
#define    Nivel_2      340   //Nomeando o valor 340 como Nivel_2
#define    Nivel_3      511   //Nomeando o valor 511 como Nivel_3
#define    Nivel_4      682   //Nomeando o valor 682 como Nivel_4
#define    Nivel_5      853   //Nomeando o valor 853 como Nivel_5

// DIRETIVA para mapeamento de hardware correspondente a umidade do solo

#define    analogi    A0   //Estabelece o pino analógico A0 como analogi

// DIRETIVA para mapeamento de hardware correspondente ao índice UV

#define   analog   1 // Estabelece o pino analógico 1 como analog

// Funções existentes nas bibliotecas utilizadas neste programa

DHT dht(DHTPIN, DHTTYPE);    // objeto utilizado para comunicação com o sensor DHT22
Adafruit_BMP280 sensor_bmp; // função nomeada como sensor_bmp existente na biblioteca
//Adafruit_BMP280

// Variáveis declaradas como volatile: associadas com a função solicitada
// em uma interrupção do pluviômetro
```

```
volatile unsigned long agora = 0; // armazena tempo decorrido em microsegundos desde que
// o processador foi inicializado
volatile unsigned long auxAgora = 0; //armazena último tempo decorrido em microsegundos
// no qual ocorreu uma interrupção do pluviômetro
volatile unsigned long tempoult = 0; // armazena diferença entre tempo decorrido atual e
 // último tempo decorrido no qual ocorreu uma interrupção do pluviômetro

//Declaração de Variáveis e atribuição de valor inicial

double mmChuva = 0;  // contador de chuva acumulada
int pinochuva = 2;  // ligação do pluviômetro com o pino D2 do Arduino
int adc2_value = 0x00;  // umidade do solo

boolean DEBUG=true;

// Chave de escrita do canal com os dados meteorológicos no ThingSpeak

String apiKey = "TMFBWX7ACEBROCQU";     // Digitar a chave de escrita do canal criado
//na plataforma ThingSpeak para receber os dados meteorológicos

// Estabeler conexão WiFi do ESP8266 ESP-01 com a rede criada por um ponto de acesso

String ssid="Loading";    // Digitar nome da rede WiFi disponível para acesso a conexão
String password ="epson32472430CX";  // Digitar a senha para o acesso a rede informada

// Função solicitada por uma interrupção do pluviômetro
void qtdchuva(){

  agora = micros();
  tempoult= abs(agora - auxAgora);

  if(tempoult> 200000L){  // o L é necessário, pois a variável tempoult
                          // foi declarada como long
  IncmmChuva();         // chamada da função com contador de chuva acumulada
  auxAgora = agora;
  }

}

// Função para armazenar chuva acumulada e zerar contador caso
// a quantidade de chuva acumulada seja > 1000 mm

void IncmmChuva(){

mmChuva =   mmChuva + 0.25;  // contador para armazenar chuva acumulada

if (mmChuva > 1000)
   mmChuva = 0;     // zera contador mmChuva caso > 1000 mm
}

// Função RespostaCom que recebe como parâmetro 1000 ms
void RespostaCom(int waitTime){
   long t=millis();   // armazena tempo decorrido em milisegundos desde que o
                      // processador foi inicializado
   char c;
```

```
    while (t+waitTime>millis()){
      if (Serial2.available()){  // Verificando se há dados disponíveis
                                 // para leitura no buffer
        c=Serial2.read();  // Lê o byte mais recente apontado no buffer
                           // de entrada da Serial2
        if (DEBUG) Serial.print(c); // imprime c no Serial Monitor
      }
    }

}

// Função para escrever os dados meteorológicos na plataforma do ThingSpeak

boolean EscreverThingSpeak(float value1, float value2, float value3, int value4, double
value5, float value6){
String cmd = "AT+CIPSTART=\"TCP\",\"";  // String para iniciar uma conexão TCP com um
//endereço, neste caso será para o IP da plataforma ThingSpeak
  cmd += "184.106.153.149";      // IP do servidor ThingSpeak
  cmd += "\",80";                // Número da porta de destino dos dados no endereço
184.106.153.149
  Serial2.println(cmd); // imprime string cmd na porta Serial2
                        //(selecionada para comunicação entre o Arduino e o ESP8266)
  if (DEBUG) Serial.println(cmd);
  if(Serial2.find("Error")){   // Verifica se no Buffer de entrada da Serial2 tem uma string =
//"Error", se sim, retorna TRUE, caso contrário, retorna FALSE
  if (DEBUG) Serial.println("AT+CIPSTART error"); // imprime "AT+CIPSTART error" no
//Serial Monitor caso tenha encontrado a string Error na Serial2
  return false;
  }

  String getStr = "GET /update?api_key=";   // String que ao final será composta pela
//instrução GET, chave do canal criado para receber os dados meteorológicos
 // e os valores dos sensores recebidos e processados pelo Arduino
  getStr += apiKey; // chave do canal
  getStr +="&field1=";
  getStr += String(value1);  // corresponde ao valor de temperatura
  getStr +="&field2=";
  getStr += String(value2); // corresponde ao valor de umidade relativa do ar
  getStr +="&field3=";
  getStr += String(value3); // corresponde ao valor de índice UV
  getStr +="&field4=";
  getStr += String(value4);  // corresponde ao valor de umidade do solo
  getStr +="&field5=";
  getStr += String(value5); // corresponde ao valor de chuva acumulada
  getStr +="&field6=";
  getStr += String(value6); // corresponde ao valor de Pressão Atmosférica
  // ...
  getStr += "\r\n\r\n";

  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length()); // Nesta etapa a string cmd é composta pelo comando
//AT+CIPSEND = e o tamanho da string getStr
  Serial2.println(cmd);                  // Imprime na porta Serial2 a string cmd
  if (DEBUG)  Serial.println(cmd);  // imprime no Serial Monitor a string cmd
  delay(100);
```

```
  if(Serial2.find(">")){
    Serial2.print(getStr);
    if (DEBUG)  Serial.print(getStr);
  }
  else{
    Serial2.println("AT+CIPCLOSE");   //Finalizar conexão TCP

    if (DEBUG)   Serial.println("AT+CIPCLOSE"); //Imprime no Serial Monitor que a conexão
                                                                    // TCP foi finalizada
    return false;
  }
  return true;
}

// Função para escrever os dados meteorológicos em um servidor web localizado no
localhost

boolean httpGet(float t, float u, float i, int us, double mC, float p)
{

String cmd = "AT+CIPSTART=\"TCP\",\""; // String para iniciar uma conexão TCP com um
endereço, neste caso será para o IP do localhost que está sendo utilizado através do
XAMPP = endereço IPv4
  cmd += "192.168.0.8";         // IP do localhost
  cmd += "\",8095";               // Número da porta de destino dos dados no endereço
                                  // IP do localhost
  Serial2.println(cmd);          // Imprime string cmd na porta Serial2
                                 // (selecionada para comunicação entre o Arduino e o ESP8266)
  if (DEBUG) Serial.println(cmd);   // Imprime no Serial Monitor a string cmd
  if(Serial2.find("Error")){        // Verifica se no Buffer de entrada da Serial2 tem uma string =
//"Error", se sim, retorna TRUE,caso contrário, retorna FALSE
    if (DEBUG) Serial.println("AT+CIPSTART error"); // imprime "AT+CIPSTART error" no
//Serial Monitor caso tenha encontrado a string Error na Serial2
    return false;
  }
  delay(1000);

// GET STRING

String getStr = "GET /arduino/salvardados.php?";  //String que ao final será composta pela
//instrução GET, pasta arduino que foi criada no XAMPP, o arquivo com script desenvolvido
// em php que irá inserir os valores dos sensores  para uma tabela criada
// no banco de dados e os valores das medições dos sensores
getStr += "T=";
getStr += String(t); // corresponde ao valor de temperatura
getStr +="&Um=";
getStr += String(u); // corresponde ao valor de umidade relativa do ar
getStr +="&IndUV=";
getStr += String(i); // corresponde ao valor de índice UV
getStr +="&UmSolo=";
getStr += String(us);  // corresponde ao valor de umidade do solo
getStr +="&mmChuva=";
getStr += String(mC);  // corresponde ao valor de chuva acumulada
getStr +="&P=";
getStr += String(p);  // corresponde ao valor de pressão atmosférica
```

```
getStr += "\r\n\r\n";
Serial.println(getStr);

Serial.println("\n");
delay(1000);

// ENVIAR DATA LENGHT

 cmd = "AT+CIPSEND=";
  cmd += String(getStr.length());   // Nesta etapa a string cmd é composta pelo comando
//AT+CIPSEND= e o tamanho da string getStr
  Serial2.println(cmd);                  // Imprime na porta Serial2 a string cmd
  if (DEBUG)  Serial.println(cmd); // imprime no Serial Monitor a string cmd

  delay(100);
  if(Serial2.find(">")){
    Serial2.print(getStr);
    if (DEBUG)  Serial.print(getStr);
  }
  else{
    Serial2.println("AT+CIPCLOSE");   //Finalizar conexão TCP

    if (DEBUG)   Serial.println("AT+CIPCLOSE"); // Imprime no Serial Monitor que a conexão
                                                                  // TCP foi finalizada
    return false;
  }
  return true;

delay(1000);

}
// setup
void setup() {

 DEBUG=true;          // Habilitar debug serial

 Serial.begin(9600);  // Configurando baud rate entre o Arduino e o computador

  // Verificando a conexão do sensor BMP280

 Serial.println("Testando modulo BMP280");

  if (!sensor_bmp.begin())
  {
    Serial.println("Sensor não encontrado. Verificar as conexoes!");
    while (1);
  }

 //

  pinMode(pinochuva, INPUT_PULLUP); // configura pinochuva como entrada e com resistor
// de PULLUP ativado
  dht.begin();          // inicializando sensor DHT22

  pinMode(analog, INPUT);     // Configura o pino analógico I como entrada
```

```
  pinMode(analogi,  INPUT);  // Configura o pino analogico zero como entrada


  Serial2.begin(115200);  // Configurando baud rate entre o Arduino e o ESP8266 ESP-01

  Serial2.println("AT+CWMODE=1");   // Definindo o ESP8266 como cliente

  RespostaCom(1000);  // Chamada da função RespostaCom

  Serial2.println("AT+CWJAP=\""+ssid+"\",\""+password+"\"");  // Comando para conectar o
//ESP8266 ESP-01 ao Access Point através do SSID  e da senha informada no início do
//programa
  RespostaCom(5000); // Chamada da função RespostaCom

  if (DEBUG) Serial.println("Completo");   // Imprime no Serial Monitor que a conexão foi
//estabelecida

  attachInterrupt(digitalPinToInterrupt(pinochuva), qtdchuva, RISING); // Interrupção do
pluviômetro

}
void loop() {

float t = 0;     // declaração da variável para receber o valor da temperatura
float h = 0;     // declaração da variável para receber o valor de umidade
float an_InUv = 0; // declaração da variável para receber o valor do pino analógico de índice
UV
float I_UV  = 0; // Recebe Valor de índice UV após tratamento dos dados
float mVolts = 0; // Recebe valor em mV, correspondente ao resultado do valor
                  //  recebido do conversor analógico digital para mV
float p = 0; // recebe valor de pressão atmosférica

p = sensor_bmp.readPressure(); // Leitura do valor de pressão
t = dht.readTemperature();     // Leiturra do valor de temperatura ambiente
h = dht.readHumidity();        // Leitura do valor de umidade relativa do ar

// Início verificação do valor de índice Uv

an_InUv = analogRead(analog); // Ler valor presente no pino analog configurado como
                              // entrada analógica

    // Coversor A/D do microcontrolador MEGA 2560 possui 10 bits de resolução
    // logo o valor retornado por an_InUv estará na faixa entre 0 e 1023
    // de acordo com o valor presente no pino.
    // Valor de tensão de referência para o conversor A/D = 5 V

    mVolts = (an_InUv * (5.0 / 1023.0)) * 1000;  // conversão do cálculo do conversor
//analógico digital para mV
   // Fim cálculo conversão para mV

   // Verificação do valor de índice uV a partir do valor calculado em mV

    if(mVolts > 0 && mVolts < 50)          I_UV = 0;
```

```
    else if(mVolts > 50 && mVolts <= 227)   I_UV = 0;

    else if(mVolts > 227 && mVolts <= 318)  I_UV = 1;

    else if (mVolts > 318 && mVolts <= 408) I_UV = 2;

    else if (mVolts > 408 && mVolts <= 503) I_UV = 3;

    else if (mVolts > 503 && mVolts <= 606) I_UV = 4;

    else if (mVolts > 606 && mVolts <= 696) I_UV = 5;

    else if (mVolts > 696 && mVolts <= 795) I_UV = 6;

    else if (mVolts > 795 && mVolts <= 881) I_UV = 7;

    else if (mVolts > 881 && mVolts <= 976) I_UV = 8;

    else if (mVolts > 976 && mVolts <= 1079) I_UV = 9;

    else if (mVolts > 1079 && mVolts <= 1170) I_UV = 10;

    else if (mVolts > 1170) I_UV = 11;

// Fim da verificação do valor de índice Uv

// Início verificação do valor de umidade do solo

    adc2_value = analogRead(analogi); // Ler valor presente no pino analogi configurado
//como entrada analógica

   // Verificação do valor de índice uV a partir do valor retornado por adc2_value

    if (adc2_value > 0 && adc2_value < Nivel_1)            // se adc2_value nessa faixa então
//determina nível de umidade 0
  {
    Serial.println("Status: Solo totalmente umido");
  }

  else if (adc2_value > Nivel_1 && adc2_value < Nivel_2)   // se adc2_value nessa faixa
//então determina nível de umidade 1
  {
    Serial.println("Status: Umidade media");
  }

  else if (adc2_value > Nivel_2 && adc2_value < Nivel_3)   //se adc2_value nessa faixa então
//determina nível de umidade 2
  {
    Serial.println("Status: Umidade minima");
  }

  else if (adc2_value > Nivel_3 && adc2_value < Nivel_4)   //se adc2_value nessa faixa então
//determina nível de umidade 3
  {
    Serial.println("Status: Solo secando");
```

```
  }

  else if (adc2_value > Nivel_4 && adc2_value < Nivel_5)   //se adc2_value nessa faixa então
//determina nível de umidade 4
  {
    Serial.println("Status: Solo quase seco");
  }

  else if (adc2_value > Nivel_5 && adc2_value < 1024)     //se adc2_value nessa faixa então
//determina nível de umidade 5
  {
    Serial.println("Status: Solo seco");
  }

  // Fim da verificação do valor de umidade do solo

        if (DEBUG) Serial.println("Temperatura="+String(t)+" *C"); // imprime no Serial Monitor
//o valor da temperatura
        if (DEBUG) Serial.println("Humidade="+String(h)+" %"); // imprime no Serial Monitor o
//valor da umidade relativa do ar
        if (DEBUG) Serial.println("IndiceUV="+String(I_UV)+""); // imprime no Serial Monitor o
//valor de índice Uv
        if (DEBUG) Serial.println("Umidade do Solo="+String(adc2_value)+""); // imprime no
//Serial Monitor o valor de umidade do solo
        Serial.println("Pressao="+String(p)+" Pa"); // imprime no Serial Monitor o valor de
//pressão atmosférica

        EscreverThingSpeak(t,h,I_UV, adc2_value, mmChuva, p);    // Chamada da função
//EscreverThingSpeak para escrever os valores na plataforma de armazenamento de dados
// na nuvem ThingSpeak

        delay(30000); // necessário aguardar pelo menos 30 segundos entre atualização dos
//valores para a plataforma

    // Leitura e processamento dos valores medidos pelos sensores novamente para
//sucessivo armazenamento no banco de dados MySQL

    p = sensor_bmp.readPressure(); // Leitura do valor de pressão
    t = dht.readTemperature();
    h = dht.readHumidity();

    an_InUv = analogRead(analog);

    mVolts = (an_InUv * (5.0 / 1023.0)) * 1000;

    if(mVolts > 0 && mVolts < 50)          I_UV = 0;

    else if(mVolts > 50 && mVolts <= 227)   I_UV = 0;

    else if(mVolts > 227 && mVolts <= 318)  I_UV = 1;

    else if (mVolts > 318 && mVolts <= 408) I_UV = 2;

    else if (mVolts > 408 && mVolts <= 503) I_UV = 3;
```

```
    else if (mVolts > 503 && mVolts <= 606) I_UV = 4;

    else if (mVolts > 606 && mVolts <= 696) I_UV = 5;

    else if (mVolts > 696 && mVolts <= 795) I_UV = 6;

    else if (mVolts > 795 && mVolts <= 881) I_UV = 7;

    else if (mVolts > 881 && mVolts <= 976) I_UV = 8;

    else if (mVolts > 976 && mVolts <= 1079) I_UV = 9;

    else if (mVolts > 1079 && mVolts <= 1170) I_UV = 10;

    else if (mVolts > 1170) I_UV = 11;

    adc2_value = analogRead(analogi);

    if (adc2_value > 0 && adc2_value < Nivel_1)
  {
    Serial.println("Status: Solo totalmente umido");
  }
   else if (adc2_value > Nivel_1 && adc2_value < Nivel_2)
  {
    Serial.println("Status: Umidade media");
  }
  else if (adc2_value > Nivel_2 && adc2_value < Nivel_3)
  {
    Serial.println("Status: Umidade minima");
  }

  else if (adc2_value > Nivel_3 && adc2_value < Nivel_4)
  {
    Serial.println("Status: Solo secando");
   }
  else if (adc2_value > Nivel_4 && adc2_value < Nivel_5)
  {
    Serial.println("Status: Solo quase seco");
   }
  else if (adc2_value > Nivel_5 && adc2_value < 1024)
  {
    Serial.println("Status: Solo seco");
  }

  httpGet(t,h, I_UV, adc2_value, mmChuva, p); // chamada da função httpGet para enviar os
//dados para o arquivo salvardados.php, o mesmo possui um código escrito em linguagem
de programação PHP e está localizado no XAMPP

  delay(30000);  // o código em PHP necessita de pelo menos 30 s para receber atualização
dos valores
}
//Fim do Programa
```

APÊNDICE B – Script em PHP

```
<?php   // Delimitador do código em PHP
        // dessa forma o servidor web entende que trata-se de um código
        // escrito em PHP
// Declaração de Variáveis
$servername = "localhost";  // endereço do servidor que o
                            // banco de dados está hospedado
$username = "root";    // nome do usuário cadastrado no banco de dados, sendo que
                       // quando a instalação do MySQL é realizada através
                       // do XAMPP, o usuário é criado por padrão como root

$password = "";     // por padrão quando a instalação do MySQL é realizada através
                    // do XAMPP não há senha para acessar o banco de dados
$db="estacao_meteorologica"; // Recebe o nome do banco de dados que será
                             // realizada a conexão
$conn = mysqli_connect($servername, $username, $password, $db); // A função
//mysqli_connect estabele a conexão com o banco de dados. A var $conn recebe o retorno
//da conexão se a conexão for estabelecida a variável $conn retorna TRUE

if (!$conn) {    // Caso o valor retornado seja diferente de verdadeiro
                 // a função  mysqli_connect_error  irá retornar  uma string representando
                 //o último erro que aconteceu com a última chamada a função mysqli_connect()
die("Connection failed: " . mysqli_connect_error());
}
// Através da declaração abaixo armazenada em $sql
// será inserido nos campos informados (`T`, `Um`, `IndUV`, `UmSolo`, `mmChv`, `P`)
// na tabela estação meteorológica os valores dos dados diretamente de uma URL que
possui o endereço desse arquivo em PHP (/arduino/salvardados.php?)
$sql="INSERT INTO `tabelaestacaometeorologica` (`T`, `Um`, `IndUV`, `UmSolo`, `mmChv`,
`P`) VALUES('".$_GET['T']."',  '".$_GET['Um']."',  '".$_GET['IndUV']."','".$_GET['UmSolo']."',
'".$_GET['mmChv']."', '".$_GET['P']."')";
$qry = mysqli_query($conn ,$sql); // Realizando uma consulta no banco de dados
?>
```